Using Latent-Structure to Detect Objects on the Web

Luciano Barbosa
AT&T Labs - Research
lbarbosa@research.att.com

Juliana Freire
University of Utah
juliana@cs.utah.edu

ABSTRACT

An important requirement for emerging applications which aim to locate and integrate content distributed over the Web is to identify pages that are relevant for a given domain or task. In this paper, we address the problem of identifying pages that contain objects with a *latent structure*, i.e., the structure is implicitly represented in the page. We propose an algorithm which, given a set of instances of an object type, derives rules by automatically extracting statistically significant patterns present *inside* the objects. These rules can then be used to detect the presence of these objects in new, unseen pages. Our approach has several advantages when compared against learning-based text classifiers. Because it relies only on positive examples, constructing accurate object detectors is simpler than constructing learning classifiers, which require both positive and negative examples. Also, besides providing a classification decision for the presence of an object, the derived detectors are able to pinpoint the location of the object inside a Web page. This enables our algorithm to extract additional object fragments and apply online learning to automatically update the rules as new documents become available. An experimental evaluation, using a representative set of domains, indicates that our approach is effective. It is able to learn structural patterns and derive detectors that outperform state-of-art text classifiers and the online learning component leads to substantial improvements over the initial detectors.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Selection process.

General Terms

Algorithms, Design, Experimentation.

Keywords

Web objects, rule inference, online learning, information extraction

1. INTRODUCTION

There has been a substantial increase in the volume of (semi-) structured data on the Web. Not surprisingly, several applications have emerged that aim to organize structured Web data and make them more easily accessible and queryable. These include large-scale information integration systems [3, 13], specialized search engines, and structured data portals [8, 15, 5]. An important requirement for these applications is the ability to find new information sources, be it to increase the coverage for domains (or object classes) they already support, or to find new kinds of objects. Consider for example the DBLife system [5]. DBLife integrates online information that is relevant to the database research community, presenting a unified view of entities (e.g., conference announcements, researchers, references to scientific articles) and relationships in the domain (e.g., served on PC, published a paper, collaborated with). To keep the data repository current, the system must be able to locate new entity instances (e.g., new papers, researchers) and if there is a need expand the repository, it must locate new entities that are relevant in the domain (e.g., Web casts, videos).

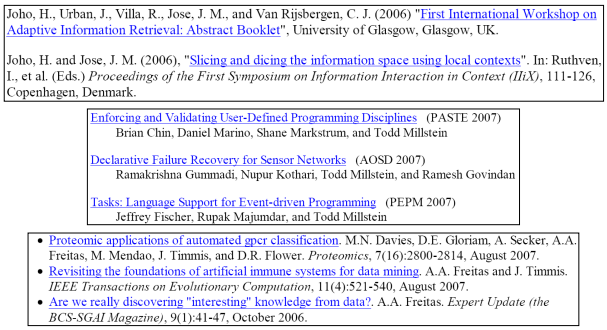


Figure 1: Variability in formatting of bibliographic references.

In this paper, we examine the problem of identifying Web entities (or objects) consisting of a series of attributes. For instance, a bibliographic reference is an object composed of attributes such as author, venue, year, etc. These attributes may contain some internal structure: while *year* is a string that contains 4 digits, *venue* often contains terms such as "conference" or "journal". Identifying these objects is challenging because there is wide variability in how they are represented across different sites. Consider the bibliographic references shown in Figure 1. Different people use different conventions to represent a reference. While some put the title, followed by the authors and then the conference and year, others start with authors. There are also many alternatives used for abbreviating certain attributes (e.g., SIGMOD Conference, SIGMOD Conf.) and sometimes attribute names are omitted (e.g., SIGMOD).

A possible solution to detect Web objects is to use traditional text classification techniques [17], but these have important limitations. Structured objects are often located pages that contain other information. Figure 2 shows a recipe page. Besides the recipe object, there are also ads, navigation bars, and forms. Because in text classification, documents are viewed as a bag of words, without

WebDB '10 Indianapolis, IN USA

Figure 2: Page containing a recipe and additional content.

a reliable mechanism to locate the recipe within the page, it can be hard to construct an accurate classifier. Furthermore, building a learning classifier often requires a large number of both positive and negative examples. Constructing an accurate classifier is thus not only time consuming but also very challenging. Since there are so many distinct objects on the Web, given a target object, it is not clear what other objects would constitute a representative set of negative examples that would help in the classification process.

