

# Web based Named Entity Recognition

Dissertation

submitted in partial fulfillment of the requirements  
for the degree of

**Master of Technology**

by

**Sandesh Tawari**

(Roll no. 04329014)

under the guidance of

**Prof. Sunita Sarawagi**



Kanwal Rekhi School of Information Technology

Indian Institute of Technology Bombay

2006



# Dissertation Approval Sheet

This is to certify that the dissertation entitled  
**Web based Named Entity Recognition**

by

**Sandesh Tawari**

(Roll no. 04329014)

is approved for the degree of **Master of Technology**.

---

Prof. Sunita Sarawagi

(Supervisor)

---

Prof. Soumen Chakrabarti

(Internal Examiner)

---

Prof. S. Sudarshan

(Additional Internal Examiner)

---

Prof. Milind Sohoni

(Chairperson)

Date: \_\_\_\_\_

Place: \_\_\_\_\_



# INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

## CERTIFICATE OF COURSE WORK

This is to certify that **Mr. Sandesh Tawari** was admitted to the candidacy of the M.Tech. Degree and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
<b>Semester 1 (Jul – Nov 2004)</b>			
1.	HSS699	Communication and Presentation Skills (P/NP)	4
2.	IT603	Data Base Management Systems	6
3.	IT611	Object Oriented Techniques - Part I & II	6
4.	IT619	IT Foundation Laboratory	10
5.	IT623	Foundation of IT - Part II	6
6.	IT694	Seminar	4
<b>Semester 2 (Jan – Apr 2005)</b>			
7.	HSS700	Applied Economics	6
8.	CS610	Hypertext retrieval and mining	6
9.	IT608	Data Mining and Data Warehousing	6
10.	IT630	Principles and Practice of Distibuted computing	6
11.	IT680	Systems Laboratory	6
<b>Semester 3 (Jul – Nov 2005)</b>			
12.	CS601	Algorithms & Complexity	6
<b>M.Tech. Project</b>			
13.	IT696	M.Tech. Project Stage - I (Jul 2005)	18
14.	IT697	M.Tech. Project Stage - II (Jan 2006)	30
15.	IT698	M.Tech. Project Stage - III (Jul 2006)	42

I.I.T. Bombay

Dy. Registrar(Academic)

Dated:



# Abstract

Extracting structured entities from unstructured data sources is an important operation in many applications, including data warehousing and web data integration. In this thesis, we consider the problem of Named Entity Recognition (NER) from webpages. The scale, unstructuredness, and diversity of the web pose challenges to NER on the webpages. Traditionally, rule based techniques like Wrapper Induction Systems have been used for this task but these techniques are domain specific and not robust. We intend to use statistical learning based approaches.

Webpages are a rich source of multi-fielded entities such as product records, contact information and profile information. Such entities have complex internal structure and often there are spatial relationships between parts of an entity. Webpages are typically organized in terms of blocks which have hierarchical, adjacency and other forms of relationships. Capturing contextual interactions between blocks is necessary to extract complex entities. Unlike common sequence tagging tasks, the notion of adjacency is not precise in case of blocks on a webpage, and capturing this adjacency or neighbourhood relationship between blocks is a challenging task. The rich HTML structure that encloses the web content provides strong visual and spatial cues, in addition to textual information.

In this project, our aim is to build a framework that will assist in entity extraction from webpages by exploiting textual, visual and spatial properties. We concentrate mostly on entities composed of several sub-entities that are dispersed on a webpage. We use state-of-the-art statistical learners, CRFs and SVMs with a rich and varied set of features to perform the extraction task. We have experimented with both independent and collective labeling of blocks within a webpage. We report experiments on a real-life product extraction problem and analyze extraction accuracy under varying factors including training objectives, uniqueness constraints during inference and feature combinations.



# Contents

<b>Abstract</b>	<b>9</b>
<b>List of figures</b>	<b>15</b>
<b>List of tables</b>	<b>17</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Information Extraction from Web-pages . . . . .	4
1.2 Existing Approaches . . . . .	5
1.3 Organization of the Thesis . . . . .	5
<b>2 Literature Survey</b>	<b>7</b>
2.1 Machine Learning Background . . . . .	7
2.1.1 Conditional Random Fields (CRFs) . . . . .	8
2.1.2 Support Vector Machines (SVMs) . . . . .	11
2.2 Related Work . . . . .	12
2.2.1 Wrapper Induction Techniques . . . . .	12
2.2.2 Vision based Page Segmentation (VIPS) . . . . .	14
2.2.3 Visual Layout Driven Information Extraction from Websites . . . . .	15
2.2.4 2D-Conditional Random Fields . . . . .	16
2.2.5 Discriminative Random Fields . . . . .	17
2.2.6 MAP Estimation in MRFs via Rank Aggregation [1] . . . . .	18
<b>3 Information Extraction from Webpages</b>	<b>21</b>
3.1 Problem Definition . . . . .	21
3.2 Issues, Challenges, and Assumptions . . . . .	22
3.2.1 Noise in Web-pages . . . . .	22

3.2.2	Unit of Labeling . . . . .	23
3.2.3	Capturing Neighbourhood . . . . .	23
3.2.4	Entity Structure . . . . .	25
<b>4</b>	<b>Our Approach</b>	<b>27</b>
4.1	System Overview . . . . .	28
4.2	Obtaining Training Data . . . . .	29
4.3	Obtaining and Cleaning Formatting Information . . . . .	30
4.4	Filtering Stage . . . . .	31
4.5	Spatial and Similarity Layers . . . . .	32
4.6	Independent Labeling of Blocks . . . . .	34
4.6.1	Features . . . . .	34
4.6.2	Information Extraction Models . . . . .	38
4.6.3	Best Label Selection . . . . .	39
4.7	Collective Labeling of Blocks . . . . .	39
4.7.1	Features . . . . .	39
4.7.2	Information Extraction Models . . . . .	42
<b>5</b>	<b>Experiments and Analysis</b>	<b>43</b>
5.1	Experiments with Independent Labeling Approach . . . . .	43
5.1.1	Effects of Different Feature Sets . . . . .	43
5.1.2	Effect of Best Label Selection . . . . .	46
5.1.3	Effect of Dataset Size . . . . .	47
5.2	Experiments with Collective Labeling Approach . . . . .	48
5.2.1	Effects of Different Feature Sets . . . . .	50
5.2.2	Comparison of Gradient Descent and Collin’s Trainers . . . . .	51
5.2.3	Comparison of Constrained and Unconstrained Max-product for Collin’s Trainer . . . . .	52
5.2.4	Effect of Varying Beamsize on Collin’s Top-k Trainer . . . . .	52
<b>6</b>	<b>Conclusion and Future Work</b>	<b>55</b>
	<b>Bibliography</b>	<b>59</b>

**Acknowledgements**

**63**



# List of Figures

3.1	Skeleton of a web-page in terms of blocks and separators. . . . .	22
	(a) Original page . . . . .	22
	(b) Skeleton of page . . . . .	22
3.2	A web-page and its skeleton where the <i>title</i> entity only a part of block. . .	24
	(a) Original page . . . . .	24
	(b) Skeleton of page . . . . .	24
3.3	A web-page and its skeleton where the <i>title</i> entity is split across three blocks.	24
	(a) Original page . . . . .	24
	(b) Skeleton of page . . . . .	24
3.4	Web-pages demonstrating ambiguity in capturing concept of neighbour- hood for the entity <i>price</i> . . . . .	24
	(a) Neighbour of <i>price</i> in left . . . . .	24
	(b) Neighbour of <i>price</i> at top . . . . .	24
	(c) Neighbourhood ambiguous for <i>price</i> . . . . .	24
4.1	Flow diagram depicting main components and flow during training and deployment. . . . .	28
5.1	Effect of best label selection for Product information extraction task. . . .	46
5.2	Effect of training dataset size with CRFs. . . . .	47
	(a) Product name . . . . .	47
	(b) Product price . . . . .	47
	(c) Product image . . . . .	47
5.3	Effect of training dataset size with SVM (Polynomial kernel of degree 3). .	47
	(a) Product name . . . . .	47

(b)	Product price . . . . .	47
(c)	Product image . . . . .	47
5.4	Effect of training dataset size with SVM (RBF kernel). . . . .	48
(a)	Product name . . . . .	48
(b)	Product price . . . . .	48
(c)	Product image . . . . .	48

# List of Tables

5.1	Feature sets used for Independent Labeling experiments. . . . .	44
5.2	F1 values for various feature sets and models without best label selection. .	45
5.3	F1 values for various feature sets and models with best label selection enabled.	45
5.4	Feature sets used for Collective Labeling experiments. . . . .	49
5.5	F1 values for various feature sets with Collective Labeling. . . . .	50
5.6	Gradient Descent trainer vs. Collin’s trainer for Collective Labeling. . . . .	51
5.7	Unconstrained vs. Constrained max-product based training for Collin’s trainer for Collective Labeling. Each cell shows F1 score average (standard deviation). . . . .	52
5.8	Effect of varying beamsize for Collin’s top-k trainer for Collective Labeling. Each cell shows F1 score average (standard deviation). . . . .	53



1



# Chapter 1

## Introduction

The web is a huge repository of information, and continues to grow at a rapid pace, constantly being augmented and maintained by millions of people. The scale, unstructuredness and diversity of the web makes manual searching unfeasible. This has led to a lot of interest in techniques for automatic extraction of information from web-pages. The main idea is to extract structured entities from web pages which are unstructured in nature. Examples of such tasks are job-advertisement extraction, product information extraction, and the like. While a part of this information is in the form of machine generated HTML pages following some structure, a larger part is in the form of HTML pages generated in an ad-hoc manner. Many techniques have been devised to extract the information from the former part of the web. It is the latter part, i.e the unstructured web, that still poses a challenge. The part of the web in unstructured format, is very rich in terms of the information contained and cannot be neglected. A technique, such as ours, would play a prime role in many applications that need to extract structured entities from web-pages. Examples of such applications are

- Data integration systems that populate a structured database from a heterogeneous set of sources like web-pages, free text documents, reports etc.. Examples of such systems are a Product Information Server which extracts records describing products from websites of different vendors and provides an aggregated view of them, and a Job Matcher which extracts job descriptions from websites of different employers and profile details from resumes or personal pages of people, and does a matching of job providers with job seekers.
- Profile management systems that can automatically extract contact information from the visited pages and segment them, and populate an address-book. A similar

application is a bibliography management tool that comes integrated with browsers and has the ability to detect that the page being downloaded is an article or a paper, and then automatically extracts bibtex entry and prompts user for saving it into a bibliography server.

- A parallel work being carried out by author's colleague is about automatically identifying regularity in domain and leveraging the regularity to improve the global model for that domain. One of the inputs that their system would need is a set of web pages with labeled entities and our technique can be used for the same.

## 1.1 Information Extraction from Web-pages

Information Extraction (IE) aims to populate a database with information extracted from unstructured sources. Named Entity Recognition (NER) is an IE task which aims to locate mentions of named or pre-specified entities, e.g. person names, locations, and organizations, in freely formatted text sources. In our case the source of information are web-pages.

Traditionally, NER tasks over text documents have dealt with extraction of small entities such as person names, locations, and organizations. Such entities have very simple internal structure and it suffices to use only textual cues such as the lexical properties, a small context of words around, and match with dictionaries. Web-pages are rich sources of multi-fielded entities such as addresses, a publication records, product records, and contact information. We need more cues to capture the complex internal structures of such entities. A web-page with its HTML markups is rich in formatting information. Web-pages are typically organized in terms of blocks which have hierarchical, adjacency, and other forms of relationships. Semantically related parts of a web-page are laid out in a manner which depicts their relationship with each other. For instance, the heading of a block in a web-page gives information about semantic category of block. Such properties can provide vital evidences for analyzing a web-page and extracting complex entities from it. Consider the scenario in which a person is given a product information web-page in a language unknown to the person and he/she is asked to identify fields like product title, price, and image. The person would use cues such as title occurs at top of the page and appears distinctly, title occurs close to the product image, and price has digits in it and

occurs close to a submit button. An NER task over a web-page can additionally exploit visual and spatial properties of the text such as font, colour, position and associated headings. If these cues are captured through feature engineering then they can aid a computer program to do automatic extraction.

## 1.2 Existing Approaches

Traditionally, NER on web-pages has been done using manual approaches like hand-coded rules and semi-automatic approaches like Wrapper Induction[2]. Hand-coded rules are domain specific and not robust. Wrapper Induction based techniques assume that the web pages have some inherent structure and, therefore, suffice only for pages automatically generated from structured data source like a DBMS. Both these techniques have limited scope and do not generalize. We intend to use learning based approaches like Conditional Random Fields (CRFs)[3] and Support Vector Machines (SVMs)[4]. Such methods have statistical foundations and generalize well.

Statistical models like CRFs and SVMs identify entities based on their features. Therefore it is essential that we use right set of features. In this project, our aim is to build a framework that would allow to capture various aspects of a web-page cleanly. Entity extraction systems can then be written over such an enabling framework.

