Information Extraction from Semi-Structured Websites

Colin D. Lockard

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Hannaneh Hajishirzi, Chair

Xin Luna Dong

Daniel Weld

Program Authorized to Offer Degree:

Computer Science and Engineering

University of Washington

Abstract

Information Extraction from Semi-Structured Websites

Colin D. Lockard

Chair of the Supervisory Committee:
Assistant Professor Hannaneh Hajishirzi
Computer Science and Engineering

The World Wide Web contains countless semi-structured websites, which present information via text embedded in rich layout and visual features. These websites can be a source of information for populating knowledge bases if the facts they present can be extracted and transformed into a structured form, a goal that researchers have pursued for over twenty years. A fundamental opportunity and challenge of extracting from these sources is the variety of signals that can be harnessed to learn an extraction model, from textual semantics to layout semantics to page-to-page consistency of formatting.

Extraction from semi-structured sources has been explored by researchers from the natural language processing, data mining, and database communities, but most of this work uses only a subset of the signals available, limiting their ability to scale solutions to extract from the large number and variety of such sites on the Web. In this thesis, we address this problem with a line of research that advances the state of semi-structured extraction by taking advantage of existing knowledge bases, as well as using modern machine learning methods to build rich representations of the textual, layout, and visual semantics of webpages. We present a suite of methods that will enable information extraction from semi-structured sources, addressing scenarios that include both closed and open domain information extraction and varying levels of prior knowledge about a subject domain.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

While completing a PhD takes effort on the part of the student, it takes even more dedication, patience, and wisdom on the part of the advisor. In Hanna Hajishirzi, I have been blessed to have an advisor who has inspired me not only as a researcher but also as a person. Hanna was a wonderful teacher, guide, and supporter in my research, but also created a community in our lab which helped all of us get through the many challenges of the past few years.

I have been doubly fortunate to have also worked under the supervision of Luna Dong throughout much of my PhD. Luna's incredible talent and spirit are obvious to everyone in our research community, but her ability to guide and inspire young researchers is equally as strong.

Dan Weld and Stephen Soderland were both instrumental parts of my first years at UW. I often mentally return to conversations from those early meetings, because while the technical specifics we were discussing may not be relevant to my current work, there were core lessons imparted about how to approach research that have stuck with me to this day. While Stephen is no longer with us, I am proud to be one of many researchers he trained or collaborated with who are each doing some small part to carry on his legacy.

While completing my studies at UW, I have been lucky to collaborate closely with and learn from many great researchers at Amazon, including Prashant Shiralkar and Arash Einolghozati. I would also like to thank my friends and labmates at UW, whose humor and brilliance made hard times easier and easy times incredibly fun. Among many others, they include John Thickstun, Robbie Weber, James Ferguson, Aida Amini, Dave Wadden, Yi Luan, Rik Koncel-Kedziorski, Sachin Mehta, Sewon Min, Akari Asai, Gabriel Ilharco, Bhargavi Paranjape, Yizhong Wang, Minjoon Seo, and Ellen Wu.

I have also been privileged to learn from many talented faculty over the years. This includes

both the NLP group at the Allen School, including Yejin Choi, Noah Smith, and Luke Zettlemoyer, as well as the CS department at Mills College, including Almudena Konrad, Ellen Spertus, and Susan Wang. I would also like to thank Meliha Yetişgen for serving as the GSR on my committee and providing helpful feedback.

Finally, I would like to recognize my parents, Craig and Kathy, and my late brother, Chris, whose support throughout my life has made possible anything worthwhile that I have achieved. Chris's example lives on in his sons Josh and Aaron. I hope someday my nephews will be able to understand how their father made the world a brighter place for everyone around him.

DEDICATION

to Stephen Soderland

Chapter 1

INTRODUCTION

The collection and organization of knowledge into a centralized and indexed form has been pursued for millennia. One early encyclopedia, Pliny the Elder's *Naturalis Historia*, claimed to distill knowledge about 20,000 topics collected from 2,000 volumes in multiple languages into a single set of books [80]. In recent years, many modern-day heirs to Pliny's tradition have worked together to create encyclopedic repositories of knowledge on the World Wide Web, from general knowledge encyclopedias such as Wikipedia[1] to domain-focused websites such as the Internet Movie Database[2].

While these collections of knowledge can be browsed and read by humans in much the same way *Naturalis Historia* was nearly two thousand years ago, albeit on digital screens and in hyperlinked form, recent years have seen an interest in representing factual knowledge about the world in the form of structured knowledge bases (KBs). These semantic networks, such as Freebase [11], consist of a large set of facts represented in well defined forms such as *(subject, predicate, object)* triples and can be traversed and queried by automated systems. Knowledge bases have recently been successfully applied to improve many applications including search, automated question answering, and personal assistants such as Amazon's Alexa [95].

Populating knowledge bases with a set of facts sufficient to serve downstream applications remains a challenge. The amount of collected human knowledge has grown greatly since Pliny the Elder's time, and it is beyond hope to expect any single person to attempt to compile the enumerable facts necessary to populate a rich knowledge base about even a single topic. We must instead turn to automated methods to construct such KBs from existing resources such as text.

---

[1] `www.wikipedia.org`

[2] `www.imdb.com`

The World Wide Web represents a globally accessible repository of documents created by people all over the world covering almost every imaginable topic. If the knowledge present in these documents could be transformed into a structured form, the goal of creating a truly comprehensive knowledge base could be realized. Researchers interested in extracting information from Web sources have pursued a variety of techniques for harnessing this data for over two decades [18]. The targets of such extraction are typically entities (such as persons, organizations, or cities), relationships between entities, and events involving one or more entities.

We can roughly divide work on information extraction from the Web into three main lines:

1. Work that focuses on semi-structured websites, where facts are presented in text fields embedded in a well-structured layout, with the layout of a given webpage generated from a template also used to create other pages of that site. This work typically harnesses the page-to-page consistency of layout to learn an extraction model that can parse the page structure of an individual website.

2. Work that focuses entirely on sentences of natural language text and attempts to learn lexical, syntactic, or semantic patterns used to describe a particular type of entity, relationship, or event.

3. Work that focuses on extraction from tabular data on the Web, which often relies on deducing the relations represented by matching a subset of the entities in the table to facts in an existing KB.

The primary weakness of existing work on extraction from semi-structured websites has been the reliance on structural features specific to a given website template, which means a new model has to be learned for every website targeted for extraction. This makes it challenging to scale solutions to the great number of semi-structured websites that exist. In its focus on structural features of website templates, this line of work has largely ignored the more generalizable features present in the semantics of text and the visual relationships between text fields, likely due to the fact that effectively understanding these kinds of meaning was non-trivial. However, recent years have brought great advances from the natural language processing community in building effective representations of textual semantics via word embedding systems such as BERT [24], while techniques such as graph neural networks and convolutional neural networks offer methods for representing visual and layout information.

In this work, we present a series of methods designed for extraction from semi-structured websites based on the utilization of these methods along with other insights gleaned from the various communities that have explored the topic of information extraction from the Web. Specifically, we present a suite of three methods, which make varying assumptions about the availability of prior knowledge and the type of extractions required. The first method, Ceres, assumes the existence of a large knowledge base well-aligned with the entities and relations presented on a website, and offers a distantly supervised approach to extracting additional facts from that site. The second method, OpenCeres, extends this approach to also extract new relation types not present in the original knowledge base. The third approach, ZeroShotCeres, removes the assumption of an existing knowledge base and instead learns a re-usable model that can be applied to previously unseen websites.

The rest of this document is structured as follows: Section 1.1 defines the overall problem along with relevant terms and ideas. Chapter 2 then provides an overview of the history of work on information extraction from the Web, first reviewing techniques specifically developed for semi-structured websites but then also addressing relevant methods targeting natural language text and multi-modal documents, which will all inform our work. Chapter 3 presents the Ceres system, Chapter 4 presents the OpenCeres system, and Chapter 5 presents ZeroShotCeres. Chapter 6 then concludes the discussion.

1.1 Problem Definition

We consider the problem of information extraction from *semi-structured websites* for the population of a *knowledge base*.

Semi-structured website: A semi-structured website W consists of a set of *detail pages* that

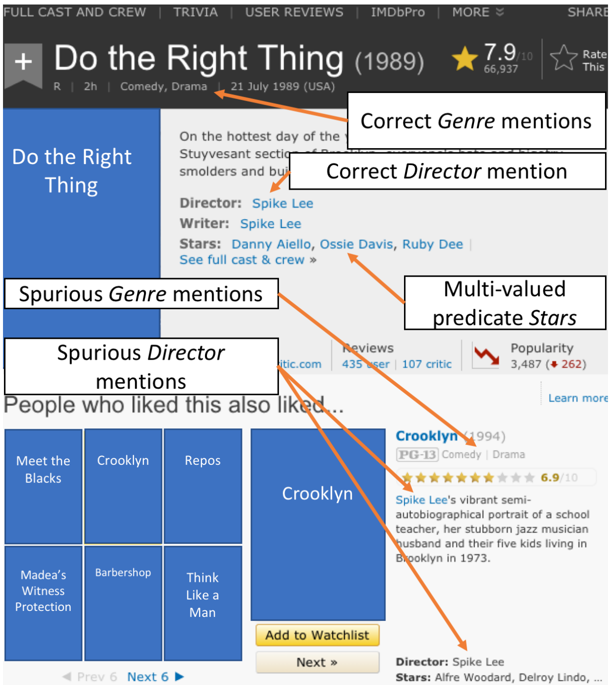

Figure 1.1: A cropped portion of the detail page from imdb.com for the film *Do the Right Thing*. Labels indicate some of the facts present as well as extraction challenges.

have been generated from a template or a set of templates, with the pages from the same template sharing a common structure and placement of information. Each page contains a set of text fields T. A detail page contains one or more facts about a particular entity, called the *topic entity* of the page, which we denote as e_{topic}. A semi-structured website typically belongs to a subject *vertical* V, where V is a general field of knowledge such as movies, finance, or sports. An example of a semi-structured website is the Internet Movie Database (IMDb), which contains thousands of pages about movies, with each page giving information such as the title, director, writer, release date, and stars of a particular film. An example page is shown in Figure 1.1. In this document we will focus on extraction from detail pages, not website entry pages or search pages; we consider it likely that every entry in an entry page corresponds to some detail page that provides more information.

While each page on a semi-structured website is expected to share a common structure, there may be minor differences from page to page. Some pages may be missing some fields, either

```
/html/body/div[1]/div/div[4]/div[3]/div[3]/div[3]/div[3]/div[4]/div[26]/b/a
/html/body/div[1]/div/div[4]/div[3]/div[3]/div[3]/div[3]/div[2]/div[10]/b/a
```

Figure 1.2: Example of absolute XPaths corresponding to the *acted in* predicate on two IMDb pages. They differ at two node indices, and the second path corresponds to the *producer of* predicate from the first page.

because that field does not exist (such as date of death for a person who is still living) or because the database underlying the website lacks the information. There may also be variations in structure because of variations in the number of instances of a predicate (for example, most films on IMDb have only a single director, but some films have multiple directors). In order to extract from a semi-structured website, an extractor must be robust to these changes in the placement of facts.

We can represent each webpage as a **DOM Tree**[3], in which each HTML tag becomes a node; a node in the tree can be uniquely defined by an absolute XPath [70], which describes the path from the root of the tree to the node. Absolute XPaths representing the same predicate on a semi-structured website tend to be similar but may have some differences. Figure 1.2 shows an example of absolute XPaths representing the "Acted In" predicate on IMDb. The upper path is from Oprah Winfrey's page whereas the lower path is from Ian McKellon's. On Winfrey's page, a "Producer" section exists in the section corresponding to the Actor section on McKellon's page, pushing her film performances down, altering the second-to-last index. Both performers act in many films, with the final index indicating the placement of the node in the film list. Notably, the upper XPath from McKellon's page does exist on Winfrey's page as well, but it represents her "producer of" relationship with the film *Selma*.

KB: A *knowledge base* (KB) is a store of facts corresponding to a given ontology. Facts are represented by *triples* of the form (s, r, o), where s is the *subject* of the relationship, r is the *relation predicate* according to our ontology, and o is the *object* of the relationship. For example,

[3] See https://www.w3.org/DOM/

a triple expressing the relationship between the film *Do the Right Thing* and its director *Spike Lee* would be *(Do the Right Thing, directed by, Spike Lee)*. Some predicates participate in a single triple with a subject, specifying a unique value (such as a birth date), while others are *multi-valued* and may participate in many triples with one subject, such as the predicate indicating an actor has acted in a film. An *ontology* defines the semantics of the relation predicates.

1.1.1 Relation Extraction

Our goal is to extract *(subject, relation, object)* knowledge triples, where the subject is the page topic e_{topic}, the object is a text field $t \in T$ containing the name of an entity (or atomic attribute value), and the relation indicates the relationship between the two entities. We consider both a setting where we are attempting to extract facts according to a pre-existing ontology ("ClosedIE") as well as the case where no ontology exists ("OpenIE").

Relation Extraction (ClosedIE): Let R define a closed set of relation types, including a special type indicating "No Relation". Relation Extraction is the assignment of each text field t to one $r_i \in R$, which indicates the relationship between the entity e_{object} mentioned in t and e_{topic}.

Open Relation Extraction (OpenIE): Given a pair of text fields (i, j), Open Relation Extraction is a binary prediction of whether i is a relation string indicating a relationship between the entity e_{object} mentioned in j and e_{topic}.

1.1.2 Zero-shot Extraction

While most prior work requires the learning of a model specific to the semi-structured website targeted for extraction, some of our work will look at zero-shot extraction. Given a semi-structured website W targeted for extraction, zero-shot extraction is the learning of a model without any use of pages from W during training. We consider two zero-shot settings:

Unseen-Website Zero-shot Extraction is the learning of a model without any use of pages from W, but with pages from some other website(s) from vertical V during training.

Unseen-Vertical Zero-shot Extraction is the learning of a model without any use of pages from

W or of pages from any website with vertical V during training.

Chapter 2
RELATED WORK

Information extraction from the Web has been a subject of interest from researchers in multiple areas of computer science, including natural language processing (NLP), data mining, and databases. We first examine the history of extraction from semi-structured sources, which until recently has focused on extracting by learning a set of rules to apply to a given HTML template, in Section 2.1. Section 2.2 considers approaches which rely only on natural language text, without considering layout or visual context. Section 2.3 considers recent work in multimodal extraction, in which researchers combine textual and visual signals from a document using state-of-the-art neural models. This set of work on multimodal extraction informs the OpenCeres and ZeroShotCeres approaches, which also aim to build rich representations of documents that unify the semantic information provided by these various signals. Finally, Section 2.4 describes some real-world attempts to apply web extraction technologies at scale.

2.1 Wrapper Induction

The most influential approach to information extraction from semi-structured websites is wrapper induction, first proposed by Kushmerick *et al.* [48]. A *wrapper* is a function that takes as input a document and produces as output the extraction of a particular type of content from that document, such as the object of a given relation type. Wrappers can be manually written, but this is a time-consuming process requiring technical expertise. *Wrapper induction* is the automatic learning of a wrapper based on a set of labeled examples. The challenge of wrapper induction is defining a learning process that can efficiently explore potential steps of an extraction function from a small number of labeled examples [47].

Vertex: In more recent work, a popular approach to wrapper induction has used XPaths as the wrapper function. XPath[1] is a domain-specific language for selecting values from XML documents, including related formats such as HTML. An XPath expression consists of a series of steps that describe a path through the document tree. These steps can include the use of functions and operators such as the ability to identify nodes with attributes matching particular strings of text or the selection of an ancestor, child, or sibling of a particular node.

One notable contribution to this line of work is the Vertex system [36]. Vertex relies on a three-step pipeline to enable training: First, the pages on a website are clustered to produce homogeneous sets of pages that correspond to a single template. Second, a subset of these pages is sampled to be manually annotated. This sampling process attempts to identify the pages that will be most informative for learning. Third, an XPath-based model is learned for each target predicate based on a set of local DOM features. This model is then applied to the remaining pages of the site for extraction.

Vertex's clustering approach first transforms each page into a set of shingles, or continuous sequences of HTML tags of a defined length. A min-hashing approach is then applied to these shingles to produce a fixed-dimension representation of the page. It has been shown that the distance between such min-hashed shingles approximates the Jaccard distance between each pair of pages' shingle sets [12]. A three-pass algorithm then identifies partially masked shingle vectors that cover large numbers of pages and assigns each page to a cluster.

For each resulting cluster, a set of pages is then sampled for annotation. This is accomplished with a greedy algorithm that approximates a maximum coverage search for the pages containing absolute XPaths that cover the largest number of the set of absolute XPaths seen across the cluster. An alternate approach to such sampling was proposed by Arasu *et al.* [3]. Their method attempts to identify equivalence classes of tokens mentioned on the page and produces a sample maximizing coverage over the largest and most common classes.

After obtaining manual annotations for pages in the sampled set, Vertex then applies a rule mining approach based on the Apriori algorithm [2]. They enumerate a set of local structural

[1]https://www.w3.org/TR/xpath/all/

features based on HTML tags, ID and class variables, and DOM sibling relationships and use these to identify an XPath function that covers the annotated set with high precision. With 20 training pages, Vertex reports nearly perfect accuracy across a set of six target relation types. While highly accurate, the weakness of the approach is its reliance on accurate labeled data for each cluster on which it is applied.

2.1.1 Automatic Extraction Approaches

Due to the need to have annotations for each website template for which a wrapper is to be produced, there have been numerous attempts to automatically induce wrappers based on some amount of outside prior knowledge. One approach toward reducing human effort is **DIADEM** [33]. Rather than requiring laborious annotations for each website cluster, DIADEM instead relies on humans to write type-specific joint entity and label recognizer functions. These functions attempt to identify objects of a relation by finding a pair of text elements that match defined rules for the relation and object strings. This reduces manual effort while maintaining an F1 above 0.9.

LODIE [34] represents a distantly supervised approach to the problem. This approach exploits the existence of Linked Open Data (LOD) in the form of the DBpedia KB [4]. This knowledge becomes a gazetteer which is used to automatically annotate instances of classes that appear on webpages. LODIE then assumes an extremely high degree of page-to-page consistency by ranking the XPaths of all annotated values and keeping only those with the most common path for each class. These are used in a wrapper induction process, producing extractions with F1 results from the mid 70s to the low 90s depending on subject domain. While LODIE is automatic, its strict assumptions limit both precision and recall when websites have more layout variation or present objects of a consistent class that are not actually related to the e_{topic}, such as the spurious mentions noted in Figure 1.1.

2.1.2 OpenIE

Until recently, the problem of Open Information Extraction (OpenIE) [8] from semi-structured sources had not been widely explored. One notable approach that does address an OpenIE setting is **WEIR** [13], which assumes a setting in which multiple websites exist about the same subject domain, with some overlap in the knowledge presented among the different sites. WEIR exploits this redundancy of data by presenting a generative model where an abstract set of relations and values is being exposed via partial views in the different sites. It accomplishes this by enumerating a set of XPath rules for each site, then keeps rules that provide both locally consistent extractions within the site but also produce a similar set of results to rules on the other sites. After completing this process, they then search for a potential predicate string by finding a string common to multiple sites that was also present as one step in the learned XPath function. While this marked an important step in considering an OpenIE setting, the accuracy of their predicate string identification varied from the mid-50s to 70s, leaving room for improvement.

2.2 Extraction from natural language text

Information extraction from unstructured text can be divided into two classes: In the first, the aim is to extract a single contiguous span of text and assign it a class label. The most prominent task in this class is named entity recognition (NER), and is most often formalized as a sequence tagging problem; sentiment classification would also fall into this category. The second class of problem is tasks that require understanding a relationship between two or more spans of text. The extraction of relations and events falls into this category, and more complex methods are necessary to identify, represent, and classify these relationships. I focus here on work related to textual representation and relation extraction, since that most relates to the methods presented in this thesis.

2.2.1 Representations of Unstructured Text

Natural language text can be considered as a sequence of discrete tokens, with the tokens corresponding to words, individual characters, or other sub-word components. In order to introduce either individual words or a full sequence as input to most machine learning models, it is necessary to transform it into a fixed dimensional representation. One method for this is with sparse binary vectors in which each position represents a word in the vocabulary. For a sequence, counts or weights such as TF-IDF [86] can be used at each position to represent the content of the sequence. The problem with these approaches is that they do not capture semantic similarity between words and do not represent the order of words in a sequence.