Detecting Objects with Latent Structure. We propose a new and scalable approach for detecting latent-structured objects (*LSOs*) in Web pages. Given samples of objects in a class or domain, Latent Structure Object Detection (*LSOD*) *uses hypothesis testing to identify patterns (the latent structure) that are statistically significant, and derives a small set of rules based on these patterns*. Similar to Class Sequential Rules (CSR) [12], we want to create rules based on sequential patterns. In our context, a sequence is an ordered list of items that represent a fragment of an object. For instance, in the recipe domain, the fragment "1 cup of milk" can be represented by the sequence <1, cup, of, milk >. *LSOD* infers rules (sequences) from the input fragments (or sentences) that are able to detect *similar* objects in other pages.

In contrast to approaches to grammar and DTD inference [1, 6], *LSOD* does not aim to derive rules that capture a formal description of the *whole object* (generative model). Instead, our goal is to derive rules that yield high precision but that also capture the wide variability in content and structure of objects, and as a result, are able identify objects across multiple Web sites (descriptive model).

Besides providing a scalable mechanism for detecting documents that contain structured Web objects, *LSOD* can be used to harvest these objects. Because the derived rules capture the structure of the object, they can be used to *locate the object in a page*. This helps both the page segmentation and object extraction processes. The ability to locate the object within a document also enables *LSOD* to *incrementally and automatically update the detection rules through online learning*. In addition to reducing the costs involved in rule maintenance, online learning can attenuate potential biases present in the initial model and increase rule accuracy and coverage.

Contributions and Outline. Our main contributions are summarized as follows:

- We propose a new algorithm for automatically deriving detection rules for structured Web objects. Relying only on positive examples, the algorithm uses statistical methods to derive rules that are both general and accurate (Section 2). We show how these rules can be used to determine whether a page contains a desired object as well as to locate the object in the page (Section 3);
- We discuss how online learning can be used to automatically improve the detection rules (Section 4);
- We present experiments which demonstrate that our approach is effective and outperforms state-of-the-art text classification techniques (Section 5).

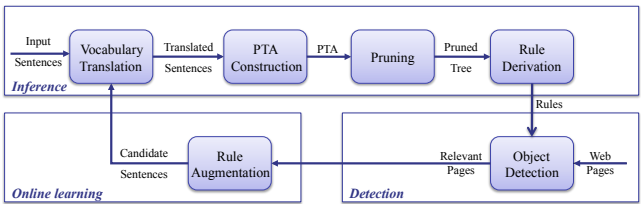


Figure 3: System architecture.

2. LATENT STRUCTURE INFERENCE

Given a set of sentences S that represent instances of objects or object fragments in a domain of interest D, the inference process must identify patterns and generate rules that recognize these patterns. These rules, in turn, can be used to classify sentences as belonging to D. As illustrated in Figure 3, the inference process of *LSOD* consists of four steps. During *Vocabulary Translation*, input sentences from S are translated using a more abstract and restricted vocabulary to capture strong patterns. In *PTA Construction*, the translated sentences are merged into a prefix tree acceptor (PTA). Because the PTA can be over-specialized with respect to the input set of sentences, *Pruning* eliminates some of the PTA branches to make it more general. In the final step, *Rule Derivation*, rules are extracted from the pruned tree to be used in the *Object Detection*.

2.1 Vocabulary Translation

Given the high variability in the vocabulary used in sentences of latent-structured objects (LSOs), to capture *strong* patterns, we adopt a more abstract representation for these sentences. We define a Vocabulary Translation Function (VTF) which maps the words present in the input sentences to a more restricted vocabulary, $VTF : V \mapsto V'$, where V is the set of words extracted from the training sentences (after stemming and stopword removal) and V' is the set of words in the restricted vocabulary.