## 1.3 Organization of the Thesis

The thesis is organized as follows. We present our literature survey and related work in Chapter 2. In Chapter 3, we present a formal description of the problem, and describe challenges, issues and assumptions. In Chapter 4, we describe our approach towards solving the problem and the details of the currently implemented system. Experimental results and analysis is described in Chapter 5. Chapter 6 highlights our conclusions. Finally, we conclude by presenting ideas about our future efforts.



# Chapter 2

## Literature Survey

In this chapter, we present the essential machine learning background needed for rest of thesis in Section 2.1, followed by a discussion on works related closely to our work in Section 2.2.

### 2.1 Machine Learning Background

Traditionally, NER tasks have been modeled as a sequence tagging problem, where the aim is to come up with a label sequence for the given observation sequence. In our case the, observations to be labeled (blocks) have a 2D geometry and we have modeled the problem as a general graph tagging problem rather sequence tagging. We have used state of the art probabilistic graphical models - Conditional Random Fields (CRFs) - for graph tagging.

Earlier works have used SVMs and probabilistic graphical models such as HMM[5], MEMM[6][7][8], and CRF[3][9] and variants[10] for NER tasks. We discuss SVMs in more detail in Section 2.1.2. The graphical model based approaches can be categorized as generative and discriminative approaches. Generative approaches model joint probability distribution of labels and observations. A disadvantage of these models is that they have to enumerate all possible observation sequences. For tractability they make strong independence assumption and this inhibits use of overlapping features and long range data dependencies. HMMs and MRFs are classical example of generative models. Due to their limited capabilities generative models are often outperformed by discriminative models. Discriminative models model conditional probability distribution of labels given the observations, and hence they are also called as Conditional Models. They allow overlapping and non-exhaustive features and arbitrary data dependencies, and hence are

more expressive. MEMMs and CRFs are examples of conditional models. MEMMs are often outperformed by CRFs because MEMMs suffer from *label bias* problem. CRFs are more relevant to our work and we discuss them in more detail.

### 2.1.1 Conditional Random Fields (CRFs)

CRFs[3][9] are probabilistic framework for labeling and segmenting data. CRFs are a form of Undirected Graphical Models and they model conditional probability distribution  $Pr(\mathbf{Y}|\mathbf{X})$ , where  $\mathbf{X}$  is vector of random variables representing the input observation to be labeled and  $\mathbf{Y}$  is a vector of random variables representing labeling of  $\mathbf{X}$ . The structure of  $\mathbf{X}$  can be any general graph. CRFs are feature based models which are defined in terms of feature vector  $\mathbf{F}$  and weight vector  $\mathbf{W}$ . A feature is a property of  $\mathbf{X}$  which can take any real value. CRFs belong to the exponential family of distributions and they define a log-linear distribution given as

$$Pr(\mathbf{Y}|\mathbf{X}, \mathbf{W}) = \frac{1}{Z(\mathbf{X})} \exp(\mathbf{W} \cdot \mathbf{F}(\mathbf{Y}, \mathbf{X})).$$

Here the feature vector  $\mathbf{F}(\mathbf{y}, \mathbf{x})$  forms the sufficient statistics of the distribution and  $\mathbf{W}$  is the parameter vector. The normalization factor  $Z$ , also called partition function, is given by,

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp(\mathbf{W} \cdot \mathbf{F}(\mathbf{y}', \mathbf{x}))$$

CRFs being graphical models, the probability distribution modeled by them factorizes over the underlying graph  $G$  and so the feature vector  $\mathbf{F}(\mathbf{y}, \mathbf{x})$  can be decomposed into functions defined over maximal cliques  $\{c_1, c_2, \dots, c_n\}$  of  $G$ :  $\mathbf{F}(\mathbf{y}, \mathbf{x}) = (\{\mathbf{f}_c(\mathbf{y}_c, \mathbf{x})\})$ , where  $c$  indexes the maximal cliques, and  $\mathbf{y}_c$  represents the labeling for clique  $c$ . Now in this scenario there has to be a parameter,  $W_{f_c}$  for each clique and for each over that clique. To reduce the number of parameters to be trained, the parameters are shared across cliques. Thus we have

$$\mathbf{F}(\mathbf{y}, \mathbf{x}) = \sum_c \mathbf{f}_c(\mathbf{y}_c, \mathbf{x}).$$

The most probable labeling, also called Maximum A Priori (MAP) assignment, for input observation  $\mathbf{x}$  is given by

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} Pr(\mathbf{y}|\mathbf{x}, \mathbf{W}) = \arg \max_{\mathbf{y}} \mathbf{W} \cdot \mathbf{F}(\mathbf{y}, \mathbf{x}) \quad (2.1)$$

The MAP assignment can be found by running the message passing algorithm. The complexity of message passing algorithm is  $O(|\mathcal{Y}|^w)$  where  $w$  is the treewidth<sup>1</sup> of the triangulated graph. In case of chain graphs, the best label sequence  $\mathbf{y}$  can be found using Viterbi algorithm.

Many training methods, including Generalized Iterative Scaling (GIS), LBFGS based Gradient Descent, and Collin's Voted Perceptron, have been used to train the CRF parameters,  $\mathbf{W}$ . We next present an overview of Gradient Descent and Collin's Voted Perceptron approaches for training CRFs.

### Gradient Descent Trainer

The gradient descent approach moves in a direction which maximizes the likelihood of training data. Let  $T = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^K$  be the training dataset. The objective function to be optimized is the log-likelihood of training data

$$\begin{aligned} L(\mathbf{W}) &= \sum_k \log Pr(\mathbf{y}_k | \mathbf{x}_k, \mathbf{W}) \\ &= \sum_k (\mathbf{W} \cdot \mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - \log Z(\mathbf{x}_k)) \end{aligned}$$

To avoid overfitting, a penalty term is also added to the above objective function, but for simplicity we ignore it here. To optimize, we equate the derivative of  $L(\mathbf{W})$  w.r.t  $\mathbf{W}$  to 0.

$$\nabla L(\mathbf{W}) = \sum_k (\mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - E_{Pr(\mathbf{Y}|\mathbf{x}_k, \mathbf{W})} \mathbf{F}(\mathbf{Y}, \mathbf{x}_k)) \quad (2.2)$$

The Equation 2.2 has two terms - the first term is a count of number of times a feature fires on training data and the second term is the expected value of feature as predicted by the CRF model using the current weight vector. At optimality, these two terms must be equal, i.e. the gradient must be zero. If the gradient is non-zero, then the weight vector  $\mathbf{W}$  is changed so as to minimize the gradient. An external optimizer routine such as L-BFGS can be used to obtain updated  $\mathbf{W}$  by giving it as input the current  $\mathbf{W}$  and gradient. The expected feature value can be computed by first calling message passing algorithm and then using the clique probabilities to obtain feature expectation. To emphasize again, the complexity of message passing algorithm is  $O(|\mathcal{Y}|^w)$  where  $w$  is the treewidth of the triangulated graph, and the algorithm can be very costly if the graph is highly connected and has lot of cycles.

<sup>1</sup>treewidth of a graph is defined as size of maximal clique - 1.

### Collin's Voted Perceptron Trainer

The Voted perceptron method tries to minimize the differences between the global feature vector for a training instance and the same feature vector for the best-scoring labeling of that instance obtained using the current model. For each training instance, the global feature vector is first computed. Then for each feature  $f_k$  that supports the labeling of current model, the corresponding weight component  $W_k$  is decremented by that feature's value. Next for each feature  $f_k$  that supports the actual labeling ( $\mathbf{y}_k$ ), the corresponding weight component  $W_k$  is incremented by that feature's value. Thus we have

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - \mathbf{F}(\hat{\mathbf{y}}_k, \mathbf{x}_k),$$

where  $\hat{\mathbf{y}}_k$  is the best-scoring label returned by current model.

In above algorithm the feature votes for best-scoring labeling of model are given a weight of 1. Instead of using just the best-scoring labeling of the model, the top-N best-scoring labelings of model can be used to update the weight vector. In this case, the update rule becomes

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - \frac{1}{N} \sum_{i=1}^N \mathbf{F}(\hat{\mathbf{y}}_k^i, \mathbf{x}_k),$$

where  $\hat{\mathbf{y}}_k^1, \hat{\mathbf{y}}_k^2, \dots, \hat{\mathbf{y}}_k^N$  are the top-N best-scoring labelings returned by current model.

The algorithm repeatedly makes passes over the training data, updating the weight vector for each training instance. In each pass it accumulates the errors over all instances. When algorithm seems to be making no improvement in terms of reduction in total error at end of a pass, the algorithm terminates. The final weight vector is taken to be the average of all the  $\mathbf{W}(t)$ .

One of the advantages of Collin's trainer over gradient descent trainer is that unlike the gradient descent trainer, which needs sum-product based inferencing, the Collin's trainer needs max-product based inferencing. In many applications, the graph may be highly connected and may contain cycles. The sum-product algorithm is guaranteed to converge only on acyclic graphs and may not converge to an optimal on arbitrary graphs. Whereas, the max-product algorithm is also guaranteed to converge on graphs with atmost a single loop.

### 2.1.2 Support Vector Machines (SVMs)

Support Vector Machines (SVM)[4] are discriminative supervised learning algorithms that have been used in classification and regression settings. The power of SVM lies in their ability to do complex non-linear classification, by using the kernel trick, while still providing strong generalization bounds due to the max-margin property. We first introduce some terminology and then state the optimization problem which SVMs solve.

Let the training data be  $\{(\mathbf{x}_k, y_k)\}_{k=1}^K$ , where  $y_k \in \{-1, 1\}$  and  $\mathbf{x}_k$  are data points in some euclidean space. Initially assume the training points to be linearly separable. SVMs choose a hyperplane that separates the data points neatly with maximum margin - the distance between the closest data points of both classes and the hyperplane. Such a hyperplane is called max-margin hyperplane and the points closest to the hyperplane are called support vectors. The hyperplane found by SVM is of form

$$\mathbf{w} \cdot \mathbf{x} - b = 0.$$

SVM finds two parallel hyperplanes, parallel to the max-margin hyperplane, and closest to the support vectors in either class. These parallel hyperplanes can be described by equations

$$\mathbf{w} \cdot \mathbf{x} - b = +1,$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

In order to maximize the margin, the distance of these hyperplanes has to be maximized and it must be ensured that there no data points between them. The distance between the hyperplanes is  $2/|\mathbf{w}|$ , so we want to minimize  $|\mathbf{w}|$ . To exclude data points, following constraints need to be satisfied by each data point

$$y_k(\mathbf{w} \cdot \mathbf{x}_k - b) \geq 1 \quad 1 \leq i \leq n.$$

The problem now is to minimize  $|\mathbf{w}|$  subject to the above constraint. This is a quadratic programming (QP) optimization problem and the above formulation is called Primal QP.

After the SVM has been trained, it can be used to classify unseen test data points. This is achieved by using the following decision rule;

$$\hat{c} = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \end{cases} \quad (2.3)$$

Writing the classification rule in its dual form reveals that classification is only a function of the Support vectors, i.e. the training data that lie on the margin. Also in the dual form the data points are never accessed directly but in form of a scalar quantity expressed as dot product between a pair of data points. This allows use of complicated kernel functions. Kernel functions essentially project the original data points to some higher dimensional space, take the dot product in that space and return a scalar value indicating closeness of the two data points. Examples of kernel functions include Polynomial kernels of various degrees, Radial Basis Function (RBF) kernels, and Sigmoid kernels. The kernel function trick enables SVM to find non-linear classifiers by first projecting the data points into a higher dimension space and finding a linear classifier (hyperplane) in that space. A detailed tutorial on SVMs can be found in [4].

Extensions of basic SVMs have been suggested for multi-label multi-class settings in [11][12]. The main hurdle with their approach is that the number of constraints in primal QP and hence the number of variables in dual QP formulation are exponential in number of labels. This makes computation intractable for tasks involving large number of labels. Recently Max-margin Markov Networks (*M<sup>3</sup>Net*) were introduced in [13]. *M<sup>3</sup>Net* uses the structure of graphical model to factorize the original dual variables to marginal dual variables defined over the graph cliques. This reduces the number of dual variables to a polynomial order.

## 2.2 Related Work

### 2.2.1 Wrapper Induction Techniques

Wrappers are patterns that are meant to extract tuples of information from a target web page. The web page is assumed to have some structure and sources of such pages are structured source such as DBMS. Wrappers can, therefore, be thought of as doing reverse engineering to extract the structured entities from semi-structured web pages. Automatically constructing the wrapper by learning the patterns (or set of patterns) from a set of examples tuples in a web page is called as Wrapper Induction. A lot of

work has been done in this domain. In this section, we describe some of these techniques, including HLRT Wrappers[2], Lixto[14] and Thresher[15].