The past decade has seen the growing influence of pre-trained "word embeddings", in which a word is transformed into a dense vector representation. The initial work on word embeddings learned a single fixed vector for each word in a vocabulary; the two most prominent methods in this line are Word2Vec [62] and GloVe [78]. For example, Word2Vec uses a language-modeling-like objective to learn to map words that appear in similar contexts to the same region of the vector space. These techniques have two major weaknesses: First, they are unable to usefully represent any words that were not in the training vocabulary; and second, they produce a single representation for each word, despite the fact that words can have different meanings due to homonymy and context. The first of these deficiencies was addressed with methods such as FastText [10], which learns representations of sub-word sequences of characters and can then represent a word as the sum of is constituent subsequences. A more general method is to tokenize at a sub-word level with byte-pair encoding, which starts with a vocabulary of individual characters and iteratively adds new symbols consisting of the concatenation of pairs of symbols in the vocabulary that most frequently appear together in a corpus [88].

The second problem was tackled with the invention of contextual word embeddings. The first of these methods, ELMo [79], uses a multilayer bidirectional LSTM [39] to build a representation of each word based on the sentence in which it appears. A language modeling objective is used to train the system. Unlike the methods described above, with ELMo, the output of the learning process is not a simple fixed embedding for each item in the vocabulary, but is rather the full learned weights of the BiLSTM. This idea was expanded on with BERT [24], in which the BiLSTM is replaced with a transformer [93], and a masked language modeling objective allows the full sentence context to simultaneously inform the representation.

These language model-based embedding methods have allowed researchers to leverage large amounts of unlabeled data to learn useful representations of words in a self-supervised process, with the ELMo and BERT experiments showing the utility of these representations in many downstream tasks. The LSTM and Transformer architectures used by ELMo and BERT, respectively, can also be used to build representations of full sequences. With LSTMs, the final hidden layer can be used for a classification objective; transformers can accomplish this introducing a special dummy token (typically referred to as "CLS") used for classification.

2.2.2 Relation Extraction from Unstructured Text

Similarly to researchers addressing information extraction from semi-structured sources, NLP researchers addressing the relation extraction problem have faced two key challenges: How to model their problem and how to obtain enough training data to learn an effective model. One of the most important developments in relation extraction has addressed the latter of these issues. **Distant supervision** [63] is a method for automatically creating training data based on an existing knowledge base. While distant supervision can be naively applied by simply assuming that all sentences containing mentions of a pair of entities that are known to participate in a relation express that relation, most methods attempt to more accurately assign labels, for example by maximizing a joint probability of a set of labels given the contexts in which they appear [40].

In terms of modeling the relation extraction problem, earlier work relied on pipelined systems in which features were extracted from sentences, potential entities were identified, and entity pairs were considered for classification. More recent work has often jointly modeled entity and relation extraction with end-to-end systems. One notable recent example is **DyGIE** [59]. DyGIE first processes a sentence with a BiLSTM (using GloVe and ELMo embeddings), and then creates a graph in which spans of text (enumerated up to a given length) become the nodes; edges represent probabilities of co-reference and predicate relations. An iterative process then models co-reference relations and relation extraction by repeatedly propagating information between nodes; the use of a beam reduces the propagation to the likeliest neighbors at each step. While this setting is very different from that of semi-structured webpages, the use of a graph to model relationships between potential relation participants could be applied in that setting as well.

2.3 Multi-modal extraction

I consider *multi-modal extraction* to encompass approaches that richly combine semantic signals present in multiple modalities, from the meaning of the text itself to the layout relationships between text fields to the visual aspects of the text such as font size, color, and boldness. Prior work on semi-structured extraction, as described in Section 2.1 above, did not generally attempt to model complex textual semantics since that line of work pre-dated the development of modern word embeddings. Meanwhile, work on unstructured text has largely ignored any aspects of documents that can't be contained in a simple `.txt` file. Recently some researchers have begun to unify some of the intuitions from these separate fields to consider visually complex documents more holistically.

2.3.1 BlingKPE

Addressing the problem of key-phrase extraction from webpages, Xiong *et al.* [100] present one of the first methods to combine sophisticated semantic representations of textual content with the visual features used to display that text. Key-phrase extraction is the task of discovering the phrases in a document that will be useful in downstream tasks such as indexing for search. The intuition the authors bring is that the most important textual content on the page for purposes such as this is likely to be represented with prominent font attributes, as well as in a location nearer to the top of the page, such as a title.

To represent the actual textual semantics, the approach first runs ELMo [79] over the textual content of the page, obtaining contextual embeddings for each word. Each ELMo embedding is then concatenated with an additional feature vector representing visual and layout attributes of that word. This vector contains features such as the font size, font boldness, and the x/y coordinates of the word on the page. The combined features are then fed into a transformer architecture [93] for extraction.

The authors compared the implementation of their technique, BlingKPE, with several text-only baselines for key-phrase extraction, from a simple TF-IDF approach to a recurrent neural network-based approach. Training data was obtained using weak supervision from web search logs, assuming that text on clicked pages that matched query strings could be considered as key phrases. The addition of visual information allowed BlingKPE to post a 42% improvement over these baseline in Precision@1, with a 29% improvement in Recall@1.

By combining modern contextual representations of textual semantics with visual attributes of that text, BlingKPE represents a step toward representing documents with rich visual structure. However, the integration of text and visual features is quite shallow, obtained via a simple concatenation of vectors. In addition, the task of key-phrase extraction is simpler than relation extraction since it involves only a binary classification of each word.

2.3.2 CharGrid

An important attempt to unify visual and textual aspects of documents came with CharGrid [42]. Driven by the success of computer vision algorithms, CharGrid takes as input a scanned copy of a document and operates over it as a two-dimensional grid of pixels. However, the authors note that attempting to operate purely at the pixel level would throw out the large amount of prior information carried in structures as simple as characters, forcing the model to learn to recognize characters from scratch, and from there continue on to learn task-specific textual features.

Rather than working simply on a grid of pixels with associated color values, the approach instead relies on an initial step that runs an optical character recognition (OCR) model over the document, identifying the characters corresponding to each printed letter. The color values of the pixels are then replaced with an ID representing the character occupying that pixel location (with a special value indicating blank regions of the page where OCR did not detect a character).

This 2-D grid of character values, or "CharGrid" is then given as input to a convolutional neural network architecture, specifically an approach based on the VGGNet model popular in computer vision [89]. The model produces an encoded representation of the document, which is then passed to two different decoders: The first of these, which they call the "semantic segmentation" decoder, assigns a class label to each character indicating the relation or attribute class of that piece of text.

The second decoder performs “bounding box regression”, which groups together multiples pieces of text of the same class.

The authors conduct experiments on a corpus of scanned receipts and invoices, attempting to extract eight attribute classes including invoice number, invoice amount, vendor name, and line-item description. Using 10,000 documents for training, they obtain accuracies ranging from 48% to 84% for different attributes.

While this approach represents a step forward in representing layout relationships between different text fields in a document, it has several weaknesses. Most notably, there is no attempt to model textual semantics at a level higher than the character. The model has no prior knowledge of words or contextual word meanings such as that provided by embedding models such as Word2Vec [62] or BERT [24]. In addition, while their approach does capture spatial relationships between text fields as well as font size, they do not capture visual elements of the text such as font color, typeface, or boldness, since their approach simply defines a bounding box around a character and assigns a character value to all pixels within that box.

2.3.3 GraphIE

Qian *et al.* [81] present GraphIE, a graph neural network-based approach to information extraction; a similar approach was contemporaneously published by Liu *et al.* [55]. GraphIE can be used on any text where additional contextual information can be represented in the form of a graph; experiments are conducted using co-reference graphs of entity mentions in a purely textual document, social graphs connecting users and the text of their accompanying tweets, and most relevant for our problem, a graph-based representation of a visually rich semi-structured document.

In this third setting, their dataset is a corpus of medical documents describing patients with adverse drug reactions. Much like my own work described in Chapter 5, Qian et al. first construct a graph in which each text field on the page becomes a node, and nodes are connected with edges indicating physical adjacency on the page. Four edge types are used, indicating that the connected node is above, below, to the left, or to the right of the source node. A bidirectional LSTM [39] is then used to produce a representation of the text in the text field, with the final hidden states of the

BiLSTM becoming the initial set of features for the node. GloVe embeddings [78] for each word provide the input to the LSTM.

These initial node representations and layout graph become the input to a R-GCN [87], which then enriches the representation of each text field with contextual information about its local neighborhood in the graph. The final hidden state representation of each node (text field) produced by the R-GCN is then passed as the initial hidden state to a decoder which will assign a class label to each word in the text field. This decoder consists of a BiLSTM (with the word-by-word output of the encoder BiLSTM serving as input), with a CRF layer [49] on top to produce the final predictions.

The experiments are conducted on a set of scanned semi-structured medical reports, with OCR being applied in a pre-processing step to obtain the text. The target extraction classes are a set of eight attribute types including patient initials, age, and birthday, as well as drug name and the city where the incident was reported. Average F1 across the eight classes was 0.81, a 1% improvement over a baseline that encoded using only a BiLSTM over the text without the extra context from the graph. This is a surprisingly small gain, but most of the extraction targets are likely to have been the only text of their type (such as a date) on the page. A further experiment showed that F1 was much lower (0.33) when tested on documents whose templates were not in the training set.

2.3.4 Fonduer

Fonduer [99] is an extension of the Snorkel [83] approach to a semi-structured setting. Snorkel is an application of the data programming paradigm, which has similarities to the distant supervision approach described previously, as well as the DIADEM extraction approach. In this approach, the goal is to quickly create training data by having a human create a set of labeling functions to noisily label a large dataset. A generative model is then used to "de-noise" the data and produce training annotations with confidences attached. Fonduer provides an interface for easily writing these labeling rules using visual layout relationships between text. The authors additionally propose an extraction model that concatenates a set of textual, structural, visual, and layout features. The experiments show extraction quality varying greatly by domain. Like Bling-KPE, the proposed extraction method is a “shallow” combination of features.

2.4 Applied Large-scale Web Extraction

There have been several attempts to apply information extraction techniques at web scale to build enormous knowledge bases. Most of these, such as PROSPERA [67] and DeepDive [69] focused on extraction from natural language text. One such system, NELL [17] used human-in-the-loop learning to refine its models over time. After running for five years, NELL amassed a collection of 80 million beliefs [65]. The project most relevant to this proposal is Google’s Knowledge Vault [25], which used a suite of distantly supervised systems to extract from natural language text, the DOM trees of semi-structured webpages, webtables, and Semantic Web annotations. In extracting over a billion total facts, the researchers found that 75% of extracted facts and 94% of the high-confidence facts were found in DOM trees. However, the overall precision of extractions was in the low 60s, implying much room for improvement. This provides motivation for the real-world utility of the methods proposed in this document.

Chapter 3

DISTANTLY SUPERVISED RELATION EXTRACTION WITH CERES

3.1 Introduction

Knowledge bases, consisting a large set of (subject, predicate, object) *triples* to provide factual information, have recently been successfully applied to improve many applications including Search, Question Answering, and Personal Assistant. It is critical to continuously grow knowledge bases to cover long tail information from different verticals (*i.e.*, domains) and different languages [51], and as such, many researchers have worked on automatic knowledge extraction from the Web [25, 23, 34, 33, 17].

Among various types of Web sources, we argue that semi-structured websites (*e.g.*, IMDb, as shown in Figure 1.1) are one of the most promising knowledge sources. The Knowledge Vault project [25] reported that after applying automatic knowledge extraction on DOM trees of semi-structured websites, texts, web-tables, and semantic web annotations, 75% of the extracted facts and 94% of the high-confidence facts were covered by DOM trees. This is not surprising: semi-structured websites are typically populated by data from large underlying databases, thus containing richer information than webtables and web annotations; the underlying databases often focus on factual information, making it more suitable as knowledge sources than free texts.

Despite the huge opportunities, automatic knowledge extraction from semi-structured sources has not received the attention it deserves from the research community. Unlike natural language text, semi-structured sources provide (arguably) more structure, but in a different way, with fewer well-developed tools for tasks like named-entity recognition and entity linking. Unlike webtables [15], semi-structured data lack the table structure (rows and columns) that helps identify entities and relations. Semi-structured sources also present challenges for automatic extraction because the structure, or layout, differs from website to website, so the extraction model trained for one website cannot be used for another. Even between webpages that are generated from the same template, pages may differ due to missing fields, varying numbers of instances, conditional formatting, and ads and recommendation strategies.

Traditional DOM extraction typically uses *wrapper induction*: given a website, wrapper induction asks for manual annotations, often on only a handful of pages, and derives the extraction patterns, usually presented as XPaths, that can be applied to the whole website [48]. Although wrapper induction has been quite mature in terms of extraction quality, obtaining precision over 95% [36], it requires annotations on every website, an expensive and time-consuming step if one wishes to extract from many websites.

In order to create an automated process requiring no human annotation, distant supervision has been proposed for text extraction [63]. In this process, training data are generated automatically by aligning sentences with existing seed knowledge bases using simple heuristics; that is, if a sentence mentions two entities of a triple in the knowledge base, the sentence is annotated to assert the relation in the triple. Although the training data can be noisy, the large number of annotations on a large corpus of texts still enables learning fairly robust extraction models.

Unfortunately, distant supervision does not easily apply to semi-structured websites. This is mainly because the unit for annotation changes from a sentence to a webpage, normally containing much more information. The sheer volume of text on a webpage first poses efficiency challenges: a single web page may contain mentions of hundreds of entities, which can match to thousands of potential candidate entities; examining relations between every pair of candidate entities can be very expensive. Even worse, the large number of entities mentioned in a webpage may cause many spurious annotations, leading to low-quality extraction models. Indeed, a recent attempt to apply distantly supervised extraction on semi-structured data obtained quite low accuracy: Knowledge Vault trained two distantly supervised DOM extractors but their accuracy is only around 63% [26].

In this chapter, we present CERES[1], a knowledge extraction framework that improves the distant supervision assumption for DOM extraction. The key underlying idea is to best leverage the

[1] Ceres is the Roman goddess of the harvest.

unique characteristics of semi-structured data to create fairly accurate annotations. First, we focus on detail pages, each of which describes an individual entity, and propose an automatic annotation algorithm that proceeds in two steps: it first identifies the *topic entity*, which is the primary subject of a page, and then annotates entities on the page that are known (via the seed KB) to have relationships with that entity. This reduces the annotation complexity from quadratic to linear and drops a significant number of spurious annotations. Second, we leverage the common structure between key-value pairs within a webpage, and across webpages in a website, to further improve annotation quality.

Our new extraction method significantly improves extraction quality. On the SWDE dataset [38], which has been used as a benchmark for DOM extraction, we are able to obtain an average accuracy of over 90% in various verticals, even higher than many annotation-based wrapper induction methods in the literature. Large-scale experiments on over 400,000 pages from dozens of multilingual long-tail websites harvested 1.25 million facts at a precision of 90%, with the ratio of 1:2.6 between annotated popular entities and extracted entities that include many long-tail entities.

This chapter makes the following contributions.

1. We describe CERES, an end-to-end distantly supervised knowledge extraction framework that can apply to semi-structured websites independent of domains and web sources.

2. We propose an advanced distant supervision annotation process for semi-structured web sources by leveraging the unique structural characteristics of such data.

3. In addition to showing the efficacy of our technique on a benchmark dataset, we include an evaluation on dozens of real-world long-tail websites containing multi-valued predicates to show applicability of our method in the wild.

We note that this approach can be combined with a manual-annotation-based approach: When entering a new domain for which no KB exists, we can use an annotation-based extractor on a few prominent sites to populate a seed KB, and then use the KB for distantly supervised extraction on other sites.

The rest of the chapter is structured as follows. Section 3.2 formally defines the problem and overviews our solution. Section 3.3 describes data annotation and Section 3.4 describes training and extraction. Section 3.5 presents our experimental results. Section 3.6 reviews related work and Section 3.7 concludes.

3.2 Problem Definition and Solution Overview

We introduce the problem in this section and illustrate the challenges (Section 3.2.1). We then review distant supervision (Section 3.2.2), and give an overview of our solution (Section 3.2.3).

3.2.1 Problem definition

KB: We assume a setting in which we have an existing seed *knowledge base* (KB) populated with at least some facts corresponding to a given ontology. Facts are represented by *triples* of the form (s, r, o), where s is the *subject* of the relationship, r is the relation *predicate* according to our ontology, and o is the *object* of the relationship. For example, a triple expressing the relationship between the film *Do the Right Thing* and its director *Spike Lee* would be *(Do the Right Thing, directed by, Spike Lee)*. Some predicates participate in a single triple with a subject, specifying a unique value (such as a birth date), while others are *multi-valued* and may participate in many triples with one subject, such as the predicate indicating an actor has acted in a film. The *ontology* defines the semantics of the relation predicates. Our goal is to further populate the KB with facts obtained from semi-structured websites.

Semi-structured website: A *semi-structured website* W consists of a set of *detail pages* that have been generated from a template or a set of templates, with the pages from the same template sharing a common structure and placement of information. Each detail page contains one or more facts about a particular entity, called the *topic entity* of the page. An example of a semi-structured website is the Internet Movie Database (IMDb)[2], which contains thousands of pages about movies, with each page giving information such as the title, director, writer, release date, and stars of a

[2] www.imdb.com

particular film. An example page is shown in Figure 1.1. In this chapter we are only interested in extraction from detail pages, not website entry pages or search pages; we consider it likely that every entry in an entry page corresponds to some detail page that provides more information.

While each page on a semi-structured website is expected to share a common structure, there may be minor differences from page to page. Some pages may be missing some fields, either because that field does not exist (such as date of death for a person who is still living) or because the database underlying the website lacks the information. There may also be variations in structure because of variations in the number of instances of a predicate (for example, most films on IMDb have only a single director, but some films have multiple directors). In order to extract from a semi-structured website, an extractor must be robust to these changes in the placement of facts.

We represent each webpage as a **DOM Tree**[3]; a node in the tree can be uniquely defined by an absolute XPath, which for brevity we simply refer to as an **XPath** [70]. XPaths representing the same predicate on a semi-structured website tend to be similar but may have some differences. Figure 1.2 shows an example of XPaths representing the "Acted In" predicate on IMDb. The upper path is from Oprah Winfrey's page whereas the lower path is from Ian McKellon's. On Winfrey's page, a "Producer" section exists in the section corresponding to the Actor section on McKellon's page, pushing her film performances down, altering the second-to-last index. Both performers act in many films, with the final index indicating the placement of the node in the film list. Notably, the upper XPath from McKellon's page does exist on Winfrey's page as well, but it represents her "producer of" relationship with the film *Selma*.

Problem Definition: Our goal is to identify the DOM nodes of a detail page containing facts corresponding to particular predicates in our ontology.

Definition 3.2.1 (DOM Extraction) *Let W be a semi-structured website and K be a seed knowledge base on a particular vertical.* DOM extraction *extracts a set of triples from W such that the subject and object of each triple is a string value on a page in W, and the predicate appears in K and indicates the relation between the subject and object as asserted by the webpage.*

[3]See https://www.w3.org/DOM/

A successful extraction shall extract *only* relations that are asserted by the website (high precision) and extract *all* relations that are asserted (high recall). We consider only predicates in the ontology, for which we can obtain training data from K. Note that we resort to existing work to verify the accuracy of the website's claims [27] and to tackle the problem of entity linkage between the extracted data and existing entities in the knowledge base [28]. We observe that most entity names correspond to full texts in a DOM tree node, so for simplicity we leave identifying entity names as substrings of texts in a DOM node or concatenation of texts from multiple DOM nodes to future work.

Finally, we clarify that even in the same website, semi-structured webpages may be generated by very different templates; for example, on IMDb, movie pages and people pages appear quite different. We first apply the clustering algorithm in [36] to cluster the webpages such that each cluster roughly corresponds to a template, and then apply our extraction algorithm to each cluster.

3.2.2 Review of distant supervision

The key challenge in extracting knowledge from the semi-structured web without human intervention is to automatically generate training data for each website; *distant supervision* has been proposed for this task. Distantly supervised relation extraction is a process that first aligns a knowledge base to an unannotated dataset to create annotations, and then uses the annotations to learn a supervised extraction model. Distant supervision was initially proposed for relation extraction from natural language text and in that context relies on the *distant supervision assumption*: if two entities are known (via the seed KB) to be involved in a relation, any sentence containing both entities expresses that relation [63]. Because this is obviously not true in all cases, much of the work in distantly supervised extraction has been devoted to learning effectively despite the noisy labels generated by this assumption.