Definition 1: [Restricted Vocabulary V'] The vocabulary V' consists of a pre-defined set of types T and the set of k words with the highest document frequencies (df_k). The pre-defined types are:

- UPPER: words whose first letter is capitalized;
- LOWER: words whose first letter is in lower case;
- N-DIGIT: token composed of n digits where;
- ALPHANUM: token composed of digits/non-digits. ■

Example 1: Let S be the set of input sentences in the original vocabulary V (after stopword removal and stemming):
S_1 = 1 cup water
S_2 = 10 cup wine
Assuming that "cup" is a word with high frequency, translating S_1 and S_2 to $V' = \{T, cup\}$ we have:
S_1 = 1-DIGIT cup LOWER
S_2 = 2-DIGIT cup LOWER ■

Note that, because the restricted vocabulary contains frequent words present in the input sentences, it is *domain-dependent*. An important benefit of using domain-specific words is that they capture specific information (both from the content and structure) of the domain objects.

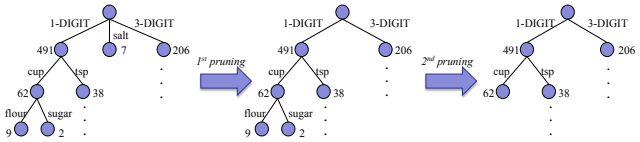


Figure 4: Example of the pruning process from the initial PTA to the resulting pruned tree.

2.2 PTA Construction and Pruning

After vocabulary translation, the inference process derives object detection rules that achieve a balance between recall and precision. To discover common patterns in the set of translated sentences, we build a prefix tree (aka trie). In a prefix tree, the root node corresponds to the empty string and all descendants of a node share the same prefix. Each node in the tree is annotated with its frequency in the set of input sentences. As this tree accepts all the input sentences, we call it Prefix Tree Acceptor (PTA) (see Figure 4).

To avoid the creation of PTAs that are too large and specialized with respect to the input sentences, the inference process removes branches that are not significant by hypothesis testing [16]. The null hypothesis H_0 is that the observed instances of a prefix Γ were generated by chance. H_0 is rejected if the probability (p-value) of observation O given H_0 (the probability of Γ having been created by chance) is extremely low, in which case Γ is considered statistically significant. To determine whether the p-value is small enough, we compare it to a significance value α.

Similar to [11], to calculate the p-value, we assume that adding a word w to an already significant prefix Γ by chance has a binomial distribution. The binomial distribution approaches a normal distribution N(μ,σ), with mean μ and variance σ, for a large number n of observations with $\mu = np$ and $\sigma^2 = np(1-p)$. Given a new word w, with probability of occurrence $p = count(w)/\sum_{i=0}^{t} count(w_i)$, the probability of choosing w to be the next token to the prefix Γ, with t occurrences in k of these prefixes, is the cumulative probability of observing at least x events: $P(k \geq x) = \int_x^{\infty} \varphi_{\mu,\sigma^2}(u)du$, where φ_{μ,σ^2} is the probability density function of the normal distribution with mean μ and variance σ.

Using this framework, we perform two pruning steps. The first one checks whether each node is statistically significant. The pruning starts from the prefixes in the first level of the PTA. If a prefix is not statistically significant (according to the hypothesis test described above), all of its descendants are removed. This pruning is applied in a breadth-first fashion until the leaves of the PTA are reached.

Example 2: Let p_1 be the root of the initial PTA. We want to check whether the specialization of p_1 , p_2 = "salt", is significant (see Figure 4). Given that the frequency of p_1 is 704 (n=704), the probability of token "salt" is 0.01 (p=0.01), the frequency of p_2 is 7 (k=7) and the α value is 0.05. Assuming $N(\mu = 7.04, \sigma = 2.64)$, the cumulative probability of at least 7 events being observed is 0.51, and it is higher than $\alpha = 0.05$. As a result, p_2 is considered not statistically significant and, therefore, removed from the tree. ■

We apply a second pruning step which verifies whether specializations (children) of the prefix are significant. More specifically, instead of checking whether the frequency of a given prefix is statistically significant as in the first pruning step, the second step verifies whether the number of sentences covered by the prefix minus the number of sentences covered by its children is significant. For instance, in the example of Figure 4, the specializations "flour" and "sugar" of "1-DIGIT cup" were considered non-significant and as result they were pruned.