Some of the earliest work in this domain appears in [2]. They propose HLRT (*head-left-right-tail*) wrappers which assume that each information tuple consists of attributes separated by special strings (say, `</TD>` or `<B>`). Further, they assume that the end of the header and the start of the tail (where header and tail are boundaries of set of tuples within a source) are marked by strings *head* and *tail*, respectively. For a tuple with  $K$  attributes (e.g.,  $K=2$  and tuple as `<TR><TD> Sandesh </TD><TD> Re.5000 </TR><TD>`) the HLRT wrapper would be a vector of  $2K+2$  strings (say, `(<TABLE>,<TD>,</TD>,<TD>,</TD>,</TABLE>)`). They rely on an Oracle to provide them labeled examples. The Oracle is not assumed to be perfect, and is composed of recognizers. A recognizer finds instances of a particular attribute on a page. The authors present an algorithm that allow to compose heuristic Oracles by corroborating recognizers. They also propose an algorithm for constructing HLRT vectors, given the examples. They provide an analysis to bound the number of examples needed to generate a satisfactory wrapper. They do not specify who/what an Oracle could be, but their experiments suggest that a recognizer could be a human who marks instances of a particular attribute as positive.

Lixto[14] is an interactive visual system for wrapper induction and it uses a variant of the technique presented in [2]. Lixto allows user to mark the examples and provide feedback (This serves the purpose of an Oracle). From the learned examples they learn an Elog grammar. Elog can be thought of as a declarative grammar that consists of set of XPath like expressions over the HTML page. It serves the purpose of a HLRT vector[2]. Finally they use this grammar to extract tuples from HTML pages and convert them into an XML document.

Thresher[15] is another tool for interactive wrapper building and it comes integrated with Haystack browser<sup>2</sup>. Thresher allows user to interactively mark examples in a web page displayed in the browser. Their system then aligns pairs of examples, by matching up their DOM subtrees, using *TreeEditDistance*. An example pattern that Thresher can produce is `(<TD>*</TD><TD>*</TD>)`. Intuitively, the wildcard `*` denotes variable contents and other parts denote layout information. Upon forming a pattern the user is asked to select (or create) a RDF class for the pattern. Thus each pattern becomes a

---

<sup>2</sup>Haystack is a semantic browser with support for RDF.

semantic object. The browser, being semantic web aware, uses these patterns to annotate pages with RDF classes, and attempts to extract values for properties of the class from the DOM subtree.

Wrapper Induction based IE techniques suffice only for semi-structured pages. They are domain specific and do not generalize beyond pages similar to ones for which the wrappers were originally designed. Also, they depend on the DOM subtree and slight change in the structure makes the wrapper ineffective. Their advantage is that they are quite fast and could serve purpose of first level filtering of noise from the web pages.

### **2.2.2 Vision based Page Segmentation (VIPS)**

VIPS[16] is an iterative rule based approach that take as input the DOM hierarchy of an HTML page, and gives a semantic block hierarchy. The nodes of the hierarchy, called as blocks, occupy rectangular area on the browser screen, and represent semantically coherent information. VIPS is an iterative algorithm, and has three main steps: Identify blocks, Find separators, Merge blocks. Initially, at the first level, the only block is the entire page. For a block at a particular level, the algorithm uses the visual and spatial properties of the block, such as font and color, to find its children blocks. The algorithm fires a set of rules in priority order upon the blocks to decide between splitting or freezing them. Blocks may be separated by vertical or horizontal separators, and the width of the separator gives the extent of dissimilarity amongst adjoining blocks. In second step, the algorithm finds the separators between the children blocks. Finally, VIPS merges the children blocks, starting from the blocks with thin separator between them. Thus a hierarchy of blocks is obtained. VIPS associates a measure of coherence, Degree of Coherence (DoC), with each block. On those blocks, that violate a threshold Predefined DoC (PDoC), VIPS is further applied.

VIPS performs excellently over most of the pages, but it is quite sensitive to the visual properties such as font information and color. So, it is very important to pass them accurately to the algorithm. VIPS fails to segment a block that contains only free text and does not find its semantic hierarchy. This is a very important limitation because, as suggested in [17] and Section 2.2.3, often, it is the case that such blocks are of prime interest.

### 2.2.3 Visual Layout Driven Information Extraction from Websites

The author proposes a two step approach towards extracting entities from web pages[17]. The entities are long in nature and hence require a different approach as compared to NER for small entities. They have utilized visual/spatial properties of a web page. Their core idea is to reject most part of web pages that are not likely to contain the entity and apply their model to the set of potential blocks to extract the entity. As a part of the effort, they have implemented the VIPS (described in Section 2.2.2). Given the training data, i.e. set of web pages with entity of interest marked in them, they use the VIPS algorithm to obtain the block hierarchy. As a part of preprocessing step to the VIPS algorithm, they render the page in a custom build tool on Mozilla browser and annotate all the nodes of DOM tree with visual and spatial properties through the tool. The VIPS algorithm uses these properties to form the semantic hierarchy. Thus the labeled pages of the training data are converted into trees. They explored various ways of annotating such a tree from the original labeling in the corresponding page. The labeled tree is broken into subtrees using heuristic rules. These subtrees then serve as training data for the level 1 model. They have experimented with CRF, Semi-markov CRF, Undirected Graphical Models and Segment CRF for the level 1 model. The prime responsibility of this model is to select a frontier of potential nodes. The model must have a high recall so that the chances of a useful block being rejected are very less. They report that the Semi-markov CRF and Segment CRF are more direct approaches and give better results. Segment CRF was an optimization over the Semi-markov CRF in the sense that, Segment CRF drastically reduced the candidate segments that need to be generated. For each of these potential blocks, a set of candidate headings are found. These blocks then serve as input to the level 2 model. For the level 2 model, they used CRF. This model is expected to have high precision.

There are many important observations from this work. It was found that in extracting the long entities semi-markov features play a very vital role. It was observed that certain words prefer to occur at specific places within a entity and in general, distributions over their relative position of occurrence with respect to start and end boundaries(of the entity) can be found. They introduced *position* features to capture these distribution

discretely. They found that lot of computations was redundantly being performed. As an effort towards avoiding re-computation, they implemented feature caching for semi-markov features in Semi-markov CRF and Segment CRF. Due to 2 level approach, they found that error propagated from level 1 to level 2. Also they could not very well exploit the aspects such as associated headings and relationship with neighbouring blocks. They feel that a unified model could do away with both these drawbacks but then such a model would be computationally intensive.

### 2.2.4 2D-Conditional Random Fields

2D Conditional Random Fields (2D-CRF)[18] is an extension to the linear CRF[3] to capture neighbourhood interactions in 2D layout of a web-page. The authors also propose approaches for parameter learning and inferencing. They use 2D-CRF for entity extraction from web-pages. An *object block* is defined to be the part of a web-page representing information about the object (entity of interest). This information usually appears as grouped together. An *object element* is defined as an atomic part representing an extractable entity. Object blocks are extracted using existing techniques such as Web-page Segmentation. The object block is then indexed on a 2D grid. For each indexed object element a *real state* is introduced. For empty cells on the grid *null states* are used. Object elements spanning multiple cells are represented by one *real state* and multiple *virtual states*. During learning and inferencing a restriction is imposed that the virtual states and the real state of an object element must have same values when a transition takes place. For each object element left, right, top, and bottom neighbours were considered. These neighbourhood relationships form the edge-set of the underlying graph (represented by the 2D grid) of 2D-CRF. Since inferencing on a 2D grid is provably intractable, so they present a heuristic algorithm. In worst case, the algorithm is of exponential order.

To obtain neighbours, the only information they use is positioning on grid and size of the object element. They assume that spatial adjacency means actual neighbourhood. But they ignore other facts such as presence of separators, visual properties of blocks and distance between object elements. Also, the only notion of neighbourhood they consider is immediate spatial adjacency, but this may not always be the case.

### 2.2.5 Discriminative Random Fields

Discriminative Random Fields (DRF) [19] is a discriminative framework for the classification of image regions by capturing neighborhood interactions in the labels as well as the observed data. The authors have extended CRFs [3] for image labeling problems. Examples of such problems include image smoothing, man-made structure detection, parts-object detection, region demarking in images, image category classification, and so on.

Images exhibit contextual dependencies of varying level. There are neighbourhood interactions amongst parts of images at different granularities. There are pixel-pixel and patch-patch interactions, interactions between parts of same objects, and object-object, region-object and region-region integrations. For instance, adjoining pixels (or small sized image patches) tend to have similar visual properties and they tend to have same label except at edges. While trying to detect presence of a computer in a image, the position of monitor spatially constraints position of keyboard and vice-versa. In man-made structure (say buildings) detection there are interactions between sky and road, sky and building, building and road, building and car and so on. Being able to capture these dependencies is a necessary for labeling images which may have poor quality, noise, ambiguities due to camera angle, poor light etc. The authors suggest that the capturing the context (through neighbourhood interactions) can lead to better classification.

Prior to their work, Markov Random Fields (MRF) have been used because they provide a probabilistic framework to model contextual interactions. However, MRFs being generative in nature do not allow to arbitrary data dependencies. MRFs do not allow label interactions to be data dependent. Whereas image labeling problems need complete freedom in modeling arbitrary data interactions which may of short range or long range. To alleviate shortcomings of MRF, the authors propose DRF which extends the CRF proposed in [3]. The conditional framework allows DRF to capture arbitrary data dependencies and model data-dependent label interactions. They propose basic DRF (for both binary class and multi-class labeling problems) and hierarchical DRF.

They extend the basic 1D-CRF to more general settings wherein the underlying graphical structure (the random field of labels) is a 2D lattice or even a general graph. Further, they use arbitrary discriminative classifiers in the unary (Association potentials) and pair-wise (Interaction potentials) potentials. Association potential,  $A(\mathbf{x}, y_i)$ , at site  $i$  of image

( $textbf{x}$ ) is the confidence score for label  $y_i$  being assigned to site  $i$ . Interaction potential,  $I(\mathbf{x}, y_i, y_j)$ , models the interaction between label  $y_i$  of site  $i$  and label  $y_j$  of site  $j$ . Both these potentials are data-dependent. The basic DRF allows to capture only short range data-interactions such as pixel-level and patch-level interactions. The authors report use of basic DRF for parts-based object detection, image smoothening and man-made structure detection in images.

The basic DRF captures only short-range interactions. The authors propose hierarchical DRF that allows simultaneously encoding of both long-range and short-range data interactions. The model consists of two layers- layer 1 for short-range and layer 2 for long-range interactions. Both these layers are modeled as DRFs. There are directed links from layer 2 to layer 1 to make computation of partition function tractable. The underlying graphs of the two layers can be any arbitrary graphs and not necessarily lattices. The labels used in layer 1 are *pseudo*-labels and the layer 2 labels are the actual labels. The layer 2 nodes induce a partition on layer 1 nodes through a *partition function*. This partitioning does not necessarily correspond to image partitioning (segmentation). The conditional probability term  $P(y|x)$  involves a summation over all possible partitioning and all possible layer 1 labelings and, hence, it is NP-Hard to evaluate. The authors propose *Sequential Max-Likelihood Approximation* wherein they freeze layer 1 labels by training layer 1 till equilibrium is reached and then other parameters are learnt by maximizing the approximate conditional likelihood. The final labeling is obtained by *replication mapping*[19] of labels in layer 2.

### 2.2.6 MAP Estimation in MRFs via Rank Aggregation [1]

Maximum A Priori (MAP) estimation on MRFs with arbitrary underlying graph is a hard problem. The complexity of running max-product belief propagation on arbitrary graphs is exponential in treewidth of the triangulated graph. The Generalized Belief Propagation algorithm can find the optimal MAP assignment only for trees and graphs with single loop. However, in general case, the algorithm tends to recount the messages and may not converge, and even when it converges the final pseudo-marginals obtained may be far from the optimal. Wainwright introduced Tree Reweighted max-product algorithm (TRW) where the original graph is decomposed into spanning trees and the original potential is expressed as a convex combination of tree-structured potentials. A slightly modified form

of message passing is used to reach consensus amongst max-marginals of the tree edges. The algorithm finds an optimal MAP assignment only if all the trees agree on a MAP assignment. The algorithm returns an upper bound on the value of optimal if it fails to converge.

A recent work[1] enhances the TRW approach by improving the bounds provided by TRW algorithm and finding the optimal in many more cases. In their approach, the graph is decomposed into a set of spanning trees. Each edge of the original graph is present in atleast one of the trees. The original set of potentials are expressed as an additive combination of potentials defined over edges of trees. For instance, if an edge of the original graph is present in two different trees, the potentials of edge in those trees sum upto the original potential of the edge as defined in the graph. For each of the trees, they run max-product algorithm to obtain top-k MAP assignments. For each tree, an upper bound on scores of solutions not included in the top-k list is also found. They then use Fagin's merge to aggregate the results from different trees and get final list of top-k assignments. The new upper bound is computed by summing individual upper bounds of the trees. If the gap - difference between score of the final optimal solution and the final upper bound - is zero, they return the optimal solution and the upper bound. Otherwise they reparameterize the original potentials to reflect the max marginal probability, and then reiterate the entire procedure. They report use of three different reparameterization methods - edge-based, tree-based and sequential update algorithm.