The distant supervision assumption is harder to apply in the DOM setting for three reasons. First, unlike natural language text, where the basic unit of analysis is a sentence that usually consists of no more than a few dozen words and mentions only a few entities, in DOM extraction the basic unit of analysis is a web page. Web pages may contain thousands of text fields, and many real-world use cases will mention hundreds or thousands of entities, which poses challenges both at annotation time and at extraction time.

To understand the difficulties in successfully annotating relation mentions in this setting, consider the IMDb page of prolific voice actor Frank Welker[4]. Welker's credits list over 800 films and TV series in which he has performed. The page also lists Welker's place and date of birth, characters he has portrayed, television episode titles, and various other entities and attributes, which may exist in our KB. This single page contains legitimate mentions of several thousand entities which may have relationships with Frank Welker, with some of them mentioned multiple times.

Second, the situation is further complicated by the fact that strings appearing on a web page may correspond to multiple entities in our KB, sometimes many entities. For example, the word "Pilot" is the name of thousands of television episodes, since it is used for the first episode of a series. In addition, other strings may incidentally match entities in our KB; for example, Welker's page has a section labeled "Biography", which also happens to be the title of a TV series. Other sections of the page may reference entities unrelated to the topic entity; for example, the detail page about the film *Do the Right Thing* shown in Figure 1.1 also mentions information about *Crooklyn*, including several of its cast members.

The result of this setting is that a single web page may contain strings corresponding to tens of thousands of entities in our KB, and hundreds or thousands of those matches may even be correct. The distant supervision assumption would have us check our KB for relationships involving all pairs of entity mentions, and if a relation was found, would have us produce an annotation for that predicate for those fields of the page. This approach is both computationally challenging, due to the immense number of potential entity pairs, as well as likely to produce noisy annotations due to the spurious matches made likely by the huge number of potential entities.

Finally, because the distant supervision assumption does not incorporate the knowledge that a detail page contains many relationships involving the same subject (the topic entity of the page), it involves labeling pairs of nodes as expressing a triple. At extraction-time, it will then be necessary

[4]`http://www.imdb.com/name/nm0919798/`

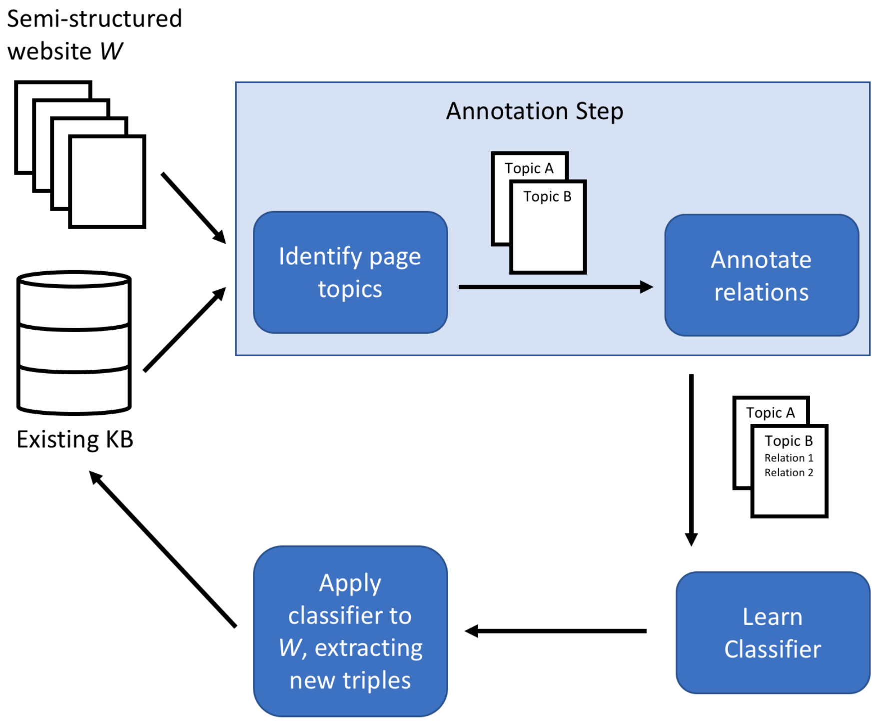

Figure 3.1: CERES architecture.

to consider all pairs of nodes containing potential entity mentions as targets for extraction. However, unlike in NLP, pre-trained NER systems are not available in the DOM setting, and thus it is not obvious how to select these nodes. Considering all possible pairs of nodes on a page is computationally infeasible, while relying on the KB for entity matching, as we do at annotation-time, would limit the ability to produce extractions involving previously unknown entities.

3.2.3 Overview of our approach

We propose CERES, a new method for automatically extracting information from semi-structured webpages. CERES makes use of an existing knowledge base to automatically label facts on the webpages that can be mapped to known facts in our KB. These annotations are then used to train a probabilistic extractor that extracts additional facts from the webpages. Our process thus involves an annotation step, a training step, and an extraction step, as shown in Figure 3.1. There are three key differences between our approach and a traditional approach to distant supervision.

First, we rely on a modification of the distant supervision assumption, which we term the *Detail Page Distant Supervision Assumption*.

Definition 3.2.2 (Detail Page DS Assumption) *Let w be a detail page, and $(s, r, o) \in K$ be a knowledge triple. If s is the topic entity for w and w contains object o, it is assumed that at least one mention of o on w expresses the relation r.*

According to this assumption, we use a two-step annotation process in which we first attempt to identify the topic entity of a page, and then annotate relations.

Second, text allows great flexibility in expression, with sentences structured differently from document to document. In contrast, webpages in a semi-structured website are expected to share a common structure. As we annotate, we consider both *local* evidence within a page, and *global* evidence across pages to improve annotation quality.

Third, traditional distantly supervised extraction relies on NER techniques to identify entities, and uses syntax and lexical features to predict relations between every pair of entities in the same sentence. As we extract knowledge from a detail page, we consider only relations between the topic entity and other entities on the page. We use the (possibly noisy) annotations to train a machine learning model that classifies a DOM node, represented by an XPath, on a webpage, and outputs a relation between the topic entity and the entity represented by the node (either a predicate present in the training data or an "OTHER" label indicating that there is not a relation present in our ontology).

We admit that in this way we will not be able to extract relations between non-topic entities, such as the *directed by* relation between the *Crooklyn* movie and its director in Figure 1.1. However, we believe such relations, if important, are likely to be covered by another detail page with the subject as the topic entity (*i.e.*, the webpage for the *Crooklyn* movie).

3.3 Automatic Annotation

The purpose of the annotation process is to produce a set of examples that will be used to train a supervised extractor. With that in mind, we place a high value on maintaining the precision of our annotations even if it prevents us from annotating some pages. To maintain high precision, we base our annotation process on three observations.

1. **Consistency:** Detail pages on a website often have roughly the same format, with information expressed in a similar way across pages.

2. **Informativeness:** Each detail page typically provides multiple attributes of an entity, and the values are typically fairly diverse among entities (*i.e.*, it seldom happens that all entities across the website share the same object for a predicate).

3. **Uniqueness:** An entity is typically described by a single detail page or at most a couple of detail pages (some with more data in particular aspects).

These observations will guide us in our annotation process. We re-state the two key differences from annotation in traditional distant supervision. First, we will identify the topic entity for each page, before annotating all objects of predicates shown on the page. Second, in both steps we will harness the semi-structured nature of the data by relying on global information gathered across all pages of a website in addition to the local information on each page.

3.3.1 Topic Identification

Given a set of n webpages $w_1, ..., w_n$ from a semi-structured website W, our first step is to identify the topic entity of each page when it applies.

3.3.1.1 Local Topic Candidate Identification

Local candidate identification proceeds in two steps: it first identifies all entities on the page, and then finds the one that is most likely to be the topic entity. Our intuition is that since the purpose of the detail page is to state facts about the topic entity, more strings on the page should bear a relation to the topic entity than to other entities.

Step 1. Entity identification. We begin this process by attempting to identify all entities that may be listed on the page. Each text field on the webpage is matched against the KB using fuzzy string matching process presented in [37]. This yields a set of all entities in the KB that may be mentioned on the page. We will refer to this set of m potential entities on w_i as the $pageSet$. As noted previously, this could contain thousands or tens of thousands of entities, each of which is a potential topic entity of the page.

According to our *Uniqueness* observation, we automatically compile a list of strings appearing in a large percentage (*e.g.*, 0.01%) of triples and do not consider them as potential topics. In addition, we discard strings with low information content, such as single digit numbers, years, and names of countries.

Step 2. Topic candidate identification. For each entity s_j in $pageSet$, we then look in the KB to get a set of entities that are the object of a relation in which $s_j, j \in [1, m]$, is the subject; we will refer to this set as $entitySet_j$. We then compute the Jaccard similarity between $pageSet$ and $entitySet_j$ for each $e_j \in pageSet$:

$$J(pageSet, entitySet_j) = \frac{|pageSet \cap entitySet_j|}{|pageSet \cup entitySet_j|} \tag{3.1}$$

The initial topic candidate c_i for page w_i is then the entity in $pageSet$ with maximum Jaccard similarity:

$$c_i = \arg\max_j J(pageSet, entitySet_j) \tag{3.2}$$

Note that if the topic entity of a page does not appear in our KB, we cannot hope to have identified it at this point, but given the volume of text on a typical webpage, it is likely that there was a spurious match against some entity in the KB. Even if the topic is in our KB, it is possible that another entity was identified as the candidate topic if we had little information about the true topic in the KB. This motivates us to further apply a filtering step before assigning topics leveraging global formatting patterns.

3.3.1.2 Global topic identification

Our global topic identification step leverages the three observations we have made on semi-structured websites.

Step 1. Topic filtering by uniqueness: According to our *Uniqueness* observation, if one entity is identified as the topic of many pages, it is unlikely to be the true topic. For example, if the word "Help" appears on every page and happens to match an entity in our KB, it may be identified as a candidate topic for many pages where the real topic does not exist in the KB. To counteract this problem, we discard any topic identified as a candidate topic of a large number of pages (*e.g.*, ≥ 5 pages).

Step 2. Finding the dominant XPath: Our *consistency* observation tells us that the text field containing the name or identifier of the topic entity should be in roughly the same location from page to page. We use this intuition to examine all candidate topics to try to deduce the areas of the page where they are most likely to occur.

For each page w_i, the XPaths to all mentions of topic candidates are collected (note that the candidate topic may appear in multiple text fields on the page). For all candidate topics across the entire website W, we produce counts of how often each path appears. These paths are then ranked in descending order by count. If a path is highly ranked, there are many pages in which the candidate topic is found at that location on the page.

We then re-examine each page w_i and find the highest-ranked path that exists on the page. For all potential entities j mentioned in that text field, the entity with the highest score $J(pageSet, entitySet_j)$ as previously calculated is taken as the topic entity of the page.

Step 3. Webpage filtering by informativeness. Finally, according to our *informativeness* observation, we only consider webpages with a predetermined minimum number of relation annotations (*e.g.*, ≥ 3); otherwise no topic is chosen for that page and it is discarded from the annotation process.

Parameter setting. We note that at a few places we need to set parameters such as when to discard a topic if it is identified as the topic entity for many pages, and when to discard a webpage if the number of annotations is too small. Our goal is to filter obvious noise and expect our learning algorithm to be robust to remaining noise. We thus set such parameters according to our empirical observations and tend to set a small value.

3.3.2 Relation annotation

Topic identification finds a topic for a set of webpages; at this point, we have retrieved from the KB all triples related to the topic entity and located all potential mentions of their objects on the page. The next step is to annotate those object mentions with the relations the webpage asserts.

If an object has only one mention on the page, and participates in only one triple with the topic entity, we can simply label that mention with the relation in the triple. However, that is often not the case. A traditional approach to distant supervision would have us annotate all object mentions with all predicates they represented; however, the DOM setting again presents challenges and opportunities that motivate a more thoughtful process.

There are two potential problems. First, the object may participate in multiple relations with the topic, and all of these relationships may be represented on the page. If there is frequent overlap between two predicates (e.g., writers and directors of movies are often the same person), and they are mentioned in close proximity on the page, it will be hard to train a model to distinguish them.

Second, the object may have relationships with entities other than the topic. Even though the majority of the information on a detail page is about the topic entity, there are often relevant entities shown at the side or bottom of the pages as recommendations. Such relevant entities often share common objects with the topic entity; for example, in Figure 1.1, the *Comedy* genre applies to both the topic movie *Do the Right Thing*, and to a recommended movie *Crooklyn*. However, other genres that apply to *Crooklyn* may not apply to the topic movie and blindly extracting them may cause mistakes.

Since we emphasize precision over recall for annotation, we annotate *no more than one* mention of each object for a predicate. We try to harness local and global information to derive the correct annotation. In case the information for some predicates is duplicated on a page, we may miss labeling these true instances; however, this is acceptable since there is little benefit to extract the same triple twice. We compare our relation annotation approach with the traditional approach in our experiments (Section 3.5.4).

3.3.2.1 Local evidence for filtering

We first seek local evidence to resolve the ambiguity when an object may have multiple mentions on the webpage or may participate in multiple relations. Our intuition is that when multiple objects exist for a predicate, webpages are likely to put all these objects together on the page, such as in a list of names.

Given the DOM node for a $(pred, obj)$ pair, we find other objects for the same predicate. We identify their closest ancestor nodes that do not contain another mention of the same object, and choose the ancestor whose subtree contains the highest count of objects we find for the predicate. In case of a tie, we resort to global evidence, as we describe shortly.

Example 3.3.1 *In Figure 1.1, Spike Lee's name appears in both the director and writer sections of the page. It is not visible in the figure, but Lee also appears in the cast list of the film as well. Let us assume our KB contains triples representing all these relationships, as well as the film's relationship with several other actors. When we attempt to annotate Lee's "acted in" relationship, we will choose the mention in the cast list because it is in the same section of the page as other objects of the "acted in" predicate.*

3.3.2.2 Global evidence via clustering

Recall that in the topic identification step, we follow the *consistency* observation and rank XPaths by how frequently they contain candidate topics, and prefer those with higher count. The *consistency* observation applies to relations as well; however, since some multi-valued predicates are often represented in long lists and tend to occur in parts of the page with a higher variety of formatting than the page topic, the XPaths of relation candidates may be sparser than they are for topic candidates. For this reason, we instead cluster the XPaths of all potential object mentions of a predicate across pages, and prefer those XPaths that appear in larger clusters.

We use an agglomerative clustering approach, where in each iteration we find two nodes with the closest distance, and merge the clusters they belong to, until we reach the desired number of clusters. The distance function between two DOM nodes is defined as the Levenshtein distance [50] between their corresponding XPaths. The desired number of clusters is set to the maximum number of mentions of a single object in a webpage, such that all mentions of an object on a page can be placed into separate clusters.

This clustering step is only used in two cases: 1) the local evidence is insufficient to make an annotation decision; and 2) the same object appears as a value of a predicate in more than half of the annotated pages (recall the *informativeness* observation). It allows us to maintain high precision in our annotations, while still ensuring that we make annotations on the majority of pages.

Example 3.3.2 *In Figure 1.1, the genres for the film are mentioned twice, once correctly and once in a section describing a different movie in a recommendation; this occurs frequently on IMDb. Both sections contain two genres, so there is a tie from local evidence. After clustering all potential Genre mentions across all IMDb pages, we find that the mentions at the top of the page fall into a larger cluster, since all pages have the correct genres listed in a similar location. The mentions lower down in the page fall into a smaller cluster, since the genres of related films only sometimes have perfect overlap with the topic film. As a result, we annotate the mentions on the top of the page and discard those at the bottom.*

3.4 Training and Extraction

We treat DOM extraction as a probabilistic multi-class classification problem where the input is a DOM node and the output is the probability of the node indicating a particular relation; in other words, the classes consist of all predicates in our ontology, along with an "OTHER" class. A caveat is that the DOM node that contains the topic entity is considered as expressing the "name" relation for the topic. The challenges we face in training such an ML model mainly come from the automatically generated training data, which can be oftentimes noisy and incomplete.

3.4.1 Training Examples

Our annotation process produces positive training labels, but does not explicitly produce negative labels. For each webpage that has received annotations, we select randomly r unlabeled DOM nodes to use as negative examples (our "OTHER" class) for each positive example; in our experiments we used $r = 3$.

Since our annotations are incomplete and it is possible that some unlabeled nodes actually do express a relation, we thus shall not select the negative labels completely randomly. If we have labeled multiple positive examples of a predicate on a page, and they differ only in indices of their XPaths, we consider it likely that they belong to a list of values, such as the list of cast members for a film. When generating negative examples, we exclude other nodes that differ from these positives only at these indices, since they are likely to be part of the same list.

3.4.2 Training

Presumably we can train any ML model, such as logistic regression, random forest, and SVM. We experimented with several classifiers, but ultimately found the best results by modeling the probability of a relation given a node belonging to class k as a multinomial logistic regression problem:

$$Pr(Y = k \mid X) = \frac{e^{\beta_{k0} + \beta_k^T X}}{1 + \sum\limits_{i=1}^{M} e^{\beta_{i0} + \beta_i^T X}}$$

where X is a vector of features representing the node, β_k is the set of weights associated with relation class $k \in \{1, \ldots, M\}$, and β_{k0} is a class-specific intercept term. We harness the DOM structure to create two types of features to represent each node.

Structural features: We extract features relating to the attributes of neighboring nodes of the target node, as presented by Gulhane *et al.* in the Vertex project [36]. Specifically, we examine the node itself, ancestors of the node, and siblings of those ancestors (up to a width of 5 on either side) for the following HTML element attributes: `tag`, `class`, `ID`, `itemprop`, `itemtype`, and `property`. For each attribute of these nodes, we construct features as a 4-tuple consisting of *(attribute name, attribute value, number of levels of ancestry from the target node, sibling number).*

Node text features: In addition to the structural features, we generate features based on the texts of nearby nodes. We automatically compile a list of strings that appear frequently on the website and check if any of these strings appears nearby the node being classified; if so, a feature is created consisting of the word and the path through the tree to the node containing the string.

3.4.3 Extraction

In extraction, we apply the logistic regression model we learned to all DOM nodes on each page of the website. When we are able to identify the "name" node on a page, we consider the rest of the extractions from this webpage as objects and use the text in the topic node as the subject for those extracted triples. Our probabilistic extractor outputs a confidence value for each extraction; varying the confidence threshold required to make an extraction allows a trade-off between precision and recall.

3.5 Experimental Evaluation

Three experiments show the success of our techniques in several different verticals and settings. First, we show that our technique achieves state-of-the-art results on multiple verticals and is competitive with top systems trained on manually annotated data. Second, we show the advantages of our full annotation process over a baseline via experiments on IMDb, a complex website with many relationships. Third, to demonstrate the performance of our system on real-world data, we extracted over a million facts from dozens of long-tail websites in the Movie vertical.

3.5.1 Data sets

3.5.1.1 SWDE

To evaluate the performance of our approach across a range of verticals, we employed a subset of the Structured Web Data Extraction dataset (SWDE) [38] as our testbed. SWDE comprises ground truth annotations of 4–5 predicates for 8 verticals with 10 websites in each vertical and 200–2000

Table 3.1: Four verticals of the SWDE dataset used in evaluation.

Vertical	**#Sites**	**#Pages**	**Attribute**
Book	10	20,000	title, author, ISBN-13, publisher, publication_date
Movie	10	20,000	title, director, genre, rating
NBA Player	10	4,405	name, height, team, weight
University	10	16,705	name, phone, website, type

pages for each site. In our evaluation, we used four verticals, namely Movie, Book, NBA Player, and University, which all have named entities and have sufficient overlap between sites such that a KB constructed from one site should contain at least some of the same entities as the others. An overview of the dataset is provided in Table 3.1.

For the Movie vertical, we use a seed knowledge base derived from a download of the IMDb database that powers the IMDb website, with an ontology based on Freebase [11]. Table 3.2 gives a summary of this KB, which contains 85 million triples. For the rest of the three verticals, we arbitrarily chose the first website in alphabetical order from each vertical (`abebooks.com` for Books, `espn.com` for NBA Player, and `collegeboard.com` for University), and used its ground truth to construct the seed KB. We randomly selected half of the pages of each website to use for annotation and training and used the other half for evaluation; the confidence threshold was set at 0.5.

3.5.1.2 IMDb

To evaluate our approach on complex websites that provide information for many relations, we crawled IMDb[5] in May 2017, obtaining two sets of webpages: one consisting of 8,245 semi-structured pages about movies, and the other consisting of 1,600 semi-structured pages about people.

[5]Express written consent must be obtained prior to crawling IMDb; see `http://www.imdb.com/conditions`

Table 3.2: Common entity types and predicates in the KB used to distantly supervise experiments for the Movie vertical.