The tree generated by the pruning process is significantly more concise than the initial PTA. This tree is used to derive the detection rules. As a final step, rules are eliminated if they have only one element, or if they contain only terms from the generic vocabulary (e.g., UPPER, LOWER). Such rules contain patterns that are likely to occur in many different kinds of objects (negative examples), resulting in low precision.

2.3 Capturing Deep Patterns

The inference process described in the previous subsections is effective if the relevant patterns are concentrated close to the root of the tree. However, when these patterns occur deeper in the tree, they are harder to identify due to fragmentation, i.e., the presence of terms with low frequency deep in the branches of the PTA. Fragmentation leads to the derivation of rules that are too generic and thus, have low precision.

Example 3: Let S_1, S_2 and S_3 be sentences in the restricted vocabulary $V' = \{T, conference, data, using\}$:
S_1 = UPPER UPPER conference 4-DIGIT
S_2 = UPPER UPPER conference using 4-DIGIT
S_3 = UPPER UPPER conference data 4-DIGIT
The inference process would derive the rule "UPPER UPPER conference", which might be too generic. However, if the low frequency words ("using" and "data") were removed, a more precise rule would be inferred "UPPER UPPER conference 4-DIGIT". ■

A naïve way of avoiding fragmentation would be to select only a few of the most frequent words in the vocabulary. For example, if the words "using" and "data" are not among the top-k selected words, they would be eliminated from the sentences. However, fragmentation can also be caused by the pre-defined types in the restricted vocabulary V' (Definition 2.1). A more robust approach, and the one we adopt, is to consider in the sentences only the *bigrams with high-frequency*—the top-k bigrams. Since our goal is to keep words that appear co-occur frequently, we consider every word pair in a sentence as bigrams, so-called bigrams at a distance.

Definition 2: [Bigram at a Distance] Let $S = \{w_1, w_2, ..., w_n\}$ be a sentence. The bigrams at a distance that can be derived from S are all "$w_i w_j$" $\in S$ where i < j. ■

After computing the bigrams at a distance for all sentences, the bigram removal algorithm works as follows. It starts from the first word in the sentence and verifies whether the bigram "$word_1 word_2$" belongs to the list of frequent bigrams. If so, $word2$ is kept and then the next bigram "$word_2 word_3$" is tested. If not, the algorithm checks the bigram "$word_1 word_3$", and so on.

Example 4: From the sentences in previous example, we have the following bigrams: (UPPER UPPER,3); (UPPER conference,3); (conference 4-DIGIT,3); (conference using,1); (using 4-DIGIT,1); (conference data,1); (data 4-DIGIT,1]). Assuming the frequent bigrams as those whose frequency is higher than 1, the resulting sentences after the bigram removal are:
S_1 = UPPER UPPER conference 4-DIGIT
S_2 = UPPER UPPER conference 4-DIGIT
S_3 = UPPER UPPER conference 4-DIGIT
Now, the more precise rule "UPPER UPPER conference 4-DIGIT" can be inferred from these sentences. ■

To determine a suitable value of k, we execute the inference process with different values and select the output that contains rules with larger average sizes. Our assumption is that larger rules result in higher precision. If there are multiple rule sets with high average size, we pick the set with a larger number of rules since it has a higher probability of obtaining better recall.

3. LSO DETECTION

To apply the rules derived by the inference process, we must first segment the page and identify the sentences where the rules can be

applied. Segmentation is critical for matching. Notably, it must avoid breaking sentences into pieces that cannot be recognized by the rules and combining sentences that contain pieces of multiple instances of an object.

We use the layout of a Web page to guide the segmentation process. The tokenizer breaks a document based on the HTML tags that correspond to the newline characters: "p","dt","dd","tr","li", "br" and all header tags ("h1","h2", etc). After removing all of the HTML tags, pieces of text located between these special tokens are considered to be sentences. A potential problem with this approach is that, sometimes, there are line breaks within a sentence, e.g., search engine results pages often split an object (i.e., a result) into multiple lines. In these cases, as the sample sentences provided to the inference process also follow this pattern, the rules generated capture sub-structures present in each line.