# Chapter 3

## Information Extraction from Webpages

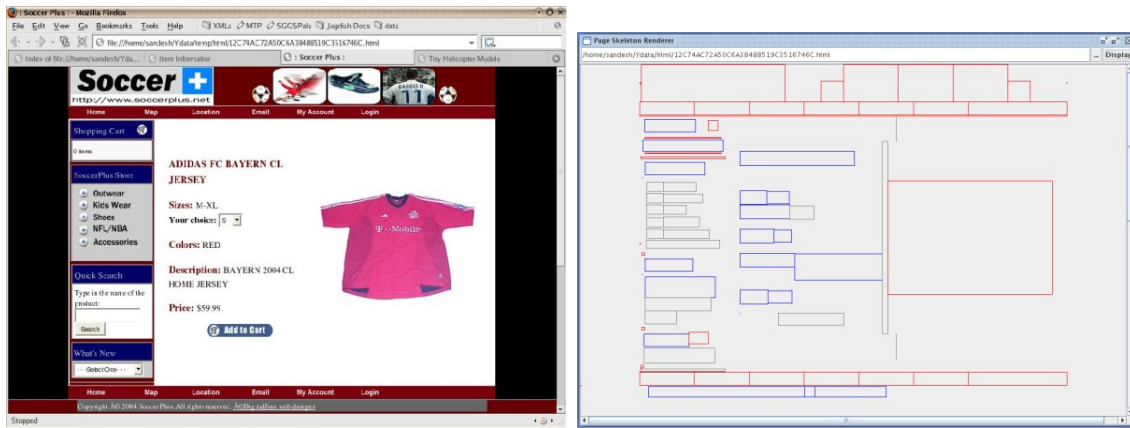
### 3.1 Problem Definition

We intend to build a framework that will assist in entity extraction from web-pages by exploiting textual properties (such as lexical properties, text in context and keywords), visual properties (such as font, color and background color), and spatial properties (such as vertical and horizontal positioning, and position with respect to other parts of the page). The entity to be extracted can be long or short and may appear contiguously or in form of a number of related sub-entities dispersed on a web-page.

#### Our Model of a Webpage

We consider a web-page to be made up of blocks and separators. A block is the smallest coherent unit of content on a web-page. For the purposes of this report, it can be assumed to be a leaf in the DOM tree representation of the web-page but in general a block could be of coarser or finer granularity than our definition. A separator is rectangular area separating two or more content bearing blocks. The blocks and separators do not have uniform sizes. Therefore, a web-page is essentially is 2D layout of irregularly placed blocks of varying sizes. In simplest case, the basic unit of extraction can be assumed to be these blocks, but, in general, the unit of extraction may not be aligned with block boundaries.

Figure 3.1 shows an example web-page and its skeleton where the blue rectangles are for text blocks, red ones for image blocks and gray ones for all other types of blocks. The whitespace between blocks represents separators.



(a) Original page

(b) Skeleton of page

Figure 3.1: Skeleton of a web-page in terms of blocks and separators.

## 3.2 Issues, Challenges, and Assumptions

The key challenges to NER on web-pages is the the diversity of pages. Further, the way HTML pages are built makes conventional techniques for NER ineffective on web-pages. In this section, we describe the main issues and key challenges, and our initial assumptions.

### 3.2.1 Noise in Web-pages

Due to the flexibility of HTML, it is very common to find pages that, when viewed on a browser appear to be the same, but have completely different HTML trees. To be able to capture relationships between blocks, it is natural to look out for a hierarchy amongst blocks. Blocks that, when rendered, appear to be spatially related (say, adjacent) may not be proximal in the DOM tree. Relationship (such as sibling) between the blocks, that is evident from the DOM tree, may not actually make any sense in the spatial layout. In short, the DOM tree can not necessarily give a good representation of the semantic hierarchy. This also hints towards the fact that it is the visual layout that helps in obtaining the semantic hierarchy. This visual layout can be captured by measuring the visual/spatial properties of a rendered page.

Web-pages generally contain a conglomeration of topics. In addition to the data of interest, a web-page contains a lot of other data which is noise as far as our task is

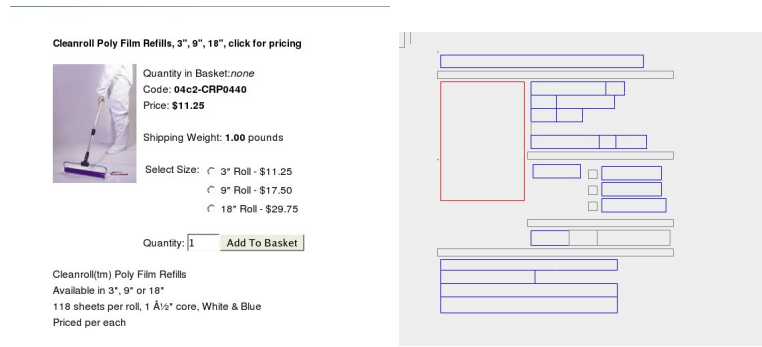
concerned. For e.g. the intended data may be in the central area and the other parts like footer, header and navigation bar may be less relevant. Devising techniques that are fast and accurate, even in the presence of noise, is a challenge. Many approaches exist that can segment a web-page into coherent blocks. Vision based Page Segmentation (VIPS)[16] and Web Page Segmentation[20] are two such approaches. Once such a segmentation is obtained, a simple classifier such as decision trees with rules such as number of links in the block, TFIDF or Jaccard match score with an dictionary of entities, can be applied to reject many of the blocks. We call the remaining part of the web-page as the *visual block* and this is the main content bearing part of the page we are interested in. To begin with, we have assumed that the *visual block* has been extracted and we start off from that point, but later we will integrate such a system into our framework.

### 3.2.2 Unit of Labeling

We have defined a block to be the smallest coherent unit of content. Considering the DOM tree representation of the web-page, the most natural representation of block is the leaf nodes. However, with this definition of a block, the (sub)entity boundaries may not necessarily align with block boundaries. Entities can span multiple blocks, occupy exactly one block, or be only a part of block. For instance, in web-pages describing consumer products, where we are interested in extracting sub-entities like product title and manufacturer-name, it is often the case that product title spans 2 blocks (see Figure 3.3 for an example) or occupies only part of a block (see Figure 3.2 for an example). The solution to this problem is to merge blocks or split blocks and do labeling at text-token level. In our initial attempts, we had assumed that entities occupy exactly one block, and now, we are exploring techniques to handle the general case.

### 3.2.3 Capturing Neighbourhood

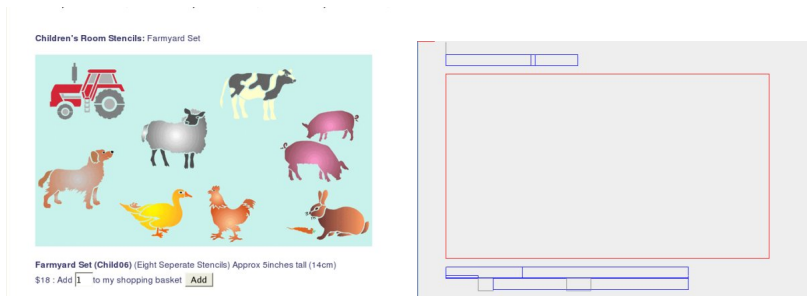
A web-page is made up of blocks and separators, and these can be of varying sizes. Given that the entities can span multiple blocks, we need to be able to merge blocks. To be able to merge blocks, it is necessary to identify neighbouring blocks. For instance, in Figure 3.3 we need to be able to merge the three blocks to label the *title* of the product correctly. Even when the neighbourhood blocks are not to be (or cannot be) merged, they



(a) Original page

(b) Skeleton of page

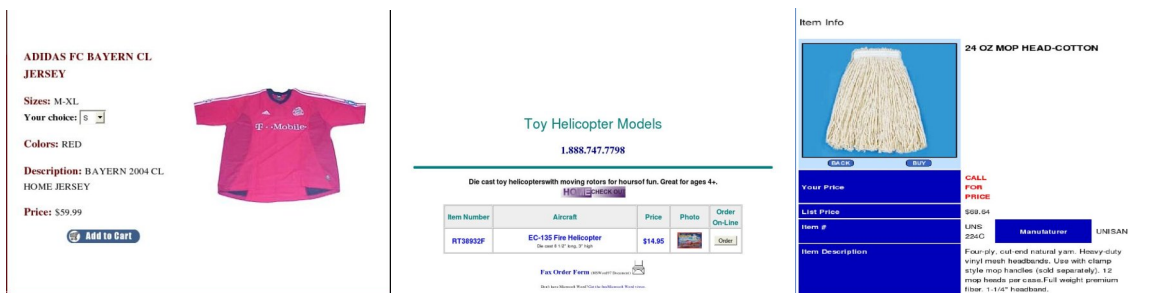
Figure 3.2: A web-page and its skeleton where the *title* entity only a part of block.



(a) Original page

(b) Skeleton of page

Figure 3.3: A web-page and its skeleton where the *title* entity is split across three blocks.



(a) Neighbour of *price* in left

(b) Neighbour of *price* at top

(c) Neighbourhood ambiguous for *price*

Figure 3.4: Web-pages demonstrating ambiguity in capturing concept of neighbourhood for the entity *price*.

still can provide evidences for better classification of each other. For instance, in a web-page describing contact information, if the neighbourhood block in left contains the text 'email:', the current block is likely to contain an email address. The neighbourhood block can be in any of the four directions - top, bottom, left or right. It often become ambiguous to establish adjacency of blocks unless we take into account a larger context. An example of inherent ambiguity in finding neighbourhood blocks can be seen in Figure 3.4. It is difficult to find neighbouring blocks of a given block programatically. For text within the block, the neighbourhood text is clearly defined, but for text near block boundaries, neighbourhood is uncertain.

### **3.2.4 Entity Structure**

Based on the nature of the web-page the effort needed in a NER task vary. Pages automatically generated from structured data source, like DBMS, are somewhat structured in nature. It is easier to extract entities from such pages. However, there is a large part of web consisting of HTML pages generated in an ad-hoc manner, either manually or with the help of tools. For the latter category of pages, long range dependencies exist between objects and a larger context information needs to be captured. Structure of the entity also determines efforts needed in the NER task. Small entities such as people name and location name are structural in nature. Textual properties are sufficient to extract such entities. Long and contiguously occurring entities such as address, publications and news-pieces not only need textual properties but also require context around to be captured. An entity's visual properties and its spatial location with respect to other entities becomes important. Entities can be composed of several sub-entities which are dispersed in a page. Task of extracting the entire entity becomes same as extracting sub-entities. These sub-entities may be tied up in a relationship and can affect labeling of each other. This last case represents the most general setting. In our initial attempts, we had ignored relationship between sub-entities and our current efforts are focused towards doing collective labeling of sub-entities.



# Chapter 4

## Our Approach

We intend to build a framework that will assist in entity extraction from web-pages by exploiting textual, visual and spatial properties. We concentrate mostly on entities composed of several sub-entities that are dispersed on a web-page. The work in [17] dealt with long contiguously appearing entities. We propose a multi-stage approach where, in the initial stage, we filter out irrelevant parts of pages using a cheap filtering stage. The latter stages apply increasingly costlier models. The initial stages have high recall and later stages have high precision. In our current attempt, we have built the system with following assumptions, but as a part of future work, we intend to incorporate more general settings.

- We have assumed that *visual block* has been extracted. A visual block is the main part of a web-page which contains the entity, and with irrelevant parts like header, footer and navigation-bar filtered out from the web-page. Instead of the entire web-page only its visual block is input to our system. We also assume that a visual block consists of only one entity i.e. sub-entities of a single entity only. When there are multiple entity records within a web-page, they usually are in some form of relationship and they can provide vital clues to the classifier and affect each other's labeling. Under our current settings, we do not exploit these clues.
- We have assumed that entities occupy exactly one *block*. As described in Section 3.1, a block is the smallest coherent unit of content on a web-page. In simplest case, it can be assumed to be a leaf in the DOM tree representation of the web-page. We are exploring techniques to handle the general case where entity boundaries need not align block boundaries and we present some of our ideas in Chapter 6.

To start with, we have build our system with an exemplary task, but most important parts of our system are generic in nature and domain independent. In the next few subsections we describe our system in detail.