Entity Type	**#Instances**	**#Predicates**
Person	7.67M	15
Film	0.43M	19
TV Series	0.12M	9
TV Episode	1.09M	18

The seed KB is the same as used for the SWDE Movie vertical described in Section 3.5.1.1. A ground truth set was generated using the Vertex++ extractor described in Section 3.5.2, and manually spot-checked to ensure correctness. As with SWDE, half of the pages were used for annotation and training and the other half for evaluation, with a 0.5 confidence threshold.

3.5.1.3 CommonCrawl movie websites

The SWDE dataset validated our ability to obtain information from major websites well-aligned to our seed KB. However, to build a comprehensive KB, we need to obtain information also from smaller niche sites from all over the world. To evaluate our approach more broadly, we extracted knowledge from a range of movie websites in CommonCrawl[6].

The CommonCrawl corpus consists of monthly snapshots of pages from millions of websites [1] on the Web. We started with a few well-known sites, including *rottentomatoes.com*, *boxofficemojo.com*, and *themoviedb.org*. Based on a Wikipedia list of the largest global film industries by admissions, box office, and number of productions[7], we then issued Google searches for terms corresponding to these countries, such as "Nigerian film database", and recorded resulting sites that had detail pages related to movies. We also issued a few additional searches related to specific genres we thought may not be well-represented in mainstream sites, including "animated film

[6]`http://commoncrawl.org/`

[7]`https://en.wikipedia.org/wiki/Film_industry`

database" and "documentary film database". After compiling our list of sites, we then checked CommonCrawl[8] and kept all sites with more than one hundred pages available. Our final list contains a broad mix of movie sites, including sites based around national film industries, genres, film music, and screen size. Most are in English, but the set also includes sites in Czech, Danish, Icelandic, Italian, Indonesian, and Slovak.

In general, most websites contain a diversity in their pages reflected in their use of distinct templates for different entity types. For example, in a given site such as Rotten Tomatoes, movie pages often appear structurally similar to each other, whereas they differ significantly from celebrity pages. Recall from Section 3.2.1 that we apply the clustering algorithm described in [36] in an attempt to recover the inherent groups of templates. Our extraction approach is then run atop these individual clusters.

In our experience, we observed that the clustering algorithm does not always succeed in identifying the different page types. For example, out of the 73,410 Rotten Tomatoes pages we obtained, 71,440 were placed into a single cluster, including almost all of the semi-structured pages as well as unstructured pages. Note that this is a challenge for our process, since our method will have to learn an extractor capable of working on multiple very different page types, including unstructured pages.

We used the same knowledge base created for the SWDE Movie vertical as the seed KB. Since we do not have labels for CommonCrawl, we create the ground truth by sampling 100 extracted triples from each site and manually checking their correctness. A triple is considered to be correct if it expresses a fact asserted on the page from which it was extracted. We do not attempt to verify the real-world accuracy of the webpage's assertion, nor do we confirm which text fields on the page provided the extraction. Note however that the ground truth cannot tell us which triples we failed to extract from the website (*i.e.*, recall).

---

[8]For each site, we scanned the CommonCrawl indices for all monthly scrapes prior to January 2018 and downloaded all pages for the site from the scrape with the largest number of unique webpages. Note that these scrapes do not necessarily obtain all pages present on a site, so the retrieved pages represent only a subset of the full site.

3.5.2 Experimental setup

Implementations: We implemented our proposed technique, a distantly supervised extraction approach, together with three baselines.

1. VERTEX++: We implemented the Vertex wrapper learning algorithm [36], which uses manual annotations to learn extraction patterns, expressed by XPaths. We further improved the extraction quality by using a richer feature set. Training annotations were manually crafted by one of the co-authors to ensure correctness. Note that because of the access to precise training annotations, VERTEX++ presumably should obtain better results than distantly supervised approaches.

2. CERES-BASELINE: This baseline operates on the original Distant Supervision Assumption; that is, annotations are produced for all entity pairs on a page that are involved in a triple in the seed KB. The extraction model is the same as that described in Section 3.4.2. However, since there is no concept of a page topic in this setting, our annotation must identify a pair of subject-object nodes for a relation; to produce features for the pair, we concatenate the features for each node. Note that at extraction time, because we do not know the topic of a page, we need to examine all possible *pairs* of DOM nodes, which is computationally infeasible; thus, we identify potential entities on the page by string matching against the KB.

3. CERES-TOPIC: This method applies the algorithm described in Section 3.3.1 for topic identification, but then annotates all mentions of an object with all applicable relations, bypassing the relation annotation process described in Section 3.3.2.

4. CERES-FULL: This is the method proposed by this chapter, applying the algorithms from both Section 3.3.1 and 3.3.2 for annotation, and techniques in Section 3.4 for extraction.

We implemented our algorithms in Python. We used the logistic regression implementation provided by Scikit-learn using the LBFGS optimizer and L2 regularization with C=1 [76], and used Scikit-learn's agglomerative clustering implementation for the clustering step. We set parameters exactly as the examples given in the texts. All experiments were run on an AWS m4.2xlarge instance with 32 GB of RAM memory.

Evaluation: Our primary metrics for evaluation are precision, recall, and F1 score, the harmonic mean of precision and recall.

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn}$$

where tp is the number of true positive extractions, fp is the number of false positives, and fn is the number of false negatives.

3.5.3 Results on SWDE

Table 3.3 compares CERES-FULL with our baselines, and the state-of-the-art results on *SWDE* in the literature. Among those systems, Hao *et al.* [38], XTPATH [19], BIGGRAMS [64], and VERTEX++ use manual annotations for training; LODIE-IDEAL and LODIE-LOD [34] conduct automatic annotation (LODIE-IDEAL compares between all web sources in a vertical and LODIE-LOD compares with Wikipedia); RR+WADAR [71, 72] and WEIR [13] applied unsupervised learning.

As with these prior works, metrics in Table 3.3 follow the methodology of Hao *et al.* [38] in evaluating precision and recall on the SWDE dataset. Because their system can extract only one field per predicate per page, they base their metrics on *page hits* for each predicate, giving credit for a page if at least one text field containing an object for that predicate is extracted. For fair comparison, we restrict our system to making one prediction per predicate per page by selecting the highest-probability extraction. We also present full results showing precision and recall across all mentions in Table 3.4.

The CERES-FULL automatic annotation process annotated a large percentage of pages when there was good overlap with the KB, producing at least one annotation on 75% of pages for Movie, 97% for NBAPlayer, 74% for University, and 11% for Book. Most of the compared systems do not report number of annotations, but XTPath allows up to 10% of data to be used in training, while Vertex++ required two pages per site. CERES-FULL achieved the best results on two of the

Table 3.3: Comparing with state-of-the-art DOM extraction systems, CERES-FULL obtains highest F-measure on two verticals. Bold indicates the best performance. (F-measures of prior work are directly taken from the papers.)

System	**Manual Labels**	**Movie**	**NBA Player**	**University**	**Book**
Hao *et al.* [38]	yes	0.79	0.82	0.83	0.86
XTPath [19]	yes	0.94	**0.98**	0.98	**0.97**
BigGrams [64]	yes	0.74	0.90	0.79	0.78
LODIE-Ideal [34]	no	0.86	0.9	0.96	0.85
LODIE-LOD [34]	no	0.76	0.87 [a]	0.91 [a]	0.78
RR+WADaR [71]	no	0.73	0.80	0.79	0.70
RR+WADaR 2 [72]	no	0.75	0.91	0.79	0.71
WEIR [13]	no	0.93	0.89	0.97	0.91
Vertex++	yes	0.90	0.97	**1.00**	0.94
CERES-Baseline	no	NA [b]	0.78	0.72	0.27
CERES-Topic	no	**0.99** [a]	0.97	0.96	0.72
CERES-Full	no	**0.99** [a]	**0.98**	0.94	0.76

[a] F1 of distantly supervised systems is calculated based on predicates that were present in the ontology of the KB used for annotation. The KB for CERES-Topic and CERES-Full did not include Movie.MPAA-Rating because lacking seed data. LODIE-LOD did not include University.Type, NBAPlayer.Height, and NBAPlayer.Weight.

[b] Could not complete run due to out-of-memory issue.

four verticals, and overall obtained better results than most other system, even those using manual annotations.

CERES-BASELINE performed poorly across all verticals. In the movie vertical, where we

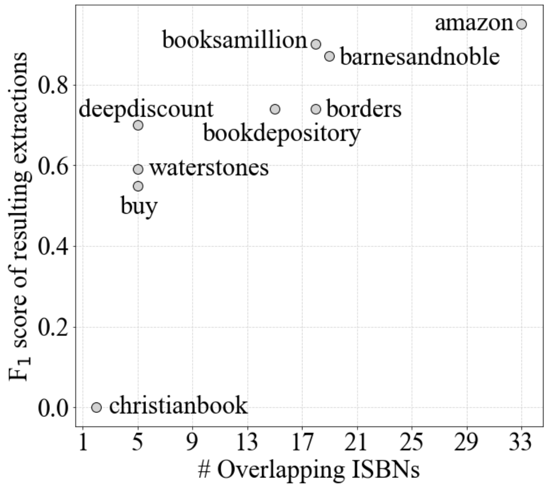

Figure 3.2: Extraction F1 vs. # of ISBNs overlapping w. the seed KB (potential number of annotated pages): lower overlap typically corresponds to lower recall. We omit acebooks.com site, which serves as the basis for the KB. An undefined F1 (when no extraction is produced) is shown as zero.

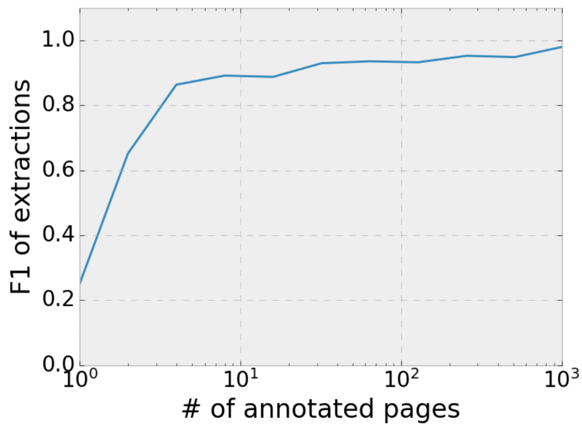

Figure 3.3: Extraction F1 improves with higher number of annotated pages used in the learning process for the SWDE Movie vertical. Note the log scale of the x-axis.

used a large seed KB constructed from IMDb, DB-BASELINE produced too many annotations and ran out of memory, even with 32 GB of RAM. CERES-TOPIC performed similarly to CERES-FULL. However, we note that many of the predicates in SWDE, such as University.Phone and Book.ISBN-13, have little ambiguity; as we will see soon in Section 3.5.4, CERES-TOPIC performs worse in a more complex setting.

Table 3.4: Comparing with the supervised extractor VERTEX++, our distantly supervised system CERES-FULL obtains comparable precision and recall across all extractions on the SWDE dataset. Bold instances indicate CERES-FULL beats VERTEX++.

Vertical	**Predicate**	**Vertex++**			**CERES-Full**		
		P	R	F1	P	R	F1
Movie	Title	1.00	1.00	1.00	1.00	1.00	1.00
	Director	0.99	0.99	0.99	0.99	0.99	0.99
	Genre	0.88	0.87	0.87	**0.93**	**0.97**	**0.95**
	MPAA Rating	1.00	1.00	1.00	NA	NA	NA
	Average	0.97	0.97	0.97	0.97	**0.99**	**0.98**
NBAPlayer	Name	0.99	0.99	0.99	**1.00**	**1.00**	**1.00**
	Team	1.00	1.00	1.00	0.91	1.00	0.95
	Weight	1.00	1.00	1.00	1.00	1.00	1.00
	Height	1.00	1.00	1.00	1.00	0.90	0.95
	Average	1.00	1.00	1.00	0.98	0.98	0.98
University	Name	1.00	1.00	1.00	1.00	1.00	1.00
	Type	1.00	1.00	1.00	0.72	0.80	0.76
	Phone	0.97	0.92	0.94	0.85	**0.95**	0.90
	Website	1.00	1.00	1.00	0.90	1.00	0.95
	Average	0.99	0.98	0.99	0.87	0.94	0.90
Book	Title	0.99	0.99	0.99	**1.00**	0.90	0.95
	Author	0.97	0.96	0.96	0.72	0.88	0.79
	Publisher	0.85	0.85	0.85	**0.97**	0.77	**0.86**
	Publication Date	0.90	0.90	0.90	**1.00**	0.40	0.57
	ISBN-13	0.94	0.94	0.94	**0.99**	0.19	0.32
	Average	0.93	0.93	0.93	**0.94**	0.63	0.70

Table 3.4 compares CERES-FULL with VERTEX++ in detail on full extractions. We observe that CERES-FULL obtains extraction quality comparable to VERTEX++ most of the cases, and sometimes even higher precision. In particular, on predicates such as Movie.Genre and University.Phone, which frequently have two or more values, we achieve recall over 0.95, even higher than the state-of-the-art annotation-based supervised extractor.

CERES-FULL performed best on the verticals where more annotations were produced, particularly the Movie and NBAPlayer verticals (note that we did not extract MPAA Rating because our KB does not contain any triple with this predicate). In contrast, we obtained our lowest F-measure on the Book vertical. This is because in the Book vertical there is very little overlap between the seed KB, built from the ground truth for `acebooks.com`, and the other websites. As shown in Figure 3.2, four of the sites had 5 or fewer pages representing books existing in our KB. However, even when we are able to annotate at most 5-20 webpages out of the 1000 pages, CERES-FULL still obtained high precision on these sites. Figure 3.3 shows that in the Movie vertical, a similar pattern emerges if a limitation is placed on the number of annotated pages that are used for learning the extractor model.

False positive extractions largely fell into two categories. First, the system sometimes extracted a node adjacent to the correct node, such as extracting the string "Author:" for the Book.Author predicate. The annotations were correct, but the features of the nodes were too similar for the classifier to distinguish. One could imagine applying similar global filtering according to the *Informativeness* observation on extractions. Second, we still make annotation errors: in one website in the University vertical, all pages of the site listed the two potential University.Type values ("public" and "private") in a search box on every page. This produced incorrect annotations and bad resulting extractions. Ignoring certain areas of the pages like search box is likely to alleviate the problem.

3.5.4 Results on IMDb

The IMDb dataset is a challenging testbed that demonstrates why our annotation scheme is critical to obtaining high extraction quality when extracting for complex domains. Recall that our KB is

Table 3.5: On IMDb, CERES-FULL obtains much higher extraction quality than CERES-TOPIC.

Domain	Predicate	CERES-Topic			CERES-Full		
		P	R	F1	P	R	F1
Person	name	1.0	1.0	**1.0**	1.0	1.0	**1.0**
	alias	0.06	1.0	0.11	0.98	1.0	**0.99**
	place of birth	0.96	0.87	0.91	1.0	0.93	**0.96**
	acted in	0.41	0.64	0.50	0.93	0.65	**0.77**
	director of	0.48	0.92	0.63	0.95	0.95	**0.95**
	writer of	0.32	0.56	0.41	0.89	0.69	**0.78**
	producer of	0.48	0.24	0.32	0.8	0.44	**0.57**
	All Extractions	0.36	0.65	0.46	0.93	0.68	0.79
Film/TV	title	1.0	1.0	**1.0**	1.0	1.0	**1.0**
	has cast member	0.93	0.46	0.62	1.0	0.49	**0.66**
	directed by	0.80	0.99	0.88	0.93	0.98	**0.95**
	written by	0.99	0.67	0.80	0.99	0.89	**0.94**
	release date	0.37	0.14	0.20	1.0	0.63	**0.77**
	release year	0.74	0.96	0.84	0.91	1.0	**0.95**
	genre	0.80	1.0	0.89	1.0	0.99	**0.99**
	(TV) episode number	1.0	1.0	**1.0**	1.0	1.0	**1.0**
	(TV episode) season number	0.98	1.0	**0.99**	0.87	1.0	0.93
	(TV episode) series	0.50	0.01	0.02	1.0	1.0	**1.0**
	All Extractions	0.88	0.59	0.70	0.99	0.65	0.78

constructed based on a download of IMDb data, so there is overlap between our KB and IMDb's website. Note, however, that the seed KB includes only a subset of facts found on the website (notably, it only contains links between people and movies if the person is a "principal" member of the film such as a lead actor or highly-billed writer or producer.) For example, for the pages in our web crawl, about 14% of has cast member facts on the webpages are present in the KB, along with 9% of producer of, 38% of director of and 58% of genre. This means that the KB is biased toward certain types of entities. This can complicate extraction in the case that this bias correlates with presentation on the webpage; however, this approximates the case of a seed KB containing only "popular" entities in a setting where we want to extract all entities, including those in the long tail. The relationship of the KB and dataset meant that we had strong keys connecting an IMDb page to a KB entity for most, but not all, of the pages, giving us ground truth to evaluate our topic identification step on this subset of pages.

One challenging aspect of IMDb is that many of the predicates, such as has cast member and director of, are multi-valued and consist of long lists of values (often 20 or more). Multi-valued predicates are a challenging and little-explored topic in semi-structured extraction, with most prior unsupervised and semi-supervised extractors restricting the problem to single-valued extractions [13, 38, 34, 37, 36]. In addition, IMDb contains many sections of the page that could easily produce false annotations (and thus extractions). For example, all People pages include a "Known For" section listing the person's four most famous films; however, this section does not correspond to a particular predicate, so any system that learns to extract it will produce erroneous extractions.

We compare the extraction quality of CERES-FULL with CERES-TOPIC in Table 3.5, and compare their annotation quality in Table 3.6. As shown in Table 3.7, we were very successful in identifying page topics. CERES-FULL annotation has decent precision (96% on Film/TV, 93% on Person), though sometimes lower recall (71% and 78%); this is because our annotation algorithms strive for high precision. Based on the slightly noisy annotations, we are able to train a robust extractor model, achieving an extraction F1 of 11% higher than CERES-Topic on Film/TV, and 72% higher on Person.

On this website, CERES-FULL obtains very high precision (99% on Film/TV pages, 93% on Person pages) and reasonable recall (65% and 68%). The fairly low recall is mainly dominated by false negatives for acted in and has cast member, which contribute a large number of triples (non-weighted recall among all predicates is 81% and 90% instead). Lower recall on these predicates, as well as producer of and writer of, is partially due to the fact that these are multi-valued predicates. For the acted in predicate, the recall loss is exacerbated by the fact that our seed

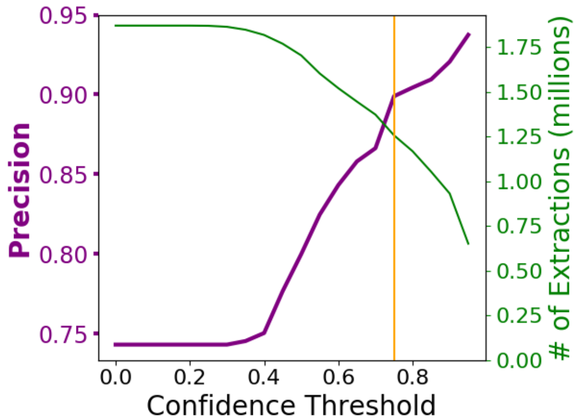

Figure 3.4: Extraction precision vs. number of extractions on the CommonCrawl dataset at various confidence thresholds; the orange bar indicates a 0.75 threshold yielding 1.25 million extractions at 90% precision.

KB only contains actors when associated IMDb character information is available, which is represented with a specific feature on the website. Lower precision on producer of and writer of is also due to the challenging nature of these predicates, whose objects are frequently only present in extraneous fields of the page such as a “Projects in Development” section, causing incorrect annotations. While our results are imperfect, the significant accuracy gain over CERES-TOPIC shows the advantages of our technique.

In contrast, CERES-TOPIC obtains much lower annotation precision than CERES-FULL, though the recall is slightly higher. The very noisy annotations lead to low quality extractors: for 7 predicates, the extraction precision is below 50%. In particular, CERES-TOPIC fails on Person pages, where there is significant ambiguity as films are mentioned in sections like “Known For” and “On Amazon Video” as well as the filmography.

3.5.5 Results on CommonCrawl movie sites

Figure 3.4 shows the precision and number of extractions from detail pages of all 33 movie websites at different thresholds for the extraction confidence; for example, a confidence threshold of 0.75 yields 1.25 million extractions at 90% precision. Table 3.8 shows a detailed breakdown of the 33 sites using an confidence threshold of 0.5; at this threshold, CERES-FULL extracted nearly 1.7

million triples off a basis of 440,000 annotations, obtaining a 4:1 extraction to annotation ratio and 83% precision. We highlight several successes by our extraction on CommonCrawl.