After segmenting a page into sentences, we apply the object detection rules. A rule matches a sentence if the first two words in the sentence match the first two elements in the rule, and the remaining elements appear in the sentence in the same order, regardless of their absolute position. If any sentence in a page p matches a rule that describes an object in D, we assume $p \in D$.

Example 5: Given the vocabulary $V' = \{T, conference\}$ (see Definition 2.1), suppose we want to verify whether rule R="UPPER UPPER conference 4-DIGIT" matches the sentence S="John Johnson, Grammar Inference, In Grammar Conference, 2004". We first remove stopwords, map S into V' generating S'="UPPER UPPER UPPER UPPER UPPER UPPER conference 4-DIGIT". S' matches R since $s_1 = r_1$ and $s_2 = r_2$, and $s_8 = r_3$ and $s_9 = r_4$. ∎

4. AUTOMATIC RULE AUGMENTATION

Because *LSOs* from different sites are often diverse, constructing a representative training set is challenging and time consuming. *LSOD* attenuates this problem by applying online learning: it automatically augments the derived rules using information obtained from newly extracted sentences discovered in the classification process. In particular, we leverage the fact that multiple sentences (corresponding to multiple objects or fragments of an object) appear in a single page. When a page is classified as belonging to the target domain by the LSO detection, the rule augmentation process (RAP) extracts candidate sentences which are co-located with the relevant sentences and add them to the training set. The augmented training set is then used to derive a new set of rules. Note that RAP is only effective if the sentences detected by the initial rules co-occur in the same document with sentences whose patterns were not covered by the initial rules.

To identify candidate sentences, RAP performs segmentation. More specifically, RAP looks at the DOM tree for the sentences that share the same parent node with the sentences considered as relevant by the rules. Although it is not always the case that all candidate sentences in the document share the same parent and it can happen that non-relevant sentences are siblings of relevant ones, as our results in Section 5 show, this simple strategy is effective in practice. Even when just a subset of the candidate sentences are siblings, they are often sufficient to increase the number of rules and consequently improve their performance.

5. EXPERIMENTS

Structured Domains. For our experimental evaluation, we selected and crawled sites in 4 different domains that contain objects with a latent structure. Three of the domains are commercial (recipes, digital cameras and movies) and one is academic (bibliographic references). As Table 1 shows, the number of sites crawled per domain varied from 40 to 233, and roughly 3000 pages were retrieved. Among these, 1270 were relevant and contained the target objects. Table 1 also shows the average sentence size for each domain (total number of words after stopword elimination). Note that sentences in bibliographic references are much longer than those in other domains, and thus more likely to contain deeper patterns.

			Test data (# Pages)	
Domain	**Avg Sent. Size**	**# Sites**	**Relevant**	**Non-relevant**
Recipe	3.8	233	654	488
Bibliography	8.3	550	200	453
Digital camera	4.3	38	226	296
Movie	3.4	40	190	183

Table 1: Details about the data used in each domain.

Baseline and Training Setup. We compared the following approaches:

- **One-class SVM** [14]: One-class SVM creates a classifier using only positive examples. We used the LIBSMV tool [2] and the positive examples were the same used by *LSOD*. Note that the examples provided consist of entire pages, not only the *LSO* sentences;
- **SVM** [9]: We used Weka [17] to build an SVM text classifier from 70 positive (same set as provided to *LSOD*) and 70 negative examples randomly selected from the crawl data. For each domain, we varied training parameters (type of kernel and C value) and selected the ones with the lowest 10-fold cross-validation error. Note that since SVM [9] requires both positive and negative examples, it should not be directly compared to *LSOD*, which relies only on positive examples. We use this configuration as a baseline for the best performance using labeled data and the state-of-the-art method for text classification;
- ***LSOD*** and ***LSOD* +RAP**: *LSOD* corresponds to our approach using sentences from 70 positive example; and *LSOD* +RAP, uses rules derived by the inference process applied to the input sentences and the sentences extracted by the Rule Augmentation Process (RAP) in one iteration.

Effectiveness of Derived Rules. As Table 2 shows, *LSOD* and *LSOD* +RAP obtained high values of F-measure (between 0.83 and 0.92), indicating that our approach is very effective for these domains. The high-precision of the derived rules suggests that they can be used to build high-quality repositories of pages that contain the target objects.