## 4.1 System Overview

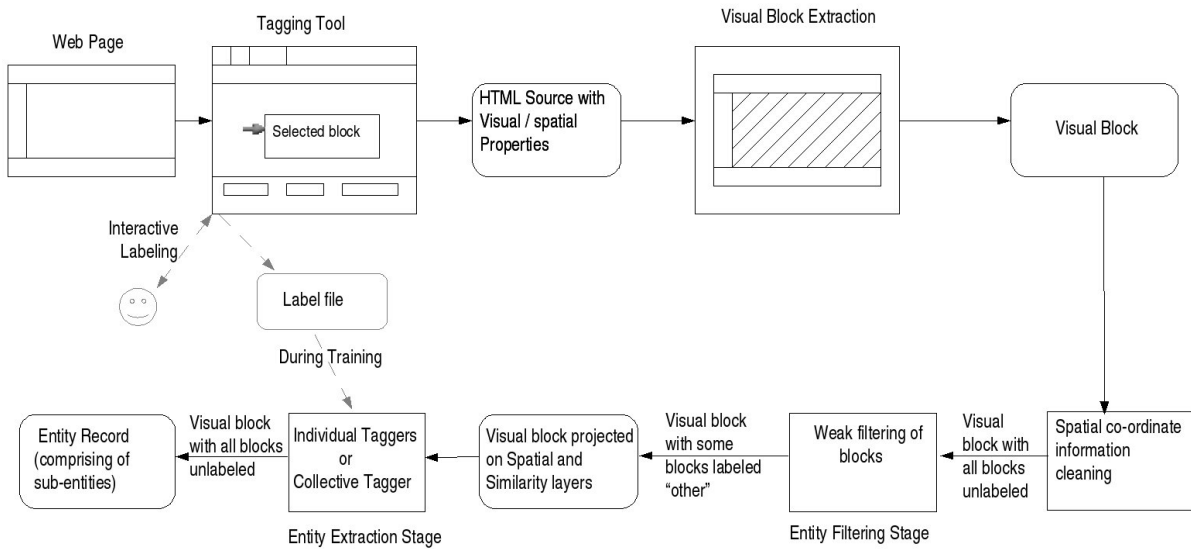


Figure 4.1: Flow diagram depicting main components and flow during training and deployment.

Figure 4.1 shows the the main components of the system and the flows during training and deployment. It gives an end-to-end view of the entire system.

Given a web-page, the tagging tool (described in Section 4.2) is used by the user to interactively tag the web-page. The tagging tool creates two output files: A label file and a visual information file. The label file contains the tagged information and the visual information file contains an XML version of the HTML source with visual, spatial and some other attributes. The tagging process and the file formats are described in more detail in Section 4.2. An external system is used to extract the *visual block* from the web-page represented by the visual file. Such systems have been described in VIPS[16] and Web Page Segmentation[20]. Such a system is complementary to our current system but later, we will integrate it into our system.

Once the visual block has been extracted, the main flow of our system begins. The

visual block is essentially a collection of DOM subtrees annotated with formatting information. The formatting information for spatial coordinates-left, right, top and bottom may not be completely correct and hence the visual block is passed through a series of filters to clean the spatial coordinates. These pre-processing steps are described in Section 4.3. Initially, all the blocks in the visual block are unlabeled. The corrected visual block is passed through Entity Filtering Stage (section 4.4). This stage serves as a weak filtering stage and its output is visual block with some of the blocks labeled as *other*<sup>1</sup>. The blocks labeled as *other* are not considered for labeling in the entity extraction stage. However, the properties of these blocks such their coordinates and textual content are used for labeling remaining blocks. The visual block is then projected on Spatial and Similarity layers (described in Section 4.5). Intuitively, these layers provide the essential data structure and higher level APIs needed for firing features in the Entity Extraction Stage. In the Entity Extraction Stage, we have used two alternative approaches for labeling the blocks - Independent Labeling of blocks (Section 4.6) and Collective Labeling of blocks (Section 4.7). During the training phase of these approaches, the label file and the visual block from the previous stage act as input to the learner which learns a statistical model. The training flow ends at this stage. During deployment phase, the only input to this stage is the visual block from the previous stage. The learnt model is applied over the unlabeled blocks and the output of the stage is an entity record comprising of constituent sub-entities.

## 4.2 Obtaining Training Data

We built our own tool for interactively tagging web-pages. The tagged information is stored in a custom built file format. The file format is generic enough to allow us to express training data for most of the web-IE tasks.

We use a simple XML file format to express the tagged information about a page. The file format allows us to express a page in terms of entities and global properties that apply to all the entities within the page. Each entity can be expressed as a collection of sub-entities and properties that apply to all its constituent sub-entities. The kind of properties that we focus on are heading, similar block, title and data (constituent text).

---

<sup>1</sup>*other* denotes label of non-entity blocks.

There is a special property, *data* that represents actual (sub)entity of interest. Sub-entities are represented by only *data* property. Properties can be viewed as a range of text within a block. Blocks correspond to the DOM tree nodes. Properties are represented by *nodeid* (preorder id), child-offset with respect to the parent node, and begin and end offsets of text within the DOM node.

We modified the tagging tool used in [17] to suit our requirement. It is an interactive tagging tool to tag web-pages for training data. The tool is written as an XUL application in javascript and XML, and is integrated with Mozilla browser. To tag a page it has to be rendered in our tool's browsing window. Tool provides GUI support to make text selections representing properties of (sub)entities. The tagged information can then be saved in the file format mentioned above. We call this file as *label file*. The tool can also output a XML file, *visual file*, corresponding to the DOM tree of source HTML annotated with attributes for *nodeid* (preorder id), child offset with respect to the parent node, visual properties such as font weight, font pixel size, color, background color and spatial coordinates-left, right, top and bottom.

Our system does not need the original HTML file and it works only based on information in the visual file. During training both visual and label files are needed, and during deployment on label file is needed. One of the limitation so method of obtaining training data is that, once the original HTML file is changed, we need to regenerate both the visual and the label files or else the system continues to use the old version of the web-page.

### 4.3 Obtaining and Cleaning Formatting Information

The formatting information including textual, visual, spatial and other attributes of blocks (DOM nodes) are obtained by rendering the web-page in Mozilla browser. The information used by us includes: tag name, text (if a text block), font pixel size, font weight, font family, bold status, italics status, link URL (if a hyperlink), alternate text (if a image), background and foreground colors, and spatial coordinates-left, right, top and bottom. But the values obtained are not completely reliable. Mozilla does not have a representation for text DOM nodes and it does not expose APIs to obtain visual/spatial information for these nodes. These are represented as `#text` elements in the DOM tree and are treated as free text. Since we cannot obtain required values for these nodes, so we use values of

their parent nodes. This a misleading approximation of the spatial coordinate values for text nodes. There are other issues such as wrong visual/spatial property values for tags such as `font`, `center` to name a few. These issues are discussed at length in [17].

To clean the incorrect spatial coordinates obtained for the text nodes and some other nodes like `font`, `center`, we use a number filters and hacks. Examples of filters used by us are: clipping bounding box of a node to fit within parent node’s bounding box, expanding bounding box of node to accommodate bounding boxes of its children and when the expansion does not violate limits imposed by its siblings and parent, and adjusting coordinates of a text node by looking at coordinates of its non-text sibling nodes. Another hack, which we found useful, was to enclose each text node with a `span` tag. The `span` tag is a non-text node and is assigned correct coordinates. This hack has to be applied while obtaining coordinates from mozilla. The other cleaning steps must be applied when the training/test data is loaded.

## 4.4 Filtering Stage

The main goal of this stage is to reject out some of the blocks by applying simple heuristics. This stage serves as a weak filtering stage which provides cheap filtering and ensures that costlier models of later stages have less work to do. This stage is used both during training as well deployment phase.

The input to this stage is a visual block. Simple domain dependent rules are applied over blocks in the visual block to decide whether a block is a potential (sub)entity or not. For instance, in case of product information extraction, the heuristics used are whether the length of text in the block exceeds threshold, whether the text in the block has special font properties like italics/bold or lexical patterns like all capitals/first letter capitalized, and whether the text in the block contains some digit. The output of this stage is a visual block with some of the blocks marked as non-entity. In later stages, these blocks are not labeled but the contents in these blocks could be still used to label other blocks.

It is necessary to ensure that this stage has high recall and it still does effective filtering. For the product information extraction task, we found that the simple rules used by us were effective in rejecting around 52% of the non-candidate blocks and gave us 99.11% recall over 288 product pages.

## 4.5 Spatial and Similarity Layers

The Spatial and Similarity Layers form the essential data structure and provide high level APIs needed by entity extraction stage. The model of web-page used in [17] was essentially the semantic hierarchy of a web-page formed by running VIPS algorithm [16] on it. In our case, we consider a web-page to be a flat 2D layout of blocks and separators of varying sizes, and the spatial and similarity layers are essentially a representation of this model. We divide the queries on this data structure into two sets - those that can be answered just by considering the physical layout and those that involve higher form of relationships such as buddy blocks and heading blocks. The spatial layer is meant for answering the former type of queries and the similarity layer for the latter. We had highlighted the importance of neighbourhood and difficulties in obtaining it in Section 3.2.3. In addition to the inherent ambiguity in defining adjacency, heading, and other relationships, computational complexity is another main concern.

### Spatial Layer

The spatial layer is raw layer that answers proximity queries and queries concerning layout properties of blocks. It captures the physical layout of the blocks and does not have formatting information. It internally uses a 2D grid data-structure to index the blocks. The spatial layer provides APIs such as

1. Given a block, return all the surrounding blocks that lie within a specified bounding box. The returned blocks may not necessarily be immediate neighbours of the source block.
2. Given a block  $b_{src}$ , return all the surrounding blocks  $b_{nbr}$  such that there are atmost  $numIntermediateNodes$  between  $b_{src}$  and  $b_{nbr}$ . For instance, if  $numIntermediateNodes$  is set to 0, then we are looking for only immediate neighbours, and if we set  $numIntermediateNodes$  to 1, then those blocks which are either immediate neighbours of  $b_{src}$  or within a neighbourhood distance of 1 from  $b_{src}$  are acceptable.
3. Given a block, return all separators associated with it. The set of separators associated with a block are rectangular areas (whitespaces) separating the block from its immediate neighbours.

4. Given a block  $b_{src}$  and some vertical/horizontal tolerance, return blocks  $b_{nbr}$  which have their top, bottom, right, or left aligned with  $b_{src}$ . We say that two blocks  $b_1$  and  $b_2$  are top-aligned if  $|b1.top - b2.top| \leq vertical\_tolerance$ .
5. Given two blocks, return the minimum and maximum distances between any two points on the periphery of blocks.

## Similarity Layer

The similarity layer essentially computes the similarities between blocks and can answer queries involving higher forms of relationships. This layer internally uses spatial layer for obtaining layout information and combines it with formatting and textual properties of blocks to answer the queries. To answer a query on a block, we pair it with all blocks within a set of potential result blocks and then for each pair we compute a score. To compute the score, we combine in different proportions, evidences such as spatial distance, top/left/right/bottom alignments, weight of the separator between them, relative sizes, text similarity, font similarity, and background color. This layer supports APIs such as

1. Given a block, return all its buddy blocks in certain direction - top, bottom, left or right. We say two blocks to be buddies of each other if they are physically close (neighbours) and are semantically related as well. For instance, in a student's homepage, the block containing text 'email' and the block containing the actual email address are actually buddy blocks. To call them as buddies, our method would consider criteria such as spatially adjacency, matching font properties, top-alignment and/or bottom alignment, similarity of background color, and degree of coherence of closest common ancestor block the in VIPS hierarchy.
2. Given a block, return potential heading blocks for it and consider only blocks within a specified window as candidates for heading. To call block  $b_h$  as potential heading of block  $b_{src}$ , our method considers criteria such as angle(s) between top-left (and top-right) corner of  $b_{src}$  and center of  $b_h$ , alignment, ratio/difference between font properties of  $b_h$  and  $b_{src}$  and text similarity.
3. Return a ranked list of potential heading blocks within the page.

There are other complex entity relationships that can be captured in this layer. For instance, often people have a navigation bar on a web-page. We would, say for some particular application, want to reject all those blocks that seemingly are part of some navigation bar. Another interesting relationship is a *virtual table*. People use different tags such as `table/tr/td`, `ul/li`, and `dl/dt/dd` or just raw layout with `br` tag to present tabular information. Capturing this tabular organization in form of a *virtual table* can provide many clues to the classifier.

The framework provided by spatial and similarity layer is general enough and can be used to model any web-page. In our case, we project only the visual block and not the entire web-page on these layers.

## 4.6 Independent Labeling of Blocks

In this approach for entity extraction, we use a statistical model to independently label the blocks. We ignore the relationship between sub-entities and do independent labeling of the blocks i.e. labeling of a block does not effect the labeling of other blocks from the same visual block. Input to this stage is a filtered visual block. A statistical model such as CRF, SVM is used to do the labeling and the output is in terms of blocks corresponding to the sub-entities of the entity. In the case of the product information extraction task, the labels of sub-entities are *title*, *price*, *image* and *other*. Once the blocks are labeled by our model, we assemble them into meaningful entity records.

We have experimented with two statistical models-CRF and SVM as information extraction models. We next describe typical features for IE tasks over web-pages. For specific domains, it is expected that a domain expert would use a different feature set.