- Our movie websites are carefully selected to include long-tail movie websites. The only exceptions are *themoviedb.org* and *rottentomatoes.com*, which provide general film information, and where we obtain extraction precisions of 100% and 91%, respectively. For the rest of the websites, we obtain a precision over 80% on half of them. As shown in Table 3.9, for 6 out of the top-10 most extracted predicates, our overall precision is above 90%.
- CERES-FULL obtains greater than 90% precision on some of the foreign-language sites in Italian (*filmitalia.org*), Danish (*danksefilm.com*), and Czech (*kinobox.cz*).
- CERES-FULL was able to obtain high precision on sites when we can annotate only on a few tens of webpages, such as *kmdb.or.kr* and *moviecrow.com*.
- Precision increases monotonically with respect to confidence, allowing for a trade-off between precision and recall.
- CERES-FULL is able to extract triples that include entities not present in the seed KB. At 0.5 confidence, the ratio between annotated topic entities and extracted entities is 1:3.22, where most newly extracted entities are long tail entities.
- One notable instance of an inability to extract being a good thing occurred on boxofficemojo.com. Despite contributing over 74,000 pages to our dataset, the pages in the CommonCrawl scrape did not include any detail pages about movies, instead consisting almost entirely of daily box office charts. We consider the fact that our system did not produce erroneous extractions from this site to be a success.

Comparison with Knowledge Vault [25]: Knowledge Vault trained two DOM extractors to extract knowledge from the Web using Freebase as the seed KB, and the precision when applying a threshold of 0.7 is 0.63 and 0.64 respectively [26]. Because there are no broken-out details provided about the performance, we cannot conduct a side-by-side comparison. However, we believe

CERES obtained significantly higher precision because (1) we obtained an average precision of 0.94 on mainstream websites in SWDE, even with quite limited training data on some domains, and (2) on long-tail multi-lingual websites in CommonCrawl, which present significant challenges for extraction, we obtained an average precision of 0.83 (at a 0.5 threshold). In addition, unlike Knowledge Vault, we allow extracting facts where the subjects and objects are not present in the seed database. On the limited set of CommonCrawl websites, we already extracted knowledge for 155K new entities.

3.5.5.1 Discussion

We also observe low accuracy in extracting some of the long-tail websites. We next discuss where our algorithm may fall short and potential improvement directions.

Disjoint webpages or non-detail pages: Many websites contain pages generated by different templates, or even non-detail pages. Ideally, our clustering algorithm shall be able to separate them. Unfortunately, a strict implementation of Vertex clustering algorithm [36] sometimes does not obtain ideal results; for example, for `sodasandpopcorn.com`, so 36% of bad extractions occurred on these pages. This causes confusion at the extraction time. Indeed, we manually removed non-detail pages in our evaluations; including them would increase the extraction count by 200,000, but reduce the precision to 0.74. A robust clustering algorithm is critical to apply our algorithm to the whole web.

Semantic ambiguity: About a third of all mistakes happen in presence of semantic ambiguity for predicates. For example, `spicyonion.com` and `filmindonesia.or.id` list all films that a person is involved in, without distinguishing the role of the person like writer, director, or actor; `the-numbers.com` contains long lists of the date and box office receipts for every day the film was in theaters, instead of just release dates; `christianfilmdatab ase.com` and `laborfilms.com` contain a list of all genres on every page, rather than just genres for the topic movie. Ideally in such cases we shall make no annotation, but because an area of the webpage contains a superset of values for a multi-valued predicate, our method will wrongly annotate and learn to extract that area.

Template variety: Some websites, such as `colonialfilm.org.uk` and `bollywoodmdb.com`, may change the order of the predicates on the webpage, with the predicate indicated by an adjacent string. CERES-FULL may fail to learn the correct pattern when the text features are not strong enough; this category accounted for 23% of errors.

Over-represented or under-represented types and predicates: Most of the websites contain information solely about movies, but our KB contains over a million TV episodes (vs. 430K movies). Topic identification may wrongly match movies to episodes, such as in websites like `dianying.com`, `colonialfilm.org.uk`, `myanimelist`, and `samdb.co.za`, accounting for 5% of total error. On the other hand, for relations such as sound editor and camera operator, which do not exist in the seed KB, we do not necessarily make annotation mistakes but can make extraction mistakes when the XPaths are very similar. This happened on `kvikmyndavefurinn.is`, `jfdb.jp`, and `sfd.sfu.sk`.

We leave for future work to investigate how many of these aforementioned mistakes can be solved by applying knowledge fusion [25, 26] on the extraction results.

3.6 Related Work

Relation extraction has been an active area of research in the Natural Language Processing community for decades. A recent overview of this area can be found in a survey by Grishman [35]. Here we focus on techniques applicable to the semi-structured Web.

Wrapper induction: Early work for extracting semi-structured data was based on a supervised learning scheme called wrapper induction [48, 102, 36, 66]. The goal of wrapper induction is to learn wrappers (or rules) for relations of interest based on a sample of annotated data for a site. They work under the assumption that the observed HTML-formatted data is an instance of a database-backed template or schema, and the goal is to partially uncover these templates. Wrappers are often able to produce extractions with high precision and often high recall. However, they suffer from three problems: 1) they work only at the site-level and require human annotation effort for each site, 2) most wrappers were not designed to handle noisy annotations, so they tend to over-generalize in the presence of noise in the training data, and 3) they are often brittle — they tend to break with even minor changes in a site's structure.

Subsequent approaches aimed to alleviate some of these shortcomings. For example, in order to reduce annotation effort, Zhai *et al.* [102] used an active learning strategy to seek annotations from human editors when existing wrappers fail to extract. Dalvi *et al.* [23] proposed a generic framework to enable existing wrapper techniques to be resilient to noisy annotations. Their method searches the space of wrappers induced by subsets of (noisy) annotations and aims to identify one that ranks high on wrapper quality while also taking repeating patterns and schema size into account.

Moreover, most wrappers were designed to work at the site level (except for [97]) requiring manual labor to create annotations for each site to be processed. A recent approach by Hao *et al.* [38] promises to extract data for all sites in a vertical by only requiring that annotations be made for all pages of one site in that vertical. This approach entails accurate labeling for all pages of a site, for example by manually crafting regular expressions to extract attribute values of interest. Such demands of high-quality, site-wide labeling may not be easily fulfilled and thus can be a severe limitation. In general, although wrappers result in high quality extractions, they are not scalable given the annotation requirements.

Automatic extraction: Recent research activity has focused on devising unsupervised approaches for extraction, also known as *automatic extraction*. The underlying assumption is that the HTML pages are a result of a small number of database-backed templates, and thus mining patterns in the DOM tree structure of web pages, the content and contexts should help uncover the templates. Approaches in this category [3, 20, 101, 34] aim to either discover templates from a set of pages or find recurring patterns from a page to extract unseen relation instances from other pages. Although these approaches do not require any supervision, their major downside is that, typically, their extracted content requires an additional non-trivial post-processing step such as identification of attribute values, clustering, or ontology alignment.

Seed KB based extraction: Approaches in this category align attribute values in KB to text values on webpages to create annotations for learning wrappers. Gulhane *et al.* [37] exploit the redundancy in the attribute value content across websites in a vertical to make extractions from a new site in the vertical. They follow an iterative approach in which, at each iteration, some pages of a new site are annotated based on matching attribute values with the KB, and these annotated pages are used to learn wrappers for extracting from rest of the pages in the site. The LODIE project [34] is another example where annotations are generated based on matching attribute values on a page to dictionaries of attribute values pre-assembled from across the Linked Open Data cloud. Owing to their use of wrappers, both approaches require accurate annotations, which can be unrealistic to expect for complex sites commonly found on the web.

An alternate approach is that used by DIADEM [33], which, rather than assuming a KB, identifies extractable fields using a well-defined ontology and a set of recognizers corresponding to each entity type as well as predicate labels. Unlike CERES, this requires manual work to define these recognizers for each predicate and for each language.

Distant supervision based extraction: More recently, research methods (including this work) have employed the distant supervision paradigm [63] for leveraging information in a KB as a source of supervision for creating potentially noisy annotations. However, these methods are mainly designed for unstructured data. Knowledge Vault (KV) [25] is an exception however, which takes advantage of the DOM tree structure to predict relations for co-occurring entities on a page. KV requires the input pair of entities to exist in the KB, and hence cannot discover new entities. CERES, on the other hand, is able to discover new entities which is immensely useful for improving a KB's coverage.

Finally, for unstructured data, the strong distant supervision assumption has been relaxed to accommodate cases when a pair of entities may not hold for any relation in the KB [85], or may have multiple overlapping relations [40]. Most recently, *data programming* [84] and *heterogeneous supervision* [53] are proposed to unite diverse, possibly conflicting sources of supervision such as annotations by humans, supervision from a KB, or any labeling function. These annotation heuristics could be applied to our approach.

3.7 Conclusions

We have presented a new method for distantly supervised relation extraction from semi-structured websites. By using an entity-linking step in our annotation process to identify detail page topic entities, followed by a clustering process to identify the areas of pages corresponding to each predicate, we produce highly accurate annotations that can be used to train a supervised extractor. Experiments on the SWDE dataset demonstrate the state-of-the-art results on our system in multiple verticals, and a large-scale extraction project on hundreds of thousands of webpages shows the real-world usefulness of our approach.

While we believe this work offers an important step in the direction of truly automatic extraction from the web, additional work is necessary to scale the process to extract from the whole web across multiple domains. This will involve methods to effectively identify semi-structured pages and correctly group them into template-based clusters, expansion of the topic identification process to deal with non-named-entities, and methods to automatically identify the domain of a website.

Table 3.6: Accuracy of the automated annotations on 20 pages from IMDb Person and Film domains. Recall is measured as the fraction of facts from KB that were correctly annotated. Comparing with CERES-TOPIC, CERES-FULL has higher precision at the cost of slightly low recall.

Domain	**Predicate**	**CERES-Topic**			**CERES-Full**		
		P	R	F1	P	R	F1
Person	alias	0.19	1.00	0.33	1.00	0.71	0.83
	place of birth	0.84	0.55	0.67	0.90	0.45	0.60
	acted in	0.63	0.99	0.77	0.98	0.83	0.90
	director of	0.26	0.99	0.41	0.88	0.36	0.51
	writer of	0.33	0.99	0.5	0.77	0.81	0.79
	producer of	0.45	0.98	0.61	0.55	0.91	0.68
	All Annotations	0.46	0.99	0.60	0.93	0.78	0.83
Film/TV	has cast member	0.83	0.88	0.86	0.99	0.80	0.89
	directed by	0.47	0.74	0.58	0.88	0.71	0.79
	written by	0.68	0.52	0.59	0.90	0.36	0.51
	release date	0.53	0.59	0.56	1.0	0.56	0.72
	release year	0.27	0.75	0.39	1.0	0.71	0.83
	genre	0.55	0.82	0.66	0.96	0.82	0.88
	(TV) episode number	0.45	0.25	0.32	1.0	0.20	0.33
	(TV episode) season number	0.89	0.40	0.55	0.88	0.35	0.50
	(TV episode) series	0.44	0.42	0.43	1.0	0.42	0.59
	All Annotations	0.53	0.80	0.61	0.96	0.71	0.83

Table 3.7: Accuracy of topic identification on the IMDb dataset.

Domain	**P**	**R**	**F1**
Person	0.99	0.76	0.86
Film/TV	0.97	0.88	0.92

Table 3.8: CERES obtains an average of 83% precision on long-tail multi-lingual movie websites from CommonCrawl when using a 0.5 confidence threshold.

Website	Focus	# of Pages	# of Annotated Pages	# of Annotations	Total Extractions	Ratio of Extracted to Annotated Pages	Ratio of Extraction to Annotation	Precision
themoviedb.org	General film information	32,143	10,182	113,302	347,690	1.87	3.07	1.00
blaxploitation.com	Blaxploitation films	670	274	553	1,182	2.40	2.14	1.00
danksefilm.com	Danish films	2,100	403	1,712	13,146	2.98	7.68	0.98
archiviodelcinemaitaliano.it	Italian films	1,573	617	2,734	13,135	2.52	4.80	0.98
filmitalia.org	Italian films	2,847	909	4,247	10,074	1.96	2.37	0.96
kmdb.or.kr	Korean films	1,351	29	137	389	3.14	2.84	0.95
britflicks.com	British films	1,464	721	3,944	4,306	1.32	1.09	0.92
rottentomatoes.com	Film reviews	73,410	18,685	82,794	410,012	2.86	4.95	0.91
moviecrow.com	Indian films	569	84	271	912	2.70	3.37	0.91
nfb.ca	Canadian films	39,780	2,130	9,802	67,428	5.28	6.88	0.91
kinobox.cz	Czech films	37,988	2,940	16,820	60,337	5.58	3.59	0.90
samdb.co.za	South African films	1,424	10	25	281	6.60	11.24	0.88
dianying.com	Chinese films	15,789	1,333	3,998	48,302	8.22	12.08	0.84
giantscreencinema.com	IMAX films	370	50	333	856	3.00	2.57	0.83
myanimelist.net	Animated films	5,588	644	3,513	55,904	7.00	15.91	0.80
hkmdb.com	Hong Kong films	6,350	741	3,491	43,486	4.19	12.46	0.75
bollywoodmdb.com	Bollywood films	1,483	167	671	3,132	4.14	4.67	0.72
soundtrackcollector.com	Movie soundtracks	4,192	1,446	6,714	24,032	2.38	3.58	0.70
spicyonion.com	Indian films	5,898	752	2,375	7,439	2.89	3.13	0.70
shortfilmcentral.com	Short films	32,613	2,610	5,188	87,100	12.47	16.79	0.69
filmindonesia.or.id	Indonesian films	2,901	577	2,198	9,178	3.83	4.18	0.67
the-numbers.com	Financial performance	74,767	23,173	141,145	430,594	2.23	3.05	0.65
sodasandpopcorn.com	Nigerian films	3,401	190	423	3,862	13.16	9.13	0.62
christianfilmdatabase.com	Christian films	2,040	712	5,420	33,127	2.20	6.11	0.59
jfdb.jp	Japanese films	1,055	69	341	1,683	2.42	4.94	0.58
kvikmyndavefurinn.is	Icelandic films	235	49	185	868	3.96	4.69	0.57
laborfilms.com	Labor movement films	566	124	445	4,969	4.38	11.17	0.45
africa-archive.com	African films	1,300	300	892	2,150	2.31	2.41	0.42
colonialfilm.org.uk	Colonial-era films	1,911	48	212	1,605	20.00	7.57	0.29
sfd.sfu.sk	Slovak films	1,711	61	140	1,727	20.05	12.34	0.21
bcdb.com	Animated films	912	17	44	0	0.00	0.00	NA
bmxmdb.com	BMX films	924	1	1	0	0.00	0.00	NA
boxofficemojo.com	Financial performance	74,507	2	4	0	0.00	0.00	NA
Total	-	433,832	70,050	414,074	1,688,913	3.22	4.08	0.83

Table 3.9: Number of annotations, extractions, and precision for the 10 most extracted predicates on the CommonCrawl dataset at a 0.5 confidence threshold.

Predicate	**#Annotations**	**#Extractions**	**Precision**
film.hasCastMember.person	78,527	441,368	0.98
person.actedIn.film	86,273	379,848	0.96
film.hasGenre.genre	40,359	175,092	0.90
film.hasReleaseDate.date	25,213	132,891	0.41
film.wasDirectedBy.person	25,159	85,244	0.94
person.directorOf.film	14,893	67,408	0.72
person.createdMusicFor.film	7,065	61,351	0.25
person.hasAlias.name	4,654	59,051	0.99
film.wasWrittenBy.person	18,643	58,645	0.93
person.writerOf.film	8,665	36,871	0.52
All Predicates	414,194	1,688,913	0.83

Chapter 4

OPEN INFORMATION EXTRACTION WITH OPENCERES

4.1 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 [60]. Previous OpenIE work has concentrated on raw text, with the aim to extract "open" triples from natural language sentences [68], with another line of work focused on extracting from webtables [15].

Semi-structured websites (e.g. IMDb) contain many pages displaying information in standalone 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 4.1 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 [25]; the Ceres system showed that one can automatically extract from semi-structured sites with a precision over 90% using distant supervision [56]. These works, however, all build on the tradition of semi-structured ClosedIE techniques

[48, 90, 36, 33].

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 [38] 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 4.1 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 chapter 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 chapter makes three contributions. Our first contribution is to formally define a new problem of OpenIE from semi-structured websites (Section 4.2). We created a benchmark[1] for this problem by enhancing the SWDE corpus [38]; 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 4.4).

Our second contribution is OpenCeres, a solution for OpenIE on semi-structured websites (Sec-

[1] `https://github.com/cdlockard/expanded_swde`

tion 4.3). 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 for distant supervision and label propagation.

Our final contribution is a comprehensive evaluation on our new benchmark dataset and online websites (Section 4.5). 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 4.7, and serve as a good baseline for future work.

4.2 Overview

4.2.1 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 4.2.1 (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*

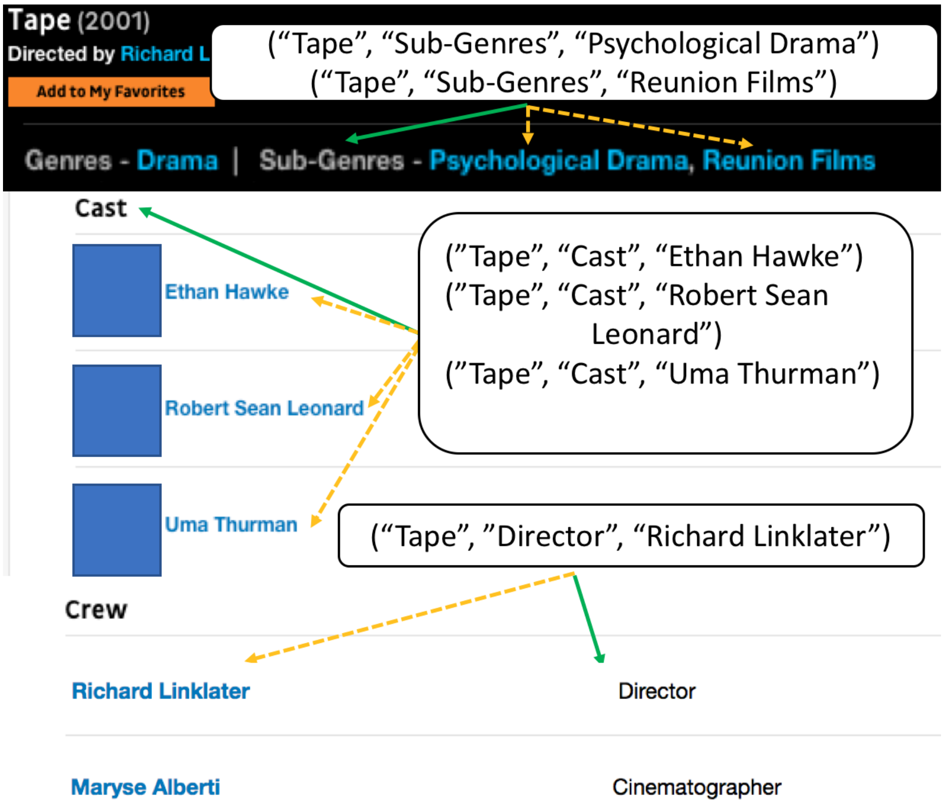

Figure 4.1: A cropped portion of the detail page from allmovie.com for the film *Tape* with some triples indicated. Solid green and dashed yellow arrows indicate predicate strings and objects respectively.

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) [32], 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.