An interesting observation about the domains that had the lowest precision (bibliographic reference and movies) is that in these domains, there are pages that contain different concepts with similar structure. For instance, in the movie specification domain (precision=0.88), some of the derived rules also matched specifications of CD albums. A similar behavior was also observed in the bibliographic reference domain (precision=0.91), where Web pages for faculty members contain objects that have structure similar to publications, including talks and grants.

The *LSOD* configuration outperforms the One-class SVM classifier by a great margin: the F-measure values are between 2.3 (Movie) and 6.8 times (Camera) the ones for the One-class SVM classifier. In most domains, *LSOD* also outperforms SVM. The only exception was recipes, where the F-measure for *LSOD* is a bit lower (0.91 versus 0.92 for SVM). Note however that *LSOD* obtained perfect precision (precision=1). Although the differences between *LSOD* and SVM are relatively small, a much greater effort is needed to construct the SVM classifier, since it requires negative examples in addition to positive ones. Besides being easier to configure, our approach is able to identify the actual objects in the page. This in contrast to the SVM classifier which views a page as a bag-of-words and would require an additional step to identify

	Bibliographic Reference			Camera Specification			Movie Specification			Recipe		
Technique	**Recall**	**Prec.**	**F-meas.**	**Recall**	**Prec.**	**F-meas.**	**Recall**	**Prec.**	**F-meas.**	**Recall**	**Prec.**	**F-meas.**
One-class SVM	0.22	0.16	0.19	0.08	0.75	0.14	0.35	0.42	0.38	0.68	0.26	0.38
SVM	0.82	0.71	0.76	0.85	0.97	0.91	0.80	0.86	0.83	0.92	0.92	0.92
LSOD	0.85	0.81	0.83	0.87	0.98	0.92	0.81	0.88	0.85	0.84	1	0.91
LSOD +RAP	0.81	0.88	0.84	0.94	0.98	0.96	0.93	0.87	0.89	0.90	0.99	0.94

Table 2: Recall, precision and F-measure of all approaches over different domains.

and extract the relevant data. Another advantage of our approach is the fact that the derived rules are simple and easy-to-understand by users. Some examples of rules include: “1-DIGIT clove garlic" for recipe and “UPPER UPPER workshop 4-DIGIT" for bibliographies.

Dealing with Fragmentation. As we discussed in Section 2.3, fragmentation leads to weak patterns. In bibliographic references, for instance, without removing any bigram and varying the top-n words in the vocabulary V' from 1 to 100, rules are only derived when n is lower than 5. For n greater than 5, the patterns generated are not statistically significant and therefore are removed by the pruning process. But even when rules are generated ($1 < n < 5$), the derived rules are too generic and do not contain sufficient domain-specific information, as a result they have low precision. For instance, when the vocabulary V' contains only the top-4 words, the precision obtained is only 0.41, while the recall 0.925.

We then selected the top-100 most frequent words and varied the threshold for bigram removal. Figure 5(a) presents the average size of the rules and Figure 5(b) the total number of rules created with the frequent words for different values for the minimum bigram frequencies. With no bigram removal (minimum frequency = 0), no rules are derived. As the minimum bigram frequency increases, the average size of the rules increases since noisy bigrams are removed. When the minimum frequency reaches 100, the number of rules reaches a peak and then starts to decrease until a point where no rules are generated, after all the ‘important’ bigrams are removed (bigram > 180). As expected, the number rules derived decreases as the minimum bigram frequency increases. Figure 5(c) shows impact of bigram removal process on the performance of the rules. Initially, as there are many rules, the recall is high and precision is low. As the minimum bigram size increases, many bigrams that cause fragmentation are removed. As a result, the recall goes down and precision increases because fewer and bigger rules are derived (see Figure 5(c)). When the minimum bigram frequency is 100, precision and recall reach their highest and lowest values (respectively). After this point, important bigrams start to be removed and the rules become more generic (recall increasing and precision decreasing) until the point when no rules can be generated (bigram frequency > 180).