### 4.6.1 Features

- **Textual Features:** These features capture lexical properties. We use seven features of this category. We maintain a dictionary of word occurrences for each state (label), where a word is counted only once for all its occurrences within a training instance (visual block). Using this dictionary, we fire three different features—*WordFeature*, *UnknownWordFeature* and *WordWindowFeature*. We also maintain a

similar dictionary of words found in the alternative text associated with images. As mentioned earlier in this report, neighbourhood is difficult to capture. Given two spatial adjacent blocks, they may or may not be neighbours. In our initial attempts, we have tried to approximate the notion of neighbourhood by making a preorder traversal over the DOM sub-trees in the visual block and assuming the left and right neighbours to be the preorder predecessor and successor.

*WordFeature* is a word specific feature which indicates presence of corresponding word in the block. It is fired for a block for a particular word-state combination if the dictionary count the word in that state exceeds some threshold. The threshold is adjustable parameter and set by the user. This feature allows to capture keywords. The rarely occurring words, such those in title, receive no or small weight.

*UnknownWordFeature* is rare word feature. It is fired a block for a particular state if dictionary-count of any of the word in the block at that state is below threshold. It allows to capture the fact that a block often contains rare words. This is particularly helpful for sub-entities like title.

Window word feature, *WordWindow*, is fired in a manner similar to word feature but over a window of neighbouring blocks. The neighbours are defined in terms of the preorder sequence. This feature allows to capture the keywords that influence presence of sub-entities in their neighbourhood.

*RegexMatch* is regular expression match feature. It is fired for a regular expression pattern on a given block when the any subsequence of word sequence in the block matches the pattern. Examples of such regular expression include patterns to check presence of digits, to check presence of special characters, and so on.

*RegexMatchPerc* feature indicates percentage match of block contents with a regular expression pattern, i.e. number of tokens matching with the regular expression. Examples of such regular expression include patterns to check amount of capital words in a string, amount of digits, and so on.

Feature for regular expression match over a window, *RegexMatchWin*, is fired in a manner similar to *RegexMatch* but over a window of neighbouring. Again, the neighbours are defined in terms of the preorder sequence.

*ImageAltText\_DictFeature* is a word specific feature which indicates presence of cor-

responding word in the alternative text associated with an image. It is fired for an image for a particular word-state combination if the dictionary count of the word in that state exceeds some threshold. The threshold is adjustable parameter and set by the user. This feature allows to capture keywords such as *buy*, *product*, *a*, *click*, and so on.

*ImageAltTextStopWords* feature indicates presence of a stop word (from list of stop words specified by user) in the alternative text associated with an image. Examples of stop words include *view*, *cart*, and *click*.

*ImageAlt\_isMissing* feature indicates if the alternative text of the input image is missing or not.

- **Formatting Features:** These features capture the special HTML formatting.

*FontBoldItalic* feature is fired to indicate whether or not the block contents are italicized or bold.

*PropertyDistinctness\_Binary* is fired to indicate whether text color, background color, and font pixel size of a block is rarest amongst all the blocks in the visual block. For each of these properties, we precompute a list of values, that the property can take, sorted in decreasing order by frequency of occurrence. This feature is fired for a given block for a particular property if the property's value for the block is rarest.

*PropertyValueRank* is fired on font pixel size and font weight. For each of these properties, we precompute a list of values, that the property can take, sorted in decreasing order by frequency of occurrence. This feature's value for a given block for a particular property is set to the rank of property value, in the sorted property value list, scaled to 10.

Hyperlink feature, *HyperText*, is fired to indicate whether or not a block is a hyperlink.

*IsStrikedThrough* feature is fired to indicate whether or not contents of a block have been striked through. For instance, on product pages, often prices of competitors are striked out.

*ImageAltTextMatchWithMaxfontText* feature indicates the match of alternative text

of an image with the top  $K$  maximum font text blocks. By capturing match of an image's alternative text with potential heading nodes, we expect to find likelihood of the image being a (sub)entity. To obtain the top  $K$  maximum font nodes, we consider all the text nodes in the upper half of the bounding box of the text nodes present in the visual block. We sort these text nodes on decreasing order of font pixel size, font weight, and vertical and horizontal positions, and retain top  $K$  amongst them. For an input image (to be labeled), we find the percentage of tokens of its alternative text matching with the tokens of a maximum font node. We fire feature with value as highest such match score amongst the top  $K$  nodes.

- **Layout Features:** These features capture the spatial properties.

Relative position features, *RelHorizontalPosition* and *RelVerticalPosition* are fired to indicate the horizontal and vertical offsets of a block within the visual block. These offsets are scaled to 100.

We find a *truncated bounding box* by ignoring all the image blocks from the visual block and finding bounding box of the remaining blocks. *RelVertPosTruncatedBlock* is fired to indicate vertical offsets of a block within the *truncated bounding box*, scaled to 100.

*ImageThetaWithMaxfontNodes* feature indicates angle made by an image block with top  $K$  maximum font nodes. To find the top  $K$  maximum font nodes, the text blocks are sorted in decreasing order of font pixel size, font weight, and vertical and horizontal positions.

*ThetaWithImage* feature indicates angle made by a text block with top  $K$  images. To find the top  $K$  images, the image blocks are sorted in decreasing order of the area occupied by them.

*ImageAreaOccupied* feature indicates percentage of area occupied by an image block within a visual block.

*ImageSizeRank* feature indicates rank of an image block, in terms of area occupied, amongst all image blocks within the visual block.

*ImageAspectRatio* feature indicates aspect ratio (ratio of width to height) an image block. Images used as separators and banner images tend to have a very high value

of aspect ratio.

## 4.6.2 Information Extraction Models

We have experimented with both CRFs and SVMs as our extraction models. CRFs are probabilistic undirected graphical models based on maximum entropy principle, whereas SVMs are discriminative models based on max-margin principle. We have used the CRF package available at [21]. In case of SVMs, we used the SMO implementation of Weka[22]. Under individual labeling assumptions, CRF basically acts like a multi-class logistic classifier. Since our basic unit of labeling is a block, each data instance is basically a block.

For both the models, the basic framework provided by the CRF package was used to fire the features. We have extended the package with new features. Many of our features are numerical in nature and can take non-negative real values. It would be ideal if we could model a probability distribution of values that a feature takes. In case of CRFs, if features are used in conventional manner, then the best approximation we get is a first order polynomial. We can overcome this situation by using polynomial features of higher order. However, certain limit has to be put on the degree of highest order of polynomial that could be used and also, the computational complexity increases. Another option is to use histograms over feature values. But then to get a good approximation, one needs to fix the bin width. In case of CRFs, we have used this approximation for numerical features. Feature values are discretized into equal sized bins. The bin width and range are free parameters to control the binning and need to be set by user. In case of SVMs, we used polynomial kernels of various degrees and RBF kernels. With non-linear kernels, SVMs are able to handle the numerical features well.

Output of CRF is probability distribution over labels, while the output of SVM does not have probabilistic interpretation. Since we are doing independent labeling, so the output of the models over a web-page can be inconsistent. While doing best label selection (Section 4.6.3), the inconsistencies for a particular label (sub-entity) in a web-page are resolved by looking at the confidence (probability) scores, for that label, assigned to the different candidate labels. Output from SVMs cannot be used for such comparison. We used a logistic regression classifier over the SVM output to get interpretable confidence scores. Each candidate with its scores (the original SVM outputs) for different labels and the actual label formed one training instance for the logistic regression classifier.

### 4.6.3 Best Label Selection

The information extraction model can mark multiple blocks from a visual block with same labels or it may end up not marking any block for some labels. For instance, CRF may label two blocks from same visual block as *title* and none of them as *price*. Such inconsistencies can arise because we are doing independent labeling and not collective labeling. Notion of inconsistency is domain dependent. In case of product information extraction task, it is a constraint that each visual block has exactly one title and price field and atmost one image. The best label selection phase assembles labeled blocks (sub-entities) into meaningful entity records. For each label, we consider blocks with that label prediction and mark the one with highest probability with that label while others are marked with label *other*. For a given block, output from CRF can be directly interpreted as a probability distribution over labels, but output of SVM cannot be directly treated as probability values. As of now, we use logistic regression classifier to convert scores of SVMs into probability values.

## 4.7 Collective Labeling of Blocks

In this approach for entity extraction, we collectively label all the blocks within a visual block. Under this mode, blocks can mutually affect each other's labeling and this enables us to capture sub-entity interactions. We cast the problem under graphical model settings and do markovian labeling of blocks. We have experimented with CRF (for general graphs) as an information extraction model. We next describe our feature set.

### 4.7.1 Features

The feature set that we use for collective labeling model is a superset of the individual labeling model feature set (Section 4.6.1). Hence, we describe only the additional features here.

- **Independent Labeling Model Features:** The independent labeling model outputs, for a given block, a probability distribution over labels. It selects the best label for each block independently and outputs certain labeling which may be inconsistent. It then does best label selection and outputs a labeling which might be

different from the original labeling of blocks. We use the various outputs of independent labeling model to form three different types of features. We found in our experiments that SVM with RBF kernels gives best results for independent labeling, and so we use SVM with RBF kernel as independent labeling model.

*SVMScore* feature indicates the probability, as output originally by the independent labeling model, of a block being assigned certain label.

*SVMLabelings* feature indicates for a given block, whether certain label is the first best (most probable) label or not. To determine the best label, it uses the original probability distribution output by the independent labeling model. Similarly, another feature is fired to indicate, whether, for a given block, certain label is the second best label or not.

*SVMBestLabels* feature indicates for a given block, whether certain label is the best label or not. To determine the best label, it uses the labeling obtained after applying best label selection over original labeling of independent labeling model.

- **Pairwise Features:** Pairwise feature fire on pair of blocks. For each block we query the spatial and/or similarity layers to get an appropriate neighbourhood and then for every possible pairing of the current block with its neighbours, we fire the specific feature. There are six formatting features and thirteen layout features. The formatting features fire only for text blocks and not images. For computational tractability we try to keep the neighbourhood sets to be of small sizes. It otherwise leads to a highly connected graph with many overlapping large sized cliques and this makes computation very slow.

*GreaterFontWeight* indicates whether the first block has greater text font weight than the second block. It is a asymmetric feature i.e.  $f(a, b) \neq f(b, a)$ .

*GreaterFontSize* indicates whether the first block has greater text font size than the second block. It is a asymmetric feature.

*EqualFontWeight* indicates whether the first block has the same text font weight as that of the second block. It is a symmetric feature i.e.  $f(a, b) = f(b, a)$ .

*EqualFontSize* indicates whether the first block has the same text font size as that of the second block. It is a symmetric feature.

*EqualBgColor* indicates whether the first block has the same background color as that of the second block. It is a symmetric feature.

*EqualColor* indicates whether the first block has the same text color as the second block. It is a symmetric feature.

*IsAbove* indicates whether the first block is above the second block. A block  $b_1$  is above another block  $b_2$  when  $b_1$ 's bottom is above or at same level as that of  $b_2$ 's top. It is a asymmetric feature.

*IsLeft* indicates whether the first block is to the left of the second block. A block  $b_1$  is left of another block  $b_2$  when  $b_1$ 's right edge is left of or coincides with left edge of  $b_2$ . It is a asymmetric feature.

*GreaterWidth* indicates whether the first block has greater width than the second block. It is a asymmetric feature.

*EqualWidth* indicates whether the first block has the same width as that of the second block. It is a symmetric feature.

*GreaterHeight* indicates whether the first block has greater width than the second block. It is a asymmetric feature.

*EqualHeight* indicates whether the first block has the same width as that of the second block. It is a symmetric feature.

*AreaRatio* indicates the ratio of areas of the two blocks. The actual feature value is  $\max(b1.area/b2.area, b2.area, b1.area)$  and so the feature is symmetric.

*TopAlignment* indicates whether the top edges of two blocks are at same level or not. It is a symmetric feature.

*BottomAlignment* indicates whether the bottom edges of two blocks are at same level or not. It is a symmetric feature.

*LeftAlignment* indicates whether the left edges of two blocks are at same horizontal position or not. It is a symmetric feature.

*RightAlignment* indicates whether the right edges of two blocks are at same horizontal position or not. It is a symmetric feature.

### 4.7.2 Information Extraction Models

We have used CRF as the entity extraction model. We used the UGEM (Undirected graphical Exponential Model) package for our experimentation. UGEM is basically an implementation of CRF for arbitrary graphs and is available locally in author's university. UGEM has three main components - the feature engine, the trainer and the inference engine. Feature engine is meant for firing features over an input data instance. The pairwise features indicate the edges to be included in the graph. We extended the existing feature engine to include new features. The inference engine includes implementations of message passing algorithms. We extended the existing inference engine and implemented Junction tree based exact inferencing algorithms and loopy belief propagation based approximate message passing algorithms. Further, we also used an implementation of constrained max product algorithm reported recently in [1]. UGEM has default implementations of likelihood based Gradient Descent trainer and voted perceptron based Collins trainer. We extended the Collin's trainer to account for top-k assignments rather than just the topmost assignment.