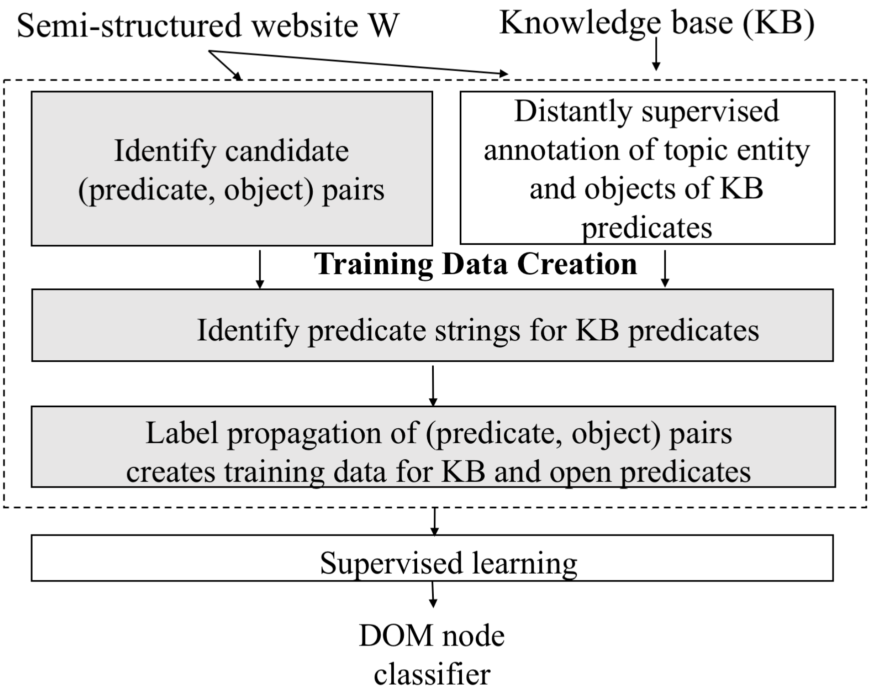

Figure 4.2: An overview of our proposed semi-structured OpenIE model learning process. Shaded areas indicate contributions of our process.

4.2.2 From ClosedIE to OpenIE

We first summarize the Ceres techniques proposed in [56], 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 4.2 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 [56]. 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.

1. Predicate-object candidate generation: We first generate potential candidate *(predicate, object)* pairs, as described in Section 4.3.1. 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 4.3.2. For example, for the page in Figure 4.1, 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 4.3.3; these new *(predicate, object)* pairs give us training labels for new predicates. For example, in Figure 4.1, 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.

4.3 Approach

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

4.3.1 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.

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

For example, consider the object strings in the Cast section of Figure 4.1, 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.

4.3.2 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 [10]) 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.

4.3.3 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 similar 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:

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

where i and j are candidate pairs, r_d calculates the DOM path, r_h calculates the horizontal distance

[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.

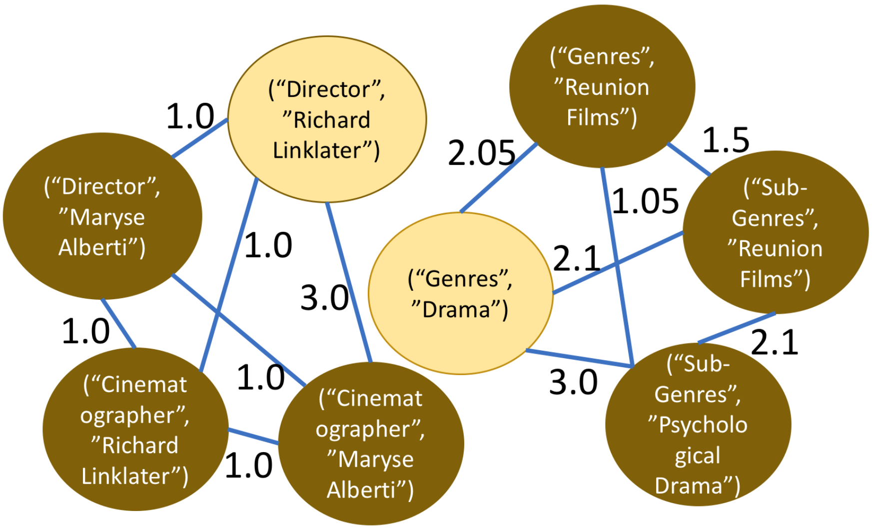

Figure 4.3: A portion of the graph corresponding to the webpage in Figure 4.1. Lighter nodes indicate seed pairs labeled by the Ceres process.

between candidate predicate and candidate object, and r_v calculates the vertical distance.

A sample graph for the webpage in Figure 4.1 is shown in Figure 4.3. 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.

To speed up propagation, we keep only the 10 top-weighted edges for each pair. On average, on the dataset in Section 4.4, 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 [52], 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 [56] extraction model.

4.4 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 [38]. However, the 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 [36]), 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 4.1. 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 4.1 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.

4.5 Experimental Evaluation

4.5.1 Experimental Setup

Datasets: Our primary dataset is the augmented SWDE corpus described in Section 4.4. In addition, we used the set of 31[5] movie websites (comprising 433,000 webpages) found in CommonCrawl[6] from Ceres. To generate seed KBs for the distant supervision, we relied on the methodology from Ceres, 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.

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.* [13], 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 4.3.1, identifies those where the predicate field ends with a colon, and

[5] We removed the two sites on which we reported Ceres made no annotations [56].

[6] www.commoncrawl.org

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 4.3, using the generated training data to train a Ceres extractor.

In addition, to understand the upper 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 4.3.2) 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 4.3.3. 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 4.4) 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.

4.5.2 Results on SWDE

Overall results: Table 4.2 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 4.3. 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 4.4 shows how the label propagation process successfully creates new training examples from a small number of seeds. While propagation does introduce some precision 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 4.4 shows that Candidate Pair Identification has increasing recall in capturing true candidate pairs in the SWDE-Movie vertical with more neighbors considered, with a trade-off in increased runtime due to the creation of more pairs; we used $k = 5$ in our experiments.

4.5.3 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

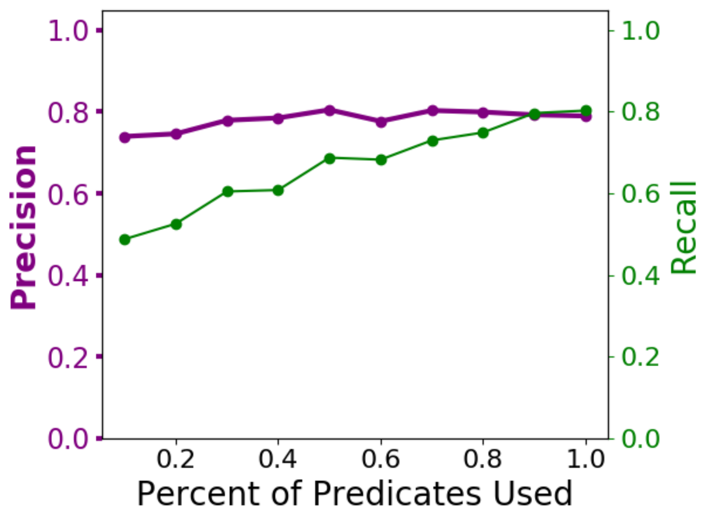

Figure 4.4: Precision and recall of the training data automatically created for the Movie domain by OpenCeres-GoldProp, after label propagation from seed data created by first sampling varying percents of the ground truth predicates for a site and then sampling a constant 25% of ground truth objects for each predicate.

Figure 4.2. 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 4.5 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.

4.6 Related Work

In unstructured text, OpenIE was originally proposed by Banko *et al.* [8], an approach extended by ReVerb [30] and Ollie [61], which relied on syntactic constraints to identify relation patterns. Our

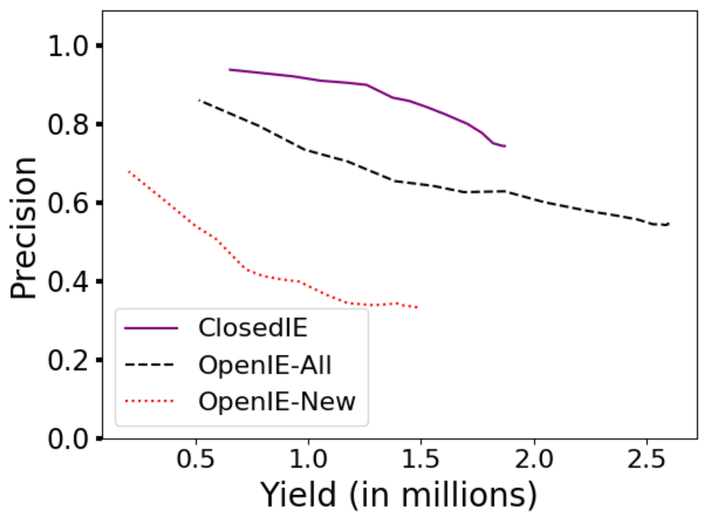

Figure 4.5: Extraction precision vs. number of extractions on the CommonCrawl dataset at various confidence thresholds; ClosedIE is the implementation of the Ceres system. OpenCeres-All shows all extractions from OpenCeres, while OpenCeres-New shows only the OpenCeres extractions with new predicates.

approach is influenced by Wu *et al.* [98], who aligned Wikipedia infobox contents to article text to automatically create training data for an extractor. Recent work on neural extraction models [21] has explored entirely supervised models learned from a modified version of the QA-SRL dataset [91].

The line of research that has most closely examined the prospect of OpenIE-style extractions using webpage structure is the work on Webtables [15, 22, 7, 14]. 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 [74, 46].

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 [48]. Work in this line of research has achieved high accuracy from only a few labeled examples, but requires manually-annotated examples for each website [36]. To remove this bottleneck, researchers have explored alternative ways to automatically create labeled data and learn models from such potentially noisy labels [23, 34, 33, 56]. However, these approaches cannot find triples for predicates that are not in the seed ontology.

The Roadrunner project [20] does attempt to identify the objects of all relations represented on a site, but does not extract predicate strings. However, the WEIR project [13] 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.

4.7 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.

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 4.1: Statistics of the augmented SWDE dataset.

System	Movie		NBA		University	
	P	R	P	R	P	R
WEIR [13]	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 4.2: Extraction precision and recall (lenient) on SWDE domains. OpenCeres on average improves over baseline by 36% on precision and by 88% on recall.

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

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

Table 4.4: 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.

Chapter 5

ZERO-SHOT INFORMATION EXTRACTION WITH ZEROSHOTCERES

5.1 Introduction

Semi-structured websites offer rich sources of high-quality data across many areas of knowledge [25]. These websites present information via text that is accompanied by rich visual and layout features that can be generalized beyond a single website. However, most prior work on information extraction (IE) from websites has largely ignored most of these features, instead relying only on HTML features specific to an individual website [31]. This requires training data for every website targeted for extraction, an approach that cannot scale up if training data must be manually created.

To circumvent manual data annotation, previous work used a distant supervision process requiring a knowledge base aligned to the website targeted for extraction [34, 56], including for OpenIE extraction [8, 13, 57]. These methods, however, can only learn a website-specific model based on seed knowledge for the site, but cannot be generalized to the majority of websites with knowledge from new verticals, by long-tail specialists, and in different languages.

In this chapter, we introduce the task of *zero-shot relation extraction* from semi-structured websites, in which a learned model is applied to extract from a website that was not represented in its training data (Figure 5.1). Moreover, we introduce ZEROSHOTCERES, a graph neural network model that encodes semantic textual and visual patterns common across different training websites and can generalize to extract information from documents with never-before-seen templates and topics.

Unlike unstructured text, which can be modeled as a sequence, or images, which can be modeled as a two-dimensional grid of pixels, it is not obvious how to operate over the many shapes and sizes of text fields on a semi-structured webpage. We illustrate our intuition using the webpage snippets in Figure 5.1: Despite their differences, each site uses alignment of relation and object

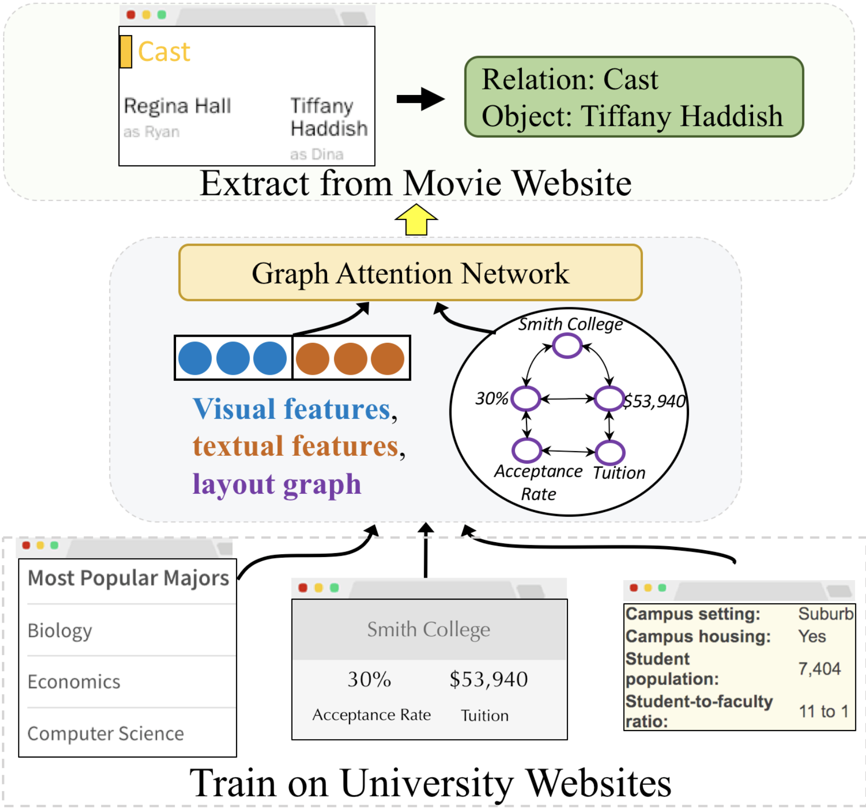

Figure 5.1: The ZeroShotCeres zero-shot open-domain information extraction process learns generalizable graph-based representations of how relations are visually presented on semi-structured websites, allowing for training on one vertical (such University sites) and extraction from another (such as Movie sites).

strings, either vertically or horizontally, to help indicate relationships; in addition, relation strings are often more prominent than their objects, either in size or boldness. Such features are semantically meaningful to readers and often consistent from site to site; thus, encoding them into the representation of webpages will allow us to generalize to unseen sites.

Our model, ZEROSHOTCERES, encodes these diverse feature types in a graph representation in which each text field becomes a node in a graph, connected by edges indicating layout relationships on the page. This abstracts away the details of the page while maintaining the core visual structure presented to the reader. A graph neural network is then applied to produce a new representation of each text field, informed by the surrounding page context. This representation is then used to extract entities and relationships from the document. This allows us to extract not only in the closed-domain setting, but also allows us to conduct OpenIE on websites about entirely new subject verticals not seen during training.

Our contributions are threefold: (a) We introduce a graph neural network model for webpage representation that integrates multi-modal information including visual, layout, and textual features, enabling generalization for IE from never-before-seen websites. (b) We propose the first approach to enable Open Information Extraction from semi-structured websites without prior knowledge or training data in the subject vertical. (c) Our method works in both OpenIE and ClosedIE settings. We conduct evaluations showing the effectiveness of the technique and exploring the challenges of zero-shot semi-structured IE, achieving a 31% improvement in F1 compared to an OpenIE baseline. The graph model gives a 26% F1 boost when extracting according to a defined schema (ClosedIE).

5.2 Related Work

DOM-based ClosedIE: The conventional approach to extraction from semi-structured websites is wrapper induction [48], in which training data for documents from a given template is used to learn a rule-based extractor based on DOM (i.e., HTML) features to apply to other documents of the same template, extracting relations according to a pre-defined ontology ("ClosedIE"). Since this approach requires training data for each template targeted for extraction, recent work has focused on reducing the manual work needed per site. Fonduer [99] provides an interface for easily creating training data, Vertex [36] uses semi-supervision to minimize the number of labels needed, LODIE [34] and Ceres [56] automatically generate training data based on distant supervision, and DIADEM [33] identifies matching rules for specific entity types.

DOM-based OpenIE: WEIR [13] and OpenCeres [57] offer OpenIE approaches to DOM extraction. The latter method uses visual features in a semi-supervised learning setting to identify candidate pairs that are visually similar to known *(relation, object)* pairs; however, the ultimate extraction model learned is still site-specific and based on DOM features rather than the more generalizable visual or textual features. Pasupat *et al.* [73] present a zero-shot method for ex-

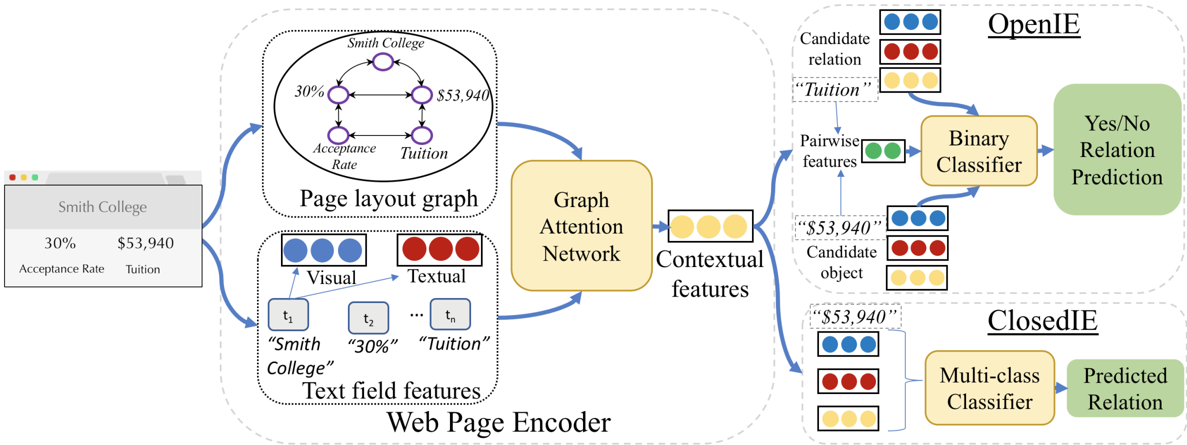

Figure 5.2: A depiction of the web page representation module (left) and relation classifiers (right).

traction from semi-structured webpages, but limit their work to extraction of entities rather than relationships and do not consider visual elements of the page.

Multi-modal extraction: The incorporation of visual information into IE was proposed by Aumann *et al.* [5], who attempted to learn a fitness function to calculate the visual similarity of a document to one in its training set to extract elements like headlines and authors. Other recent approaches that attempt to address the layout structure of documents are CharGrid [42], which represents a document as a two-dimensional grid of characters, RiSER, an extraction technique targeted at templated emails [45], and that by Liu *et al.* [54], which presents an RNN method for learning DOM-tree rules. However, none of these address the OpenIE setting, which requires understanding the relationship between different text fields on the page.

The approaches most similar to ours are GraphIE [81] and the approach by Liu *et al.* [55]. Both approaches involve constructing a graph of text fields with edges representing horizontal and vertical adjacency, followed by an application of a GCN. However, neither approach makes use of visual features beyond text field adjacency nor DOM features, and both only consider extraction from a single text field rather than OpenIE. In addition, they show only very limited results on the ability of their model to generalize beyond the templates present in the training set.

5.3 Problem and Approach Overview

5.3.1 Zero-shot relation extraction from semi-structured websites

We address the problem of extracting entities and the relationships between them as expressed by never-before-seen semi-structured websites. A *semi-structured website* typically belongs to a subject *vertical* V, where V is a general field of knowledge such as movies, finance, or sports. A semi-structured website consists of a set of *detail pages* sharing a similar template, each of which contains a set of facts about a page topic entity e_{topic}. The HTML document w defines a set of text fields T, which the web browser renders as a *webpage* according to the instructions defined in the HTML and any referenced auxiliary files such as CSS or JavaScript. The text fields have both textual and visual features, described in Section 5.4.2.1.

5.3.1.1 Relation Extraction

Our goal is to extract *(subject, relation, object)* knowledge triples, where the subject is e_{topic}, the object is a text field $t \in T$ containing the name of an entity (or atomic attribute value), and the relation indicates the relationship between the two entities.

For this work, we assume the page topic entity has already been identified, (such as by the method proposed by Ceres or by using the HTML `title` tag) and thus limit ourselves to identifying the objects and corresponding relations. We consider the following two settings:

Relation Extraction (ClosedIE): Let R define a closed set of relation types, including a special type indicating "No Relation". Relation Extraction is the assignment of each text field t to one $r_i \in R$, which indicates the relationship between the entity e_{object} mentioned in t and e_{topic}.

Open Relation Extraction (OpenIE): Given a pair of text fields (i, j), Open Relation Extraction is a binary prediction of whether i is a relation string indicating a relationship between the entity e_{object} mentioned in j and e_{topic}.

5.3.1.2 Zero-shot Extraction

Unlike prior work that requires the learning of a model specific to the semi-structured website targeted for extraction, we look at zero-shot extraction. Given a semi-structured website W targeted for extraction, zero-shot extraction is the learning of a model without any use of pages from W during training. We consider two zero-shot settings:

Unseen-Website Zero-shot Extraction is the learning of a model without any use of pages from W, but with pages from some other website(s) from vertical V during training.