The behavior of the recall and precision curves are reflected in the F-measure curve. As the values of precision and recall are similar around minimum bigram frequency 60 and 80, the F-measure reaches its peak (0.83). These results show that for domains which contain patterns deep in sentences, appropriately handling fragmentation is required in order to obtain effective rules.

Note that for the three commercial domains, since strong patterns are located close to beginning of the sentence, fragmentation is not an issue. In practice, one can always assume that there is fragmentation and perform the process described in Section 2.3. If no rules are generated when the minimum bigram frequency is increased, then it means that there are not many bigrams which cause fragmentation. In which case, bigrams need not be removed.

Rule Augmentation. To assess the effectiveness of the Rule Augmentation Process (RAP) (Section 4), we ran it over the test data to verify whether it was able to learn new patterns and improve rule quality. Table 2 gives the recall, precision and F-measure obtained by the *LSOD* +RAP configuration for the 4 domains, and shows that the system is able to learn new patterns and refine existing patterns, and as a result, derive rules that have higher recall and precision. In the commercial domains, the improvement in the F-measure values are mainly due to the increase in recall: from 0.81 to 0.93 for movie specifications; from 0.84 to 0.9 for recipe; and from 0.87 to 0.94 for camera specification. For these domains, the recall increased because new rules were discovered by RAP and added to the initial rule set. Note that this also led to a slight decrease in precision, but not enough to harm F-measure.

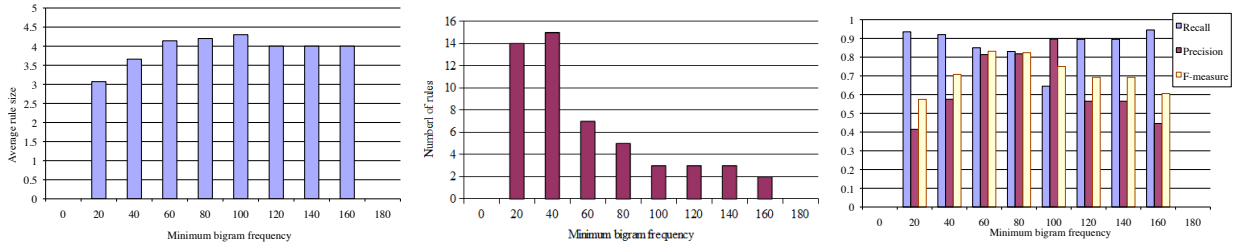
The opposite effect was observed for bibliographic references: the precision increased from 0.81 to 0.88 and recall decreased from 0.85 to 0.81. Since this domain contains deep patterns, more specialized rules were generated. Whereas the average rule size of *LSOD* was 4.2, for *LSOD* +RAP the value was 4.5. For example, the rule “UPPER UPPER confer 4-DIGIT" was specialized to “UPPER UPPER confer 2-DIGIT 4-DIGIT”. It is important to point out that in this domain, the pages are often manually created. This is in contrast to the pages in commercial domains, which are often generated from fixed templates. As a result, pages that contain bibliographic references are harder to segment. But as the numbers show, the improvement of F-measure confirms that the segmentation process performed by the RAP is able to identify relevant sentences surrounding the ones matched by the rules and learn new relevant patterns.

6. RELATED WORK

Although our solution to classify pages containing objects using their latent-structured objects is related to information extraction techniques, there are key differences. We view objects as blobs and ignore the details of their internal structure, and our goal is different: we want to simply *detect* whether a pattern that represents a Web object is present in a page. This is in contrast to information extraction techniques which aim to extract the object and its internal fields (e.g., a street address field contains a number followed by the street name).

Similar to *LSOD*, Web wrapper induction techniques also assume that Web objects follow a template [10, 4]. These techniques, however, explore the HTML structure of pages (or subsets of pages) in a Web site to infer a template and extract data records. RoadRunner [4], for example, exploits similarities among Web pages to induce a grammar for extracting objects that occur in these pages. The extraction rules need to be precise and fine-grained so that individual attributes of an object can be identified and extracted. As a result, rules must be derived for a page collection with a fairly regular HTML structure, and they are unlikely to work for other pages that contain even slight variations in the structure. In contrast, since our goal is just to identify objects, our rules can be more general and applied to objects on different sites and with heterogeneous structure.