Again since, CRFs cannot handle numerical features well, we used discretization in case of real-valued features. The max-product algorithm implemented in [1] can handle constraints on labels and we use it to ensure that the entity record extracted is consistent.

# Chapter 5

## Experiments and Analysis

In this chapter we present our experimental results for independent and collective labeling approaches. Experiments were performed with product information extraction task with a real-life dataset. We have built a product information extraction system over our framework. For this task, the entity is the product described in the input web-page and possible sub-entities to be extracted are product title, manufacturer name, price, discount, description and associated image. In our experiments, we have dealt with product title, price and image. We tagged around 288 product pages using the tagging tool described in Section 4.2. We ensured that pages were from different sites and that there was variation in them. All of these pages contained description of only one entity, i.e. they had exactly one visual block each.

### 5.1 Experiments with Independent Labeling Approach

In the case of independent labeling approach, we report performances of three models - CRF, SVM with degree three polynomial kernel, and SVM with RBF kernel for different feature sets. We also present the improvements obtained with best label selection and the effect of training dataset size on accuracy for the three models.

For our experiments, we formed different feature sets. Table 5.1 lists these feature sets. A detailed description of these features is present in Section 4.6.1.

#### 5.1.1 Effects of Different Feature Sets

We experimented with different combinations of these feature sets with all of our models. Table 5.2 shows these results for the case when no best label selection was done. Table 5.3

Feature Set	Features
Textual	<i>WordFeature</i> , <i>UnknownWordFeature</i> , <i>WordWindow</i> , <i>RegexMatch</i> , <i>RegexMatchPerc</i> , <i>RegexMatchWin</i> , <i>ImageAltText_DictFeature</i> , <i>ImageAltTextStopWords</i> , and <i>ImageAltText_isMissing</i> .
Formatting	<i>FontBoldItalic</i> , <i>PropertyDistinctness_Binary</i> , <i>PropertyValueRank</i> , <i>HyperText</i> , <i>IsStrikedThrough</i> , and <i>ImageAltTextMatchWithMaxfontText</i> .
Layout	<i>RelHorizontalPosition</i> , <i>RelVerticalPosition</i> , <i>RelVertPosTruncatedBlock</i> , <i>ImageThetaWithMaxfontNodes</i> , <i>ThetaWithImage</i> , <i>ImageAreaOccupied</i> , <i>ImageSizeRank</i> , and <i>ImageAspectRatio</i> .

Table 5.1: Feature sets used for Independent Labeling experiments.

shows the results when best label selection stage was used to remove the inconsistencies in labeling due to individual labeling of blocks from same document (Section 4.6.3). All the reported numbers were obtained by averaging over 5 experiments, where in each experiment, a random split of 101:187 for training-test data split was used.

As is evident from the experimental results, better results are obtained when the entire feature set was used. This is in support with our claim that harnessing formatting and layout clues available from a web-page can give in better results than using just the textual properties. An important observation is that for different fields different features have importance. In case of product price, the textual features are most effective amongst all the features. This is because the product prices usually have digits and currency symbols in them and are surrounded by keywords such as *price* and *quantity*, and these properties are best captured by textual features. In case of product image, the layout features are most important. This is because the product image have properties such as they being found below title and their size amongst all other images is significant. For product name, we found all the features were important.

Another important observation is that SVMs outperformed CRFs. SVM with RBF kernels and SVM with third degree polynomial kernel performed much better than CRF. The probable reason for such behaviour is that SVMs with higher order kernels better model the distribution over values of numerical feature as compared to CRFs. SVM results had less variation (measured as standard deviation amongst results of different

Feature Set	Method	Name	Price	Image	Other
Textual	SVM-RBF	7.48	85.02	17.80	91.83
	SVM-Poly	43.12	85.78	22.55	92.04
	CRF	45.14	82.83	25.20	91.71
Textual + Formatting	SVM-RBF	54.13	86.67	17.80	91.71
	SVM-Poly	58.86	86.13	22.55	91.98
	CRF	55.58	84.45	24.85	91.36
Textual + Layout	SVM-RBF	67.30	85.82	85.35	95.82
	SVM-Poly	65.49	85.19	76.62	94.90
	CRF	61.45	80.97	83.30	94.68
Textual + Formatting + Layout	SVM-RBF	71.01	86.56	85.29	95.82
	SVM-Poly	68.25	85.31	76.57	94.82
	CRF	67.94	81.91	83.16	95.06

Table 5.2: F1 values for various feature sets and models without best label selection.

Feature Set	Method	Name	Price	Image	Other
Textual	SVM-RBF	7.57	94.69	23.13	93.75
	SVM-Poly	<b>48.24</b>	<b>94.36</b>	28.20	<b>94.37</b>
	CRF	47.66	85.04	<b>31.71</b>	93.70
Textual + Formatting	SVM-RBF	66.94	<b>94.58</b>	23.13	95.12
	SVM-Poly	<b>73.93</b>	93.50	28.24	<b>95.38</b>
	CRF	68.09	88.06	<b>31.10</b>	94.75
Textual + Layout	SVM-RBF	<b>72.46</b>	<b>94.67</b>	<b>90.31</b>	<b>97.42</b>
	SVM-Poly	72.02	91.79	83.66	96.73
	CRF	65.18	83.28	84.45	95.84
Textual + Formatting + Layout	SVM-RBF	<b>80.32</b>	<b>94.58</b>	<b>90.31</b>	<b>97.77</b>
	SVM-Poly	79.29	90.62	83.72	97.08
	CRF	76.77	82.84	84.29	96.44

Table 5.3: F1 values for various feature sets and models with best label selection enabled.

experiments) as compared to CRF results and SVM gains more with best label selection.

### 5.1.2 Effect of Best Label Selection

Best label selection is used to remove the inconsistencies due to independent (individual) labeling of blocks coming from the same page. Table 5.2 and 5.3 show the results for the cases when best label selection stage is applied and not applied, respectively. Figure 5.1 shows the improvement in results of the three models with best label selection being done.

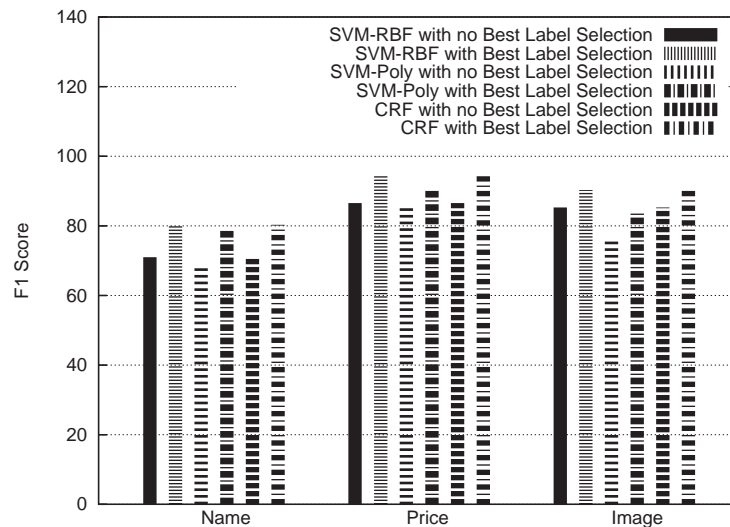


Figure 5.1: Effect of best label selection for Product information extraction task.

Best label selection boosts the F1 score for various fields. We found that best label selection tends to sometimes decrease the recall. This is because in the best label selection we only handle the cases where a certain label was assigned to more than one blocks but not those cases where a certain label was assigned to none of the blocks, and hence, the best we can expect is no change but not increase in the recall. In spite of decrement in recall, the F1 score is increased due to large increment in precision. Apart from removing inconsistencies, the best label selection stage can be thought of as a crude approximation to collective labeling. This suggests that collective labeling can result in improvement of the extraction accuracy.

### 5.1.3 Effect of Dataset Size

We varied the training dataset size to see its effect on the models' predictions. For these experiments we split the data in following way. Our dataset includes 288 labeled documents. For a chosen seed value, we randomly select 158 documents (without replacement) from the entire dataset to form test dataset. From the remaining dataset, we form training datasets of decreasing size such that there is a subset relationship between the sets. The dataset was split with 30 different seed values and the mean and variances of F1 scores were noted for all the methods and all the fields.

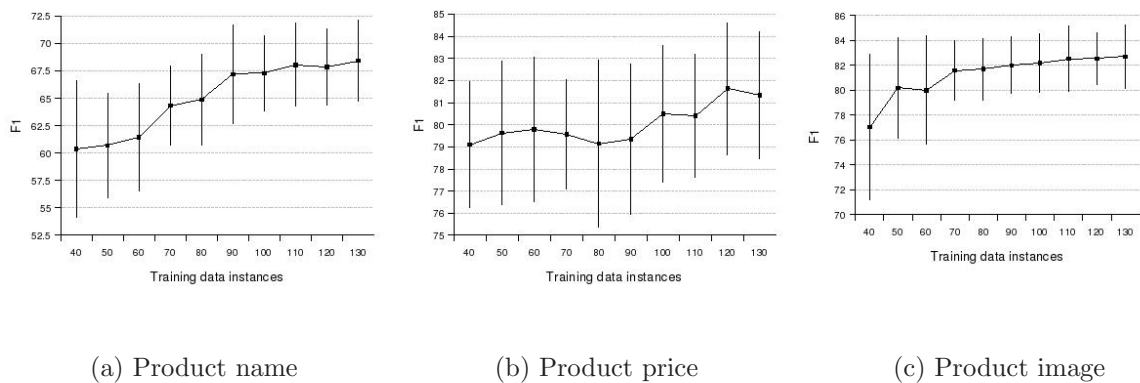


Figure 5.2: Effect of training dataset size with CRFs.



Figure 5.3: Effect of training dataset size with SVM (Polynomial kernel of degree 3).

Figures 5.2, 5.3 and 5.4 show the effect of increasing training dataset size on F1 scores of product name, price and image for CRF, SVM with polynomial kernel of degree three and SVM with RBF kernel, respectively. In case of product name, all the three

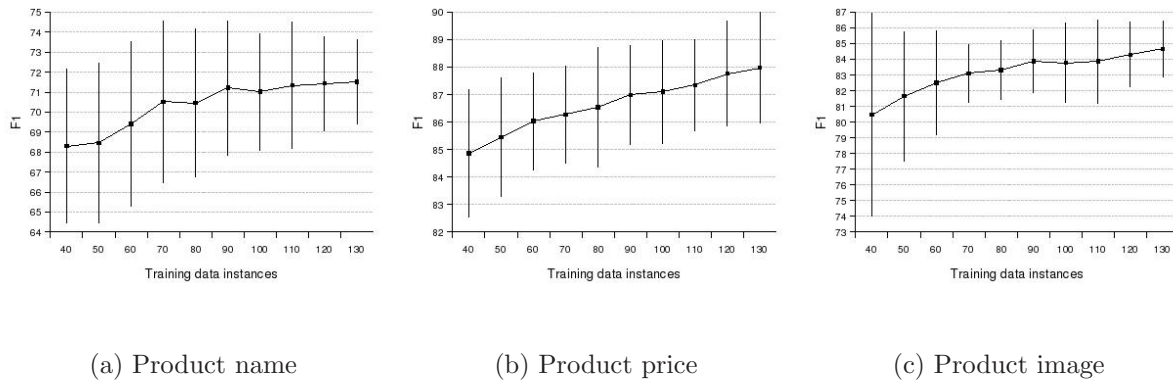


Figure 5.4: Effect of training dataset size with SVM (RBF kernel).

methods show an increase in F1 score with increase in training dataset size. CRF show an increment of around 8.5%, SVM with RBF kernel show an increment of around 3.5% and SVM with polynomial kernel show an increment of around 6.5% as the training dataset size is increased from 40 to 130 documents. In case of product price, SVMs with polynomial kernel (around 4%) and SVMs with RBF kernel (around 3%) show consistent improvement in F1 score, while with CRFs (around 2%) no useful trend is observed. For product images, all the three methods show a consistent improvement in F1 score with increase in dataset size. However, in case of SVMs with Polynomial kernels, there are some deviations. A probable reason for gain being not much in case of price field is that, for price textual features are the most important and enough textual clues might have been learnt even from small dataset with few documents.

## 5.2 Experiments with Collective Labeling Approach

For collective labeling approach, we present the performance of UGEM model with different feature sets, with different training algorithms, under different modes of training - unconstrained and constrained, and effect of beam size ( $k$ ) on Collin’s top-k trainer.