Unseen-Vertical Zero-shot Extraction is the learning of a model without any use of pages from W or of pages from any website with vertical V during training.

5.3.2 Approach Overview

Figure 5.2 depicts our approach for zero-shot relation extraction (detailed in Section 5.5) leveraging a web page representation that will capture the similarities in visual and textual semantics across websites (Section 5.4). Our web page representation module first converts each page into a layout graph (Section 5.4.1) that abstracts away the details of the page structure while maintaining the adjacency relationships between text fields. We represent each text field with an initial feature vector of visual and textual attributes. This input is passed into a graph neural network that allows for information to flow between nodes, producing a new text field representation that captures contextual information (Section 5.4.2).

To obtain a web page encoding, we leverage a pre-training step with auxiliary loss function $\mathcal{L}_{pre}$ that encourages the model to produce an intermediary representation useful for IE. This is performed via a three-way classification that determines if a text field contains a relation name, the object of some relation, or irrelevant text (Section 5.4.3). After pre-training, the weights of this GNN are frozen and it can be applied to new pages, with its output used as input into a relation extraction module, optimized with task-specific loss function $\mathcal{L}_{task}$, where the task is either OpenIE or ClosedIE, described in Section 5.5. The resulting approach minimizes our overall loss $\mathcal{L}_{\text{ZSC}\textsc{eres}}$, with:

$$\mathcal{L}_{\text{ZSC}\textsc{eres}} = \mathcal{L}_{pre} + \mathcal{L}_{task} \tag{5.1}$$

5.4 Web Page Encoder

The key idea behind our solution is to train webpage representations to capture the fundamental similarities in visual and textual semantics across websites to express relations, objects, and their relationships. The fundamental characteristics we capture, generalizable across templates and verticals, thus allow us to carry over our knowledge across websites and enable zero-shot extraction.

There are two key parts in our solution. First, we build a graph to capture the layout relationships in a more abstract form that allows us to more easily learn the common features across different sites such as the fact that relation strings are often to the left or above their objects (Section 5.4.1). Second, we apply a Graph Neural Network (GNN) to learn representations for each node capturing contextual information about its neighborhood on the webpage (Section 5.4.2), allowing information to flow through the nodes, providing context (e.g., flowing through "Cast" to a far-away node "Uma Thurman" via the closer node "Ethan Hawke" in Figure 5.3). This representation will be useful for relation extraction as described in Section 5.5.

5.4.1 Page graph construction

We encode the layout relationships between text fields in the form of a graph, G, consisting of a set of nodes N, each corresponding to a text field, and a set of edges E corresponding to relationships between the text fields. The edges capture three forms of adjacency, as shown in the example in Figure 5.3:

Horizontal: Edges are added when two text fields are horizontal neighbors on the page; that is, they have a shared vertical location and there are no other text fields between them.

Vertical: Edges are added when two text fields are vertical neighbors on the page; that is, they have an overlapping horizontal location and there are no other text fields between them.

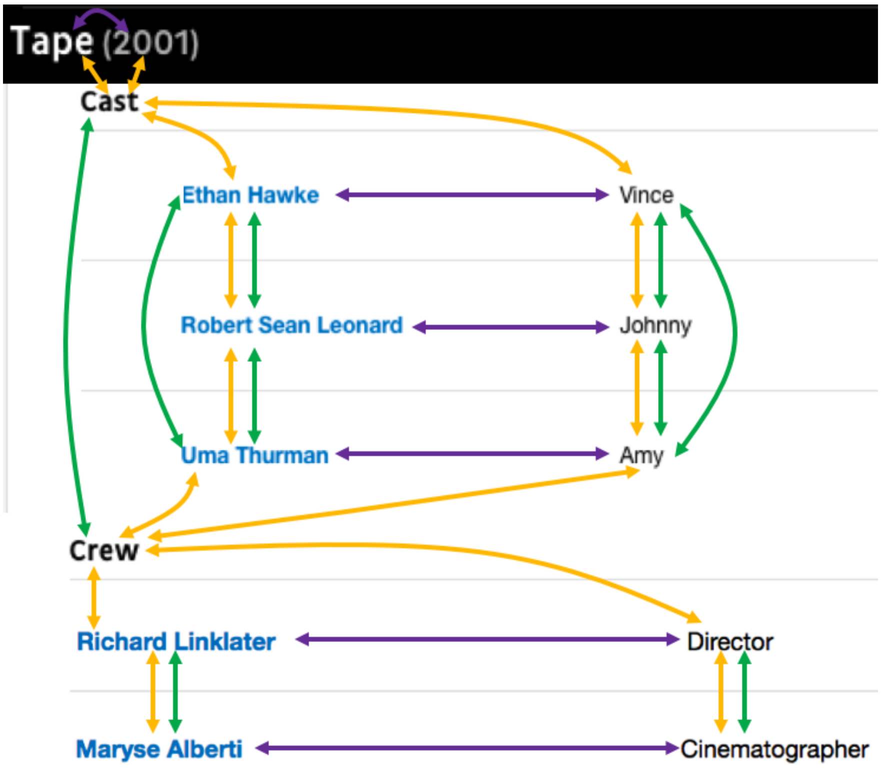

Figure 5.3: A cropped portion of the detail page from allmovie.com for the film *Tape*. Arrows overlaid showing the constructed page graph consisting of edges for each horizontal (purple), vertical (yellow) and DOM (green) relationship between text fields.

DOM: Edges are added when two text fields are siblings or cousins in the DOM tree; that is, the absolute XPaths identifying their locations differ only at a single index value.

5.4.2 Graph Neural Network (GNN)

To build a representation of each text field that incorporates the surrounding page context, we use Graph Attention Networks (GAT) [94]. The feature vector for each text field (described below) and the page graph form the input to a GAT, which then produces a new representation for each text field based on the surrounding context in the graph. Specifically, for each text field i, GAT layer l computes a representation h_i^l as follows:

$$h_i^l = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W_G^l h_j^{l-1}\right), \tag{5.2}$$

where N_i is the set of neighbors of node i in the graph, and h_j^{l-1} is the representation of node j from the preceding layer; h_j^0 indicates the input features for the node. (For each node, we add a self loop to the graph; that is, including i in N_i.) W_G^l is a learned weight matrix applied to the node features for layer $l-1$ and σ is a non-linear function, in our case a ReLU. The attention weight α_{ij} determines how influenced a node's representation is by each of its neighbors, calculated as follows:

$$\alpha_{ij} = \frac{\exp\left(\sigma\left(a^\top [W_G^l h_i^{l-1}; W_G^l h_j^{l-1}]\right)\right)}{\sum_{k \in N_i} \exp\left(\sigma\left(a^\top [W_G^l h_i^{l-1}; W_g^l h_k^{l-1}]\right)\right)}, \tag{5.3}$$

where a is a weight vector applied against the concatenation (represented by ";") of the two node's features as transformed by W_G^l and σ is a ReLU. This produces a new contextualized set of features for each node that are informed by the surrounding page context. We describe the original input features for each text field in the next section.

5.4.2.1 *Initial text field features*

For each text field on the page, we produce an initial feature vector containing both visual feature vector V and textual feature vector T. We define the input feature vector h_i^0 for text field i as:

$$h_i^0 = [T(i); V(i)] \tag{5.4}$$

where ";" represents concatenation.

Visual Features: A numeric feature vector is constructed representing the bounding box coordinates of the text field, the height and width of the bounding box, and the font size, along with one-hot features representing the typeface, font weight, font style, color, and text alignment.

Textual Features: In ClosedIE, to capture its semantics, the textual content of the text field is processed with a pre-trained BERT [24] model. To produce a representation of the entire text field, we simply average the BERT-Base output for each token in the text field. For OpenIE, since the goal is to generalize to entirely new subject verticals that may contain text not seen during training, only a single textual feature is used[1]: the percent of pages on the site on which the string in the text field appears. This frequency measure helps differentiate relation strings, which are likely to be common, from object strings, which are more likely to be rare.

5.4.3 Pre-Training Web Page Encoder

To encourage the GNN weights to capture the features necessary to represent relationships on the page, we use a pre-training step to learn the GNN representation before incorporating it into the extraction model. The pre-training task is a simplified form of the OpenIE task. To speed up training by avoiding the pairwise decisions necessary for OpenIE, we instead perform a multi-class classification of each text field into a class c in the set {Relation, Object, Other}:

$$p\left(c|h_i^l;\theta\right) = softmax\left(W_{pre}h_i^l\right) \tag{5.5}$$

where h_i^l is the output of the GNN for the text field, W_{pre} is a weight matrix, and θ comprises W_G and W_{pre}. Given a training set with T text fields, each with a ground truth class y_i^{pre}, we minimize the cross-entropy loss $\mathcal{L}_{pre}$:

$$\mathcal{L}_{pre} = -\sum_{i=1}^{T} \log p\left(y_i^{pre}|h_i^l,\theta\right) \tag{5.6}$$

To discourage overfitting to spurious details in the small number of websites in our training set, we freeze the GNN weights after pre-training and do not update them during the full OpenIE training. After pre-training we discard the linear layer W_{pre} since it is not needed for subsequent steps; instead, we directly use the GNN output h^l.

[1]This feature is also used during ClosedIE

5.5 Relation Extraction Model

Once we have the new representation h_t^l of each text field t produced by the above GNN process, we can perform our final classification.

5.5.1 OpenIE

For OpenIE, the classification decision must be made over a pair of text fields, i and j, the first containing the candidate relation string and the second containing the candidate object string. To avoid examining all possible pairs of fields, we first apply the candidate pair identification algorithm from OpenCeres, which filters down to a set of potential pairs based on physical and layout distance between text fields.

For each candidate pair, we concatenate the GNN-produced contextual features h^l for both text fields with the original features h^0 for both text fields (since some information can be diluted in the GNN), as well as a pairwise feature vector that simply contains the horizontal and vertical distance between the two text fields, and pass them into a binary classifier:

$$r_i^{OIE} = \text{FNN}\left([h_i^0; h_j^0; h_i^l; h_j^l; pairwise_{i,j}], \theta^{OIE}\right) \tag{5.7}$$

where FNN is a feed-forward neural network with parameters θ^{OIE}, ";" indicates concatenation, and r_i^{OIE} is the predicted probability that the two text fields constitute a *(relation, object)* pair. We then optimize for cross-entropy loss across training examples T with $y_i^{OIE} = 1$ if the pair is positive:

$$\mathcal{L}_{OIE} = \sum_{i=1}^{T} y_i^{OIE} \log r_i^{OIE} + \left(1 - y_i^{OIE}\right) \log\left(1 - r_i^{OIE}\right) \tag{5.8}$$

5.5.2 ClosedIE

For ClosedIE, we perform a multi-class classification using the contextual representation produced by the GNN (h_i^l) along with the original features (h_i^0) for text field i:

$$r_i^{CIE} = \text{FNN}\left([h_i^0; h_i^l], \theta^{CIE}\right) \tag{5.9}$$

where FNN is a feed-forward neural network parameterized by θ^{CIE}, ";" indicates concatenation, and r_i^{CIE} is the predicted probability of relation r in set R. We optimize for cross entropy loss $\mathcal{L}_{CIE}$:

$$\mathcal{L}_{CIE} = -\sum_{i=1}^{T} \log p\left(y_i^{CIE} | h_i^0, h_i^l, \theta^{CIE}\right) \tag{5.10}$$

where y_i^{CIE} is the true class for example i. For both ClosedIE and OpenIE we use one hidden layer in the feed-forward network.

5.6 Experimental Setup

5.6.1 Dataset

For both OpenIE and ClosedIE, our primary dataset is the extended version [57] of the **SWDE** dataset [38], which contains gold labels for OpenIE extractions for 21 English-language websites (each with one template) in three subject verticals (Movie, NBA, and University), with between 400 and 2,000 pages per site. We generated ClosedIE labels by converting the OpenIE labels to ClosedIE labels via manual alignment of OpenIE relations between websites, giving a set of 18 relations for the Movie vertical, 14 for NBA, and 13 for University, described below.

We used three SWDE Movie sites (AMCTV, AllMovie, and IMDb) as a development set and did not evaluate on them for the reported results.

5.6.1.1 ClosedIE Label Mappings

SWDE provides OpenIE labels for all binary relations between the objects mentioned on the page and the page topic entity. These labels include the relation string used to indicate the relationship, sometimes including multiple acceptable surface forms if there is more than one applicable string for the relation (usually due to more or less specific versions of the relation). The original SWDE data only includes ClosedIE labels for a small subset of relation types. To create ClosedIE ground truth for all relations on the sites, we examined all OpenIE relations across the SWDE sites and grouped them into a set of relations that each represented the same fundamental idea. In some cases, we chose to map relations into a somewhat more general category, such as mapping "Associate Producer" and "Executive Producer" into the same "Producer" concept. After obtaining this set, we eliminated all relations that appeared on fewer than 3 websites in the dataset. The set of relations used for the ClosedIE experiments is given in Table 5.1. The full mapping of OpenIE to ClosedIE relations can be found at `https://github.com/cdlockard/expanded_swde`.

5.6.2 Training Data Creation

The Extended SWDE dataset provides ground truth extractions of OpenIE predicate and object strings for the webpages it contains. However, it does not specify which text fields on the page were the source of the extractions. To create training data, we need to label a specific text field. It is usually the case that each ground truth string matches only one text field, so there is no ambiguity, but in cases where multiple text fields have the same value, we must disambiguate which one to use. We did this by identifying all matching text fields for the ground truth predicate and object and chose the pair in which the predicate and object strings have the closest Euclidean distance on the rendered page.

While this is generally a safe assumption, there are still occasional errors in the training data. In particular, we observed that the NBA vertical had considerably more ambiguous cases since most relations are numerical and the pages often contained large tables of numbers. We hypothesize that this may explain why performance on the NBA vertical is lower when using Unseen-Website training data compared to the Unseen-Vertical setting (Table 5.2).

During testing, we applied the same standard used by prior work on the dataset and accepted an answer as correct if it matched the ground truth string, regardless of which text field produced the extraction.

5.6.3 Experimental Settings

For each model tested (both our own and the baselines), we classify the training setting into the following categories indicating the level of vertical or site-specific knowledge used, in decreasing level of difficulty.

- **Level I–Unseen-Vertical Zero-shot (OpenIE only):** A model is trained on sites from two of the three verticals (e.g. NBA and University) and applied to sites from the other vertical (Movie). This is the hardest case and is important when we wish to extract knowledge from new verticals where we do not have any prior knowledge or annotations.

- **Level II–Zero-shot with Vertical Knowledge:** A model is trained on all sites but one (spanning Movie, NBA, and University) and then applied to the held-out site. As in cross-validation, experiments are repeated with each site having a turn being held out. It is easier than Level I but is still important for a new website that may not have data overlapping with other websites in the same vertical. For the ClosedIE setting, we train only on in-vertical sites.

- **Level III–Site-specific Knowledge:** This is the traditional setting used by two of our baselines where we have seed knowledge overlapping with the website data to allow training a specific model for the website. Whereas Level I-II are both zero-shot settings, Level III is not, as it allows site-specific training data via weak supervision. (We do not present results using full supervision from manual annotations since it is known from prior work (e.g., Gulhane *et al.* [36]) that full supervision from the target website yields highly accurate semi-structured extractors; we note that ZSCERES also achieves comparable results (~ 0.95 F1) in this setting.

We repeated our experiments 10 times and we report the results averaged across the runs. For OpenIE, we follow the "lenient" scoring method for SWDE introduced with OpenCeres, scoring an extraction as correct if the relation string matches any of acceptable surface forms listed by the ground truth for that object.

Models are constructed in PyTorch [75], with graph functions implemented in DGL [96] and optimization performed using Adam [43] and a batch size of 20. For OpenIE, we use a hidden layer size of 25 for the GAT and 100 for the feed-forward layer. For ClosedIE, we use a hidden layer size of 200 for all layers. We use a 2-layer GAT and dropout of 0.25. We obtain visual features by rendering the page using the headless Chrome browser and querying the values using Selenium[2].

Extraction Threshold: Since our zero-shot setting means we cannot use a development set of pages from the target site to tune the decision threshold, we instead set the threshold for each experiment to the value that attains the optimal F1 on the experiments where other sites were held-out.

OpenIE Postprocessing Rules: To ensure consistency among the extracted values, we keep only the highest confidence extraction in the case that the same text field is extracted as both a relation and object, or if multiple relations are extracted for the same object. In addition, some pages in the dataset contain relational tables, from which we sometimes extract the column headers as relations with the column contents as objects. While we believe a post-processing step could potentially recover these relational contents from our extractions, the SWDE data does not contain ground truth for such facts. Instead, we apply the heuristics described by [16] to identify these tables and remove them from our extractions.

5.6.4 Baselines and Models

We compare against several baselines:

Colon Baseline (OpenIE) This is a heuristic technique that identifies all text fields ending in a colon (":") and assumes they are relation strings, then extracts the text field to the right or below, whichever is closer, as the object. We consider it as Level I knowledge since it requires no training.

WEIR (OpenIE) This approach by Bronzi *et al.* [13] discovers relations by aligning multiple pages about the same entity. Because it requires sites to be grouped by vertical and uses a gazetteer list of entity names for the alignment, it has Level III knowledge.

OpenCeres (OpenIE) This applies the OpenCeres model, which requires a knowledge base matching some facts presented on the target website, using Level III knowledge.

ZSCERES-FFNN (Feed-forward neural network): This model takes the same features and training data as the full ZSCERES model but removes the GNN component, with versions tested with

[2] `https://www.seleniumhq.org`

both Level I (ZSCERES-FFNN Unseen-Vertical) and Level II (ZSCERES-FFNN Unseen-Website) knowledge.

ZSCERES-GNN: This applies the full model described in Section 5.4.2, with versions tested with both Level I (ZSCERES-GNN Unseen-Vertical) and Level II (ZSCERES-GNN Unseen-Website) knowledge.

5.7 Experimental Results

5.7.1 OpenIE

Level-I Knowledge: Table 5.2 shows that ZSCERES is able to extract facts in entirely new subject verticals 31% more accurately than the colon baseline. Across all SWDE sites (micro-averaging across all extractions), ZSCERES-GNN achieves an F1 of 0.45, in comparison with 0.43 for ZSCERES-FFNN, showing that the additional information provided by the page encoder allows for a better representation of the relationships between text fields.

By successfully learning general patterns of relational presentation on webpages, ZSCERES-GNN is able to train solely on a set of 16 websites about Movies and NBA players, and then extract from University websites more accurately than the WEIR and OpenCeres systems, which take advantage of Level III knowledge to learn models specific to those University sites. While OpenCeres's rich vertical knowledge allows it to attain better results in Movie and NBA, ZSCERES-GNN still posts much stronger results than the other baselines in these two verticals.

Level-II Knowledge: Figure 5.4 shows that adding the in-vertical sites to the training set (but still withholding the test site) allows the model to achieve performance better than the Level I training set that uses only out-of-vertical data.

5.7.2 ClosedIE

Table 5.3 shows the results for ClosedIE extraction. ZSCERES-GNN attains an overall F1 of 0.58 averaged across the three verticals. This significantly outperforms the feed-forward model that did not use the GNN, which attained an F1 of 0.46. While our performance on this dataset is far below

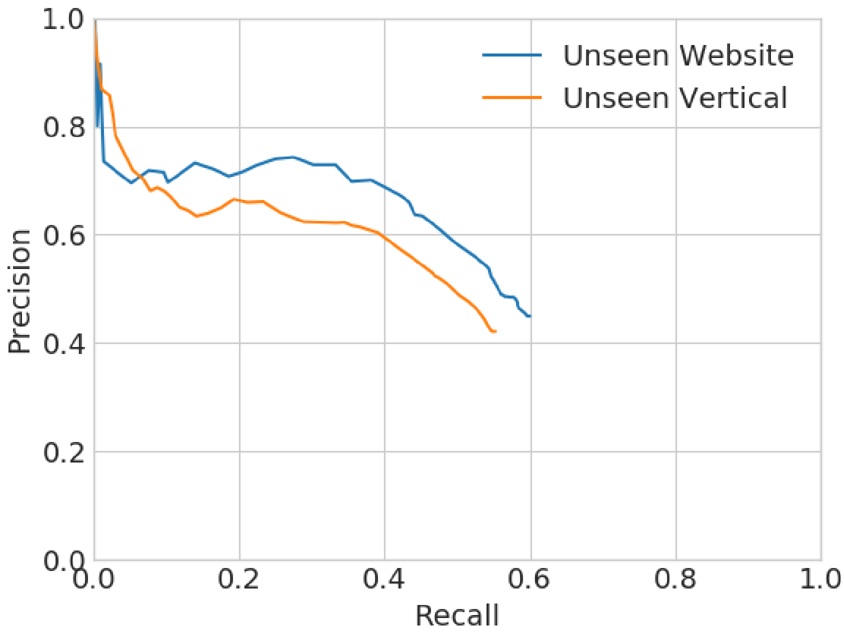

Figure 5.4: For OpenIE, using the full SWDE set (except the test site), including in-vertical training data (i.e. Level II knowledge), allows for 5-10 point gains in precision at equivalent recall compared to using only out-of-vertical training data (Level I).

the state-of-the-art for semi-structured ClosedIE (above 0.9 for all verticals), prior systems all learn site-specific models based on manual labeling or prior knowledge aligned to the website, while we have only Level II Knowledge available.