Also related to our work are pattern learning techniques [7, 11] that learn a grammar from a set of example strings, and DTD/XSD inference approaches [1, 6] that try to infer a DTD from a collection of XML documents. Similar to *LSOD*, these techniques require only positive examples. But whereas their main goal is to



(a) The average rule size as a function of the minimum bigram frequency.

(b) The total of rules as a function of the minimum bigram frequency.

(c) Recall, precision and F-measure as a function of the minimum bigram frequency.

Figure 5: Bibliographic references and fragmentation

build regular expressions to fully describe the input sentences, we are not interested in a full representation of an object, instead we aim to derive rules that capture strong patterns in these sentences to perform classification. Last, but not least, an important feature of our approach that is not present in previous pattern learning approaches is the use of online learning to improve rule quality.

Lerman et al. [11] proposed DataProg, an algorithm that learns content-based rules for identifying structural patterns in data fields, and apply it to the problem of wrapper maintenance: using these rules, they can verify whether a wrapper is broken. Although DataProg also uses a word-based representation and captures statistically significant patterns, it does not consider patterns that occur deep in a sentence. As we discussed in Section 5, the ability to capture deep patterns is crucial for domains which contain fragmented objects. Moreover, since their primary goal is wrapper verification, in their experiments, Lerman et al. only applied the learned rules to pages that follow the same template (and in many cases, that contain the same data). Because DataProg learns patterns for individual fields of an object and it assumes that the different fields can be distinguished, it is unlikely the derived rules can be effectively applied to complex objects whose structure varies from site to site.

7. CONCLUSION

In this paper, we presented *LSOD*, a new approach for identifying Web objects that have a latent-structure. By taking only positive examples into account, *LSOD* identifies statistically significant patterns and uses these to derive object detection rules. *LSOD* automatically improves the derived rules through online learning: it extracts candidate sentences which are close to sentences identified by the initial rules and uses these as additional (positive) examples. Our experimental evaluation, using a significant number of pages in different domains, shows that *LSOD* outperforms state-of-the-art text classification techniques. There are several directions we intend to pursue in future work. While our rule derivation process works regardless of a specific domain and does not rely on pre-defined types, the availability of dictionaries and descriptions of common types (e.g., people's names, phone numbers, dates) is likely to improve rule accuracy. We plan to extend our approach to leverage this information when available.

Acknowledgments. Our research has been funded by the National Science Foundation grants IIS-0905385, IIS-0844546, IIS-0746500, CNS-0751152, IIS-0713637, the Department of Energy, and an IBM Faculty Award.

8. REFERENCES

[1] G. J. Bex, W. Gelade, F. Neven, and S. Vansummeren. Learning deterministic regular expressions for the inference of schemas from xml data. In *WWW*, pages 825–834, 2008.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[3] K. C.-C. Chang, B. He, and Z. Zhang. Toward Large-Scale Integration: Building a MetaQuerier over Databases on the Web. In *CIDR*, pages 44–55, 2005.

[4] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5):731–779, 2004.

[5] P. DeRose, W. Shen, F. Chen, Y. Lee, D. Burdick, A. Doan, and R. Ramakrishnan. Dblife: A community information management platform for the database research community (demo). In *CIDR*, pages 169–172, 2007.

[6] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: learning document type descriptors from XML document collections. *Data mining and knowledge discovery*, 7(1):23–56, 2003.

[7] T. Goan, N. Benson, and O. Etzioni. A grammar inference algorithm for the World Wide Web. In *AAAI Spring Symposium on Machine Learning in Information Access*, 1996.

[8] Google Base. `http://base.google.com/`.

[9] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *ECML*, pages 137–142, 1998.

[10] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, 2002.

[11] K. Lerman, S. Minton, and C. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:149–181, 2003.

[12] B. Liu. *Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2007.

[13] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[14] L. M. Manevitz, M. Yousef, N. Cristianini, J. Shawe-taylor, and B. Williamson. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[15] Z. Nie, J.-R. Wen, and W.-Y. Ma. Object-level vertical search. In *CIDR*, pages 235–246, 2007.

[16] A. Papoulis. *Probability & Statistics*. Prentice Hall, 1990.

[17] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.