Figure 5.5 shows how adding additional training data improves performance in the Movie vertical. It appears that adding additional training sites would further improve the performance.

5.7.3 Ablation Study

Table 5.4 shows the contributions of different elements of the model in the OpenIE and ClosedIE settings as calculated on the development set of three sites in the Movie vertical. These ablations show that the GNN helps in both settings, with a larger effect in ClosedIE, which is likely due to sharing the rich information about the text of nearby text fields.

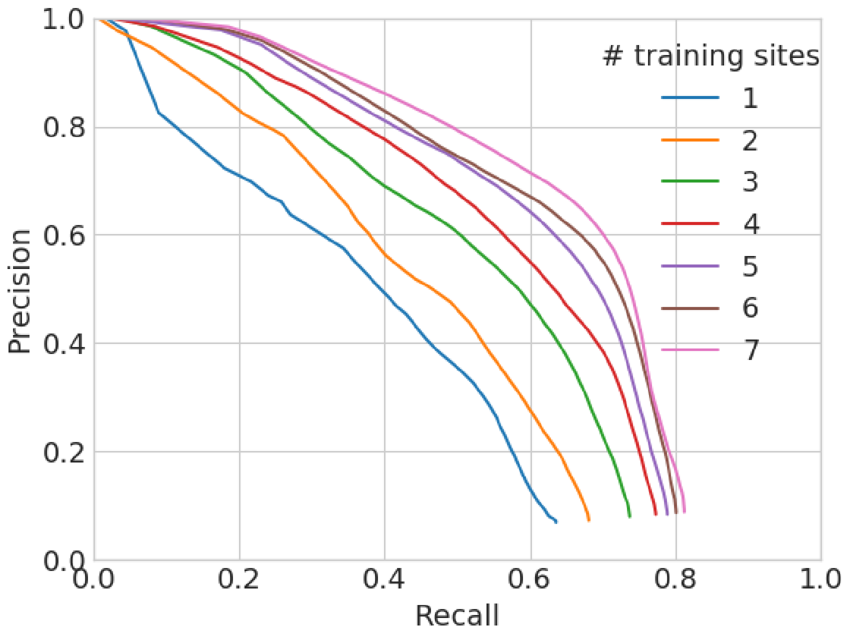

Figure 5.5: Performance on the ClosedIE Movie vertical increases significantly as more sites are added to the training data.

Pre-training is important in OpenIE but does not have a significant effect for ClosedIE. This is not surprising given that the pre-training task is closely related to the OpenIE task. Both DOM and spatial adjacency edges contribute to the success of the page layout graph for the GNN. In the ClosedIE setting, the text and layout relationships alone will generally contain sufficient information to make an extraction, while in OpenIE the visual elements (such as whether text is bold or underlined) are a strong source of consistency across websites.

5.7.4 Error Analysis

OpenIE: To understand what cases our ZSCERES-GNN model is missing, we sampled 100 error cases in each vertical from the Unseen-Vertical experiment and manually examined them. Some examples of both erroneous and correct extractions are shown in Table 5.5. False positives were largely due to the presence of two different types of n-ary relationships on the page.

The first class of errors involving n-ary relationships, making up 43% of all false positives, were where several facts have a multi-way relationship with the page topic, but individually the fields are not meaningful. For example, the NBA site USAToday includes a "Latest notes" section with links to several articles relevant to the page topic entity, mentioning the date, headline, and summary. We extract all of these objects with the "Latest notes" relation, but to obtain meaningful knowledge it would be necessary to additionally associate the correct date, headline, and summary with each other. While we can envision methods for doing this via post-processing, the SWDE benchmark considers these to be errors.

In the second class, ZSCERES correctly extracted *(relation, object)* pairs, but from page sections that contain facts about entities other than the page topic. For example, on the MatchCollege site, a section of "Similar Local Colleges" contains some of the same relations presented for the page topic, in similar formatting. These types of errors made up another 6% of false positives.

Of the remaining errors, 33% were due to the extraction of pairs where the extracted relation did not represent a relationship, while another 14% were due to the extraction of pairs with a correct relation string and incorrect object. Most false negatives occurred in long vertical lists, where some values were extracted, but not all.

ClosedIE: False negatives were most likely to occur on long lists of values (such as cast lists), where values toward the bottom of the list were sometimes missed. Recall also suffered on relations where the relation name varied significantly from site to site, or where ambiguity existed. For example, the string "Produced by" is used by some sites to indicate the producer of the film, while on other sites it indicates the production company.

5.8 Conclusions

We have introduced a zero-shot method for learning a model for relation extraction from semi-structured documents that generalizes beyond a single document template. Moreover, this approach enables OpenIE extraction from entirely new subject verticals where no prior knowledge is available. By representing a webpage as a graph defined by layout relationship between text fields, with text fields associated with both visual and textual features, we attain a 31% improvement over the baseline for new-vertical OpenIE extraction. Future extensions of this work involve a more general pre-training objective allowing for the learned representations to be useful in many tasks as well as distantly or semi-supervised approaches to benefit from more data.

Vertical	Relation
movie	movie.aka
movie	movie.box_office
movie	movie.budget
movie	movie.country
movie	movie.directed_by
movie	movie.distributor
movie	movie.genre
movie	movie.language
movie	movie.produced_by
movie	movie.production_company
movie	movie.rating
movie	movie.release_date
movie	movie.runtime
movie	movie.starring
movie	movie.synopsis
movie	movie.written_by
movie	movie.year
nbaplayer	nbaplayer.age
nbaplayer	nbaplayer.assists
nbaplayer	nbaplayer.birthdate
nbaplayer	nbaplayer.birthplace
nbaplayer	nbaplayer.college
nbaplayer	nbaplayer.draft
nbaplayer	nbaplayer.experience
nbaplayer	nbaplayer.field_goal_percentage
nbaplayer	nbaplayer.height
nbaplayer	nbaplayer.points
nbaplayer	nbaplayer.position
nbaplayer	nbaplayer.rebounds
nbaplayer	nbaplayer.weight
university	university.application_fee
university	university.calendar_system
university	university.control
university	university.enrollment
university	university.in_state_tuition
university	university.out_state_tuition
university	university.phone
university	university.religious_affiliation
university	university.setting
university	university.tuition
university	university.undergraduate_enrollment
university	university.website

Table 5.1: A listing of ClosedIE relation types mapped from OpenIE labels in SWDE

System	Site-specific Model	Level	Movie P	Movie R	Movie F1	NBA P	NBA R	NBA F1	University P	University R	University F1	Average F1
OpenCeres	Yes	III	0.71	0.84	**0.77**	0.74	0.48	**0.58**	0.65	0.29	**0.40**	**0.58**
WEIR	Yes	III	0.14	0.10	0.12	0.08	0.17	0.11	0.13	0.18	0.15	0.13
ZSCERES-FFNN Unseen-Website	No	II	0.37	0.5	0.45	0.35	0.49	0.41	0.47	0.59	**0.52**	0.46
ZSCERES-GNN Unseen-Website	No	II	0.49	0.51	**0.50**	0.47	0.39	**0.42**	0.50	0.49	0.50	**0.47**
Colon Baseline	No	I	0.47	0.19	0.27	0.51	0.33	0.40	0.46	0.31	0.37	0.35
ZSCERES-FFNN Unseen-Vertical	No	I	0.42	0.38	0.40	0.44	0.46	0.45	0.50	0.45	**0.48**	0.44
ZSCERES-GNN Unseen-Vertical	No	I	0.43	0.42	**0.42**	0.48	0.49	**0.48**	0.49	0.45	0.47	**0.46**

Table 5.2: With no vertical knowledge, ZSCERES-GNN achieves 65% higher recall and comparable precision in all verticals compared to the colon baseline. Even in comparison to approaches that use vertical knowledge to learn site-specific OpenIE models, ZSCERES achieves an F1 seven points higher in the University vertical.

System	Knowledge Level	P	R	F1
ZSCERES-FFNN	II	0.45	0.49	0.46
ZSCERES-GNN	II	0.62	0.55	**0.58**

Table 5.3: For ClosedIE, using the pre-trained GNN adds 12 F1 points in comparison to the baseline lacking contextual information.

	OpenIE F1	ClosedIE F1
Full Model	0.71	0.73
No GNN	0.68 (0.03 ↓)	0.63 (0.10 ↓)
No pre-training	0.66 (0.05 ↓)	0.73
No DOM edges	0.65 (0.06 ↓)	0.58 (0.15 ↓)
No spatial edges	0.65 (0.06 ↓)	0.62 (0.11 ↓)
No visual features	0.55 (0.16 ↓)	0.73
No BERT features	–	0.10 (0.63 ↓)
Add BERT features	0.68 (0.03 ↓)	–

Table 5.4: Ablations on the Movie development set.

Vertical	Site	Extraction			Correct	Notes
		Page Topic	Relation	Object		
Movie	Hollywood	Spanish Fly	Costume Designer	Jose Maria de Cossio	Yes	
Movie	Metacritic	Saving Face	Reviewed by	Maitland Mc-Donagh	Yes	
NBAPlayer	ESPN	Jameer Nelson	Birth Place	Chester, PA	Yes	
NBAPlayer	MSNCA	Matt Bonner	College	Florida	Yes	
University	CollegeProwler	Spring Arbor University	Admission Difficulty	Average	Yes	
University	MatchCollege	Menlo College	College Credits Accepted	AP Credit	Yes	
Movie	RottenTomatoes	Slow Burn	Tomatometer Percentage	97%	No	Subject of relation is not page topic but is an unrelated recently re-leased film
Movie	RottenTomatoes	Ginger Snaps 2	WHAT'S HOT ON RT	Trailer: Santa has a bloody Xmas	No	Extracted relation string is not a relation
Movie	Metacritic	The Constant Gardener	User Panel Options	The Constant Gardener	No	Extracted relation string is not a relation
University	CollegeProwler	Minnesota School of Business	CP Top 10 Lists	Best Performance Venues	No	Link to article not re-lated to page topic, but is a "Similar School"
University	MatchCollege	Maric College	Highest Degree	Associate's	No	Subject of relation is not page topic
NBAPlayer	FoxSports	Tony Parker	Latest News	Mon. Dec 6, 2010	No	n-ary object
NBAPlayer	MSNCA	Gilbert Arenas	Birthplace	215	No	Erroneous extraction of weight for birthplace (both text fields are nearby)

Table 5.5: Selected OpenIE Extractions from OpenCeres-GNN with Level I training (no knowledge of the subject vertical).

Chapter 6

CONCLUSION

This thesis has presented three methods for information extraction from semi-structured webpages:

1. Ceres: Ceres is a system that uses distant supervision to automatically extract information without requiring any manually generated training data. Using an existing knowledge base (KB), Ceres automatically creates training data for a new website by harnessing the consistency present from page to page on the site to find entity mentions that are likely to match known relations. Ceres then uses this (noisy) training data to learn an extraction model consisting of a multi-class classifier using logistic regression. This model performs well in the case that a rich KB exists that covers all desired relation types and overlaps with information presented on the website.

2. OpenCeres: OpenCeres expands Ceres to allow for Open Information Extraction, allowing the extraction of facts for relationships that did not exist in the seed KB. OpenCeres uses the visual consistency of relation representation on a webpage to identify likely key-value pairs based on their layout similarity with known key-value pairs. This allows for extraction of new relation types on websites where Ceres was able to successfully create training data.

3. ZeroShotCeres: ZeroShotCeres considers a setting where we have a website that may not have knowledge overlap with an existing KB, or may correspond to an entirely new knowledge domain. ZeroShotCeres takes advantage of the visual semantics of webpage layout to learn to recognize key-value pairs. ZeroShotCeres builds a graph representing (both visual and DOM) adjacency relationships on the page, and then operates over this representation

with a graph neural network. This allows for a single model to be learned based on a set of labeled training data, and then that model can be applied to websites not in the training set.

Taken together, these three methods allow for both traditional and open information extraction in many settings.

6.1 Future Work

There are many ways to build upon this line of work in the future. Large-scale self-supervised pre-training has shown great success in learning representations of text, e.g. BERT [24]; a similar method could be applied in the web setting, learning representations of webpages (including HTML structure) based on a large unlabeled corpus. This could then improve the generalizability of models trained on a smaller corpus of labeled data. There are also those directions that involve the alignment and use of extracted data. The extraction of large numbers of OpenIE triples from the Web raises the question of how the information in these triples can be aligned to a pre-existing schema or have a new schema automatically extracted from it. Semi-structured OpenIE triples in particular have a consistency not present in OpenIE triples extracted from natural language text, potentially allowing for easier alignment, such as that explored in [103]. In addition, the population of extracted data into a knowledge graph requires the linking of entities from the extracted triples to existing entities in the KG, a challenging problem. The eventual existence of a large KG extracted from the Web would open up many avenues for exploration in using the extracted knowledge in downstream applications.

BIBLIOGRAPHY

[1] Commoncrawl corpus. http://www.commoncrawl.com, 2017.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *PVLDB*, page 487–499, 1994.

[3] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, pages 337–348. ACM, 2003.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] Yonatan Aumann, Ronen Feldman, Yair Liberzon, Binyamin Rosenfeld, and Jonathan Schler. Visual information extraction. *Knowledge and Information Systems*, 10:1–15, 2006.

[6] Nguyen Bach and Sameer Badaskar. A review of relation extraction. 2007.

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

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

[9] Lidong Bing, Sneha Chaudhari, Richard C. Wang, and William W. Cohen. Improving distant supervision for information extraction using label propagation through lists. In *EMNLP*, 2015.

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

[11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[12] Andrei Z. Broder. On the resemblance and containment of documents. *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, 1997.

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

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

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

[16] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. In *WebDB*, 2008.

[17] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[18] Chia-Hui Chang, Mohammed O. Kayed, Moheb R. Girgis, and Khaled Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18:1411–1428, 2006.

[19] Joseph Paul Cohen, Wei Ding, and Abraham Bagherjeiran. Semi-supervised web wrapper repair via recursive tree matching. *CoRR*, abs/1505.01303, 2015.

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

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

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

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

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2018.

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

[26] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. From data fusion to knowledge fusion. *PVLDB*, 7(10):881–892, 2014.

[27] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: estimating the trustworthiness of web sources. *PVLDB*, 8(9):938–949, 2015.

[28] Xin Luna Dong and Divesh Srivastava. *Big Data Integration (Synthesis Lectures on Data Management)*. Morgan Claypool Publishers, 2015.

[29] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *KDD*, 2008.

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

[31] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301–323, 2014.

[32] Freebase. Compound value types in Freebase API, 2018.

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

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

[35] Ralph Grishman. Information extraction. In *The Oxford handbook of computational linguistics*. Oxford University Press, 2012.

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

[37] Pankaj Gulhane, Rajeev Rastogi, Srinivasan H. Sengamedu, and Ashwin Tengli. Exploiting content redundancy for web information extraction. *PVLDB*, 3(1):578–587, 2010.

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

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[40] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL-HLT*, 2011.

[41] Robin Jia, Cliff Wong, and Hoifung Poon. Document-level n-ary relation extraction with multiscale representation learning. In *NAACL-HLT*, 2019.

[42] Anoop R. Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. Chargrid: Towards understanding 2d documents. In *EMNLP*, 2018.

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[44] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[45] Furkan Kocayusufoglu, Ying Sheng, Nguyen Vo, James Bradley Wendt, Qi Zhao, Sandeep Tata, and Marc Najork. Riser: Learning better representations for richly structured emails. In *WWW*, 2019.

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

[47] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.

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

[49] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[50] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966.

[51] Furong Li, Xin Luna Dong, Anno Largen, and Yang Li. Knowledge verification for long tail verticals. *PVLDB*, 10(11):1370–1381, 2017.

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

[53] Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. Heterogeneous supervision for relation extraction: A representation learning approach. In *EMNLP*, 2017.

[54] Shengpeng Liu, Ying Li, and Binbin Fan. Hierarchical RNN for few-shot information extraction learning. In *ICPCSEE*, 2018.

[55] Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. Graph convolution for multimodal information extraction from visually rich documents. In *NAACL-HLT*, 2019.

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

[57] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In *NAACL-HLT*, 2019.

[58] Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In *ACL*, 2020.

[59] Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. A general framework for information extraction using dynamic span graphs. In *NAACL*, 2019.

[60] Mausam. Open information extraction systems and downstream applications. In *IJCAI*, 2016.

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

[62] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013.

[63] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*, 2009.

[64] Marcin Mironczuk. The biggrams: the semi-supervised information extraction system from html: an improvement in the wrapper induction. *Knowledge and Information Systems*, pages 1–66, 2017.

[65] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, 2015.

[66] Ion Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the third annual conference on Autonomous Agents*, pages 190–197. ACM, 1999.

[67] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM '11*, 2011.

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

[69] Feng Niu, Ce Zhang, Christopher Ré, and Jude W. Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *Int. J. Semantic Web Inf. Syst.*, 8:42–73, 2012.

[70] Dan Olteanu, Holger Meuss, Tim Furche, and FranÃ§ois Bry. Xpath: Looking forward. In *EDBT Workshops*, 2002.

[71] Stefano Ortona, Giorgio Orsi, Marcello Buoncristiano, and Tim Furche. Wadar: Joint wrapper and data repair. *PVLDB*, 8(12):1996–1999, 2015.

[72] Stefano Ortona, Giorgio Orsi, Tim Furche, and Marcello Buoncristiano. Joint repairs for web wrappers. In *ICDE*, pages 1146–1157, 2016.

[73] Panupong Pasupat and Percy Liang. Zero-shot entity extraction from web pages. In *ACL*, 2014.

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

[75] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[77] Giuseppe Della Penna and Sergio Orefice. Supporting information extraction from visual documents. *Journal of Computer and Communications*, 4(6):36–48, 2016.

[78] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[79] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *NAACL*, 2018.

[80] Pliny the Elder, John Bostock, and Henry Thomas Riley. *Delphi Complete Works of Pliny the Elder*. Delphi Classics, Hastings, UK, 2015.

[81] Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. GraphIE: A graph-based framework for information extraction. In *NAACL-HLT*, 2019.

[82] Disheng Qiu, Luciano Barbosa, Xin Dong, Yanyan Shen, and Divesh Srivastava. Dexter: Large-scale discovery and extraction of product specifications on the web. *PVLDB*, 8(13):2194–2205, 2015.

[83] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, November 2017.

[84] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *NeurIPS*, pages 3567–3575, 2016.

[85] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *ECML/PKDD*, 2010.

[86] Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning*, pages 986–987. Springer US, Boston, MA, 2010.

[87] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *European Semantic Web Conference*, pages 593–607, 2017.

[88] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.

[89] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

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

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

[92] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*, 2012.

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[94] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.

[95] James Vlahos. Amazon Alexa and the search for the one perfect answer. *Wired*, 27(3), March 2019.

[96] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[97] Tak-Lam Wong and Wai Lam. Learning to adapt web information extraction knowledge and discovering new attributes via a bayesian approach. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(4):523–536, 2010.

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

[99] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. Fonduer: Knowledge base construction from richly formatted data. *SIGMOD*, 2018:1301–1316, 2018.

[100] Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Fernando Campos, and Arnold Overwijk. Open domain web keyphrase extraction beyond language modeling. In *EMNLP/IJCNLP*, 2019.

[101] Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *WWW*, pages 76–85. ACM, 2005.

[102] Yanhong Zhai and Bing Liu. Extracting web data using instance-based learning. *World Wide Web: Internet and Web Information Systems*, 10(2):113–132, 2007.

[103] Dongxu Zhang, Subhabrata Mukherjee, Colin Lockard, Luna Dong, and Andrew McCallum. OpenKI: Integrating open information extraction and knowledge bases with relation inference. In *NAACL*, pages 762–772, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.