### Experimental Setting

We trained the UGEM model with gradient descent (Section 2.1.1) and Collin’s training (Section 2.1.1) algorithms. The gradient descent trainer needs sum-product based inferencing, whereas, the Collin’s trainer needs max-product based inferencing. When

there are constraints on entity structure like a particular label appearing exactly once or atmost once, then the labeling returned by max-product algorithm may not be consistent with the structure of entity. The max-product algorithm implemented in [1] can handle constraints on labeling. We have experimented with both constrained and unconstrained max-product based Collin’s trainer. We have also experimented with *beam size* for Collin’s trainer.

Once the model has been trained, during deployment or testing of the learnt model, we need to call the max-product algorithm to get the final labelings. We have experimented with both constrained and unconstrained max-product for final inferencing of labels. In case of product information extraction task, we use *exactly once* constraints for product title and price and *atmost once* constraint for product image.

We organize our features into following groups. A detailed description of these features is present in Section 4.6.1 and 4.7.1.

- **Base features:** All state (unary) features described in Section 4.6.1 and *SVMScore*, *SVMLabelings*, and *SVMBestLabels*.
- **Pairwise formatting features:** *GreaterFontWeight*, *GreaterFontSize*, *EqualFontWeight*, *EqualFontSize*, *EqualBgColor*, and *EqualColor*.
- **Pairwise layout features:** *IsAbove*, *IsLeft*, *GreaterWidth*, *EqualWidth*, *GreaterHeight*, *EqualHeight*, *AreaRatio*, *TopAlignment*, *BottomAlignment*, *LeftAlignment*, *RightAlignment*, and *ImageAspectRatio*.

For our experiments with different feature sets, we combine the Base, Pairwise formatting, and Pairwise layout features to form different feature sets. Table 5.4 lists these sets.

Feature Set	Features
Set A	Base features only
Set B	Base + Pairwise formatting features
Set C	Base + Pairwise layout features
Set D	Base + <i>IsAbove</i> + <i>IsLeft</i> features
Set E	All features

Table 5.4: Feature sets used for Collective Labeling experiments.

In our experiments, we noticed that Collin’s trainer with constrained max-product based training performs nearly the same as with unconstrained max-product based training. With the available implementations, we found the former to be faster than the latter. For Collin’s trainer, we found beamsize of 3 to be giving the most reasonable performance. Also, we have found the constrained max-product inferencing to be performing consistently better than the unconstrained max-product inferencing for both gradient descent and Collin’s trainer. We observed that the feature set D gives best performance overall. In rest of the section, we assume constrained training, beamsize of 3 and constrained inferencing to be the defaults for Collin’s trainer and feature set D to be the default feature set, unless mentioned otherwise. We also found that, for our task, the Collin’s trainer performs much better than gradient descent trainer, and hence we use Collin’s trainer as default trainer in feature set comparison experiments.

### 5.2.1 Effects of Different Feature Sets

We experimented with different combinations of features (Table 5.4) with Collin’s trainer. Table 5.5 shows our results for product information extraction task. All the reported numbers were obtained by averaging over 3 experiments, where in each experiment, a random split of 101:187 for training-test data split was used.

Feature Set	Name	Price	Image	Other
Set A	87.40	94.65	85.28	97.44
Set B	86.63	96.03	89.30	97.67
Set C	86.40	95.15	88.57	97.19
Set D	88.43	95.70	88.27	97.82
Set E	87.17	94.43	88.36	97.57

Table 5.5: F1 values for various feature sets with Collective Labeling.

As is evident from the experimental results, we cannot make claims about any particular feature set. However, an important observation to make is that the performance of the model with feature sets B, C, D and E are better than that of the best results of individual labeling (Table 5.3). In case of *name* we see an improvement of nearly 8% (with set D), in *price* we see slight improvement, and in *image* we see a drop of around 1% (with set B).

Even with feature set A, when no pairwise features are included, the collective labeling model performance for *name* is much better than that of the individual labeling model (Table 5.3). A probable reason for it could be that here in collective labeling constrained max-product is used for inferencing and imposing constraints, as against use of best label selection in individual labeling. The best label selection approach is limited in many ways, because we can implement only *atmost once* and not *exactly once* constraints. In best label selection the labels interact in a limited way, whereas in constrained max-product full-scale message passing is being done and it provides a more principled approach for imposing constraints.

Improvements with feature set B, C, D, and E over feature set A can be attributed to pairwise features. But we also realise that for the product data set, most of our current pairwise features do not help much and nothing specific can be said about any particular feature. We observe that *IsAbove* and *IsLeft* are the only useful pairwise layout features for product information extraction task. The weight learnt by the model for these features are in agreement with our observations such as *image* and *price* almost always occurs below *name*, *image* occurs most of the time to the left of *price*.

### 5.2.2 Comparison of Gradient Descent and Collin’s Trainers

Table 5.6 shows comparison between models learnt using Collin’s voted perceptron top-k trainer and the gradient descent trainer for product information extraction task. All the reported numbers were obtained by averaging over 3 experiments, where in each experiment, a random split of 101:187 for training-test data split was used.

Trainer	Name	Price	Image	Other
Gradient Descent	85.43	95.50	85.84	97.37
Collin’s top-k	<b>88.43</b>	<b>95.70</b>	<b>88.27</b>	<b>97.82</b>

Table 5.6: Gradient Descent trainer vs. Collin’s trainer for Collective Labeling.

As the results in Table 5.6 indicate, the Collin’s trainer performs much better than gradient decent trainer for all the fields (sub-entities).

### 5.2.3 Comparison of Constrained and Unconstrained Max-product for Collin’s Trainer

Table 5.7 shows the performances of models trained with constrained max-product based Collin’s trainer and unconstrained max-product based Collin’s trainer. We would like to remind that max-product inferencing is used in two different contexts in Collin’s trainer - one, during the training, when the model parameters are being learnt, and two, during deployment/testing, when the model is being applied to unseen data. During deployment/testing we use only constrained max-product. It is with training that the constrained and unconstrained variants of max-product were experimented.

Training method	Name	Price	Image	Other
Unconstrained max-product	87.17 (3.73)	95.53 (0.29)	89.12 (2.15)	97.82 (0.42)
Constrained max-product	<b>88.43</b> (1.10)	<b>95.70</b> (0.50)	<b>88.27</b> (1.04)	<b>97.82</b> (0.18)

Table 5.7: Unconstrained vs. Constrained max-product based training for Collin’s trainer for Collective Labeling. Each cell shows F1 score average (standard deviation).

Our results suggest that under both modes, the performance is nearly the same. However, we observe that the variance between results of different folds is more for unconstrained training than for constrained training.

### 5.2.4 Effect of Varying Beamsize on Collin’s Top-k Trainer

We vary the beamsize in Collin’s trainer. Beamsize is the number of model solutions of a data instance used by the trainer to update the current parameters. Table 5.8 shows the effect of varying beamsize on model performance.

The above results were obtained by averaging over 3 experiments, where in each experiment, a random split of 101:187 for training-test data split was used. We observe that as we increase beamsize from 1 to 3, there is an improvement in average F1 score. When the beamsize is increased from 3 to 5, there is a slight drop in *price* and *image*. A probable reason for it could be that beyond top 3 model solutions, the other model solutions are not very reliable and should not be used to update the parameters. We also notice that, when compared against the beamsize of 1, there is drop in variance of F1

Beamsize	Name	Price	Image	Other
k = 1	87.90 (2.62)	94.63 (0.92)	85.98 (2.86)	97.51 (0.35)
k = 3	88.43 (1.10 )	95.70 (0.50)	88.27 (1.04)	97.82 (0.18)
k = 5	88.43 (1.50)	95.50 (0.82)	87.96 (1.06)	97.81 (0.26)

Table 5.8: Effect of varying beamsize for Collin’s top-k trainer for Collective Labeling. Each cell shows F1 score average (standard deviation).

scores for beamsizes 3 and 5.



# Chapter 6

## Conclusion and Future Work

In this project, we explored the problem of information extraction (entities) from web-pages. We have build a general framework that exploits textual, visual (formatting), and spatial clues to extract entities of interest from web-pages. Its key features are

- It suggests a multi-level approach with fast-filtering techniques with high recall in initial levels and increasingly costlier model with high precision in later stages. Extracting out visual block from a web-page and filtering out irrelevant blocks using a rule based approach are instances of initial filtering done by us before using CRFs/SVMs.
- We capture a web-page in terms of spatial and similarity layers. Essentially, the web-page is being modeled as a flat 2D layout of blocks which can affect labeling of each other. The work in [17], model web-pages in as semantic hierarchies obtained by applying VIPS[16]. Our framework allows us to combine clues from VIPS hierarchy as well.
- The framework provides two alternative ways for entity extraction - Independent Labeling and Collective Labeling. While in independent labeling individually extracts each of the sub-entities ignoring label-dependencies, collective labeling does markovian labeling of blocks and extracts all sub-entities simultaneously. Our framework allows us to combine results of individual labeling into collective labeling, thus getting best of both worlds.

We implemented product information extraction task on our framework and have tried both individual and collective labeling approaches for this task. Our experiments indicate that, for our task, the collective labeling approach performed better than independent

labeling approach. It remains a part of future efforts to validate this claim on other datasets as well. We found that size of dataset and quality of training data largely affects performance of the learnt model.

Our experiments with individual labeling provide evidence that better results are obtained when textual features, HTML formatting and layout features are all used than when only textual features are used. Formatting features provide richer clues to classifier and layout features allow to us to capture spatial dependencies between sub-entities. In our experiments, we found that SVMs perform better than CRFs. The suggested reason for this observation is that SVMs handle the numerical attributes better than the CRFs.

In our experiments with collective labeling, we observed that the constrained max-product inferencing performed consistently better than the unconstrained max-product inferencing for both gradient descent and Collin’s trainer. We noticed that Collin’s trainer with constrained max-product based training performed nearly the same as with unconstrained max-product based training. For Collin’s trainer, we observed that Collin’s top-k trainer, with beamsize  $k$  greater than 1, performed better than normal Collin’s trainer in terms of average F1 score and variance. We also found that, for our task, the Collin’s trainer performed much better than gradient descent trainer. We observed that while collective labeling wins over individual labeling for our task, the only pairwise layout features that help in product information extraction task are *IsAbove* and *IsLeft*.

## Future Work

Information extraction from web-pages is a very challenging task and there is a lot that can be done. The current system can be extended and improved in many ways. We next present our ideas for future work on the framework.

The single most important challenge is the notion of continuity or adjacency on a web-page. Currently we model the web-page as a flat layout of blocks and separators, and use spatial and similarity layers to find neighbourhood and other relationships. An important extension would be to use this data structure to find continuity i.e. establish whether two blocks could be merged. This can assist in handling the case of sub-entities spanning multiple blocks. Combining the flat web-page model represented by spatial and similarity layer with the semantic hierarchy of a web-page gotten after running VIPS on

it, seems to be a promising approach and needs further exploration. To handle the more general case of sub-entities spanning only part of a block, we need to explore text-level labeling rather than block-level labeling. It would be very interesting to explore use of a hierarchical approach similar to [19] for this.

Our current system can handle web-pages with only single entity record. When there are multiple records within a single page, then records can mutually assist in each other's extraction. There are, in that case, different levels of dependencies - short range spatial dependencies within sub-entities of an entity and long range dependencies between two entities. A hierarchical approach similar to [19] can be used to capture these dependencies. This needs further exploration.

The layout features and some of the visual features are inherently numerical in nature. We found that CRFs do not effectively handle such features and discretization or use of polynomial features is the only refuge. SVMs, on other hand handle numerical features quite well. But traditionally SVMs have been used under individual labeling mode only and not for collective labeling of all sub-entities. Recently Max-margin Markov Networks (*M<sup>3</sup>Net*) were introduced in [13] and can be used for collective labeling. This needs further exploration.

Difficulty in obtaining training data to suit our requirements has been a major issue. Hand-tagging is time consuming process. We need to improve our current tagging tool in many ways. Quality and size of training dataset is very important factor that decides model performance. It would be interesting to explore an active learning like approach to selectively tag pages. Our current results are based on product information extraction task. To establish effectiveness of our approach, we need verify our conclusions on other datasets as well.



# Bibliography

- [1] Rahul Gupta and Sunita Sarawagi. Map estimation in mrfs via rank aggregation. In *Proceedings of the ICML Workshop on Learning in Structured Output Spaces*, 2006.
- [2] Nickolas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, 1997.
- [3] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.
- [4] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [5] Vinayak R. Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. Automatic text segmentation for extracting structured records. In *Proc. ACM SIGMOD International Conf. on Management of Data*, Santa Barabara,USA, 2001.
- [6] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, 1999.
- [7] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Sixth Workshop on Very Large Corpora New Brunswick, New Jersey. Association for Computational Linguistics.*, 1998.
- [8] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML-2000)*, Palo Alto, CA, 2000.

- 
- [9] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, 2003.
- [10] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, 2004.
- [11] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 1999.
- [12] K. Crammer and Y. Singer. The algorithmic implementation of multiclass kernel-based vector machines, 2001.
- [13] Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *NIPS*, 2003.
- [14] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. In *Proceedings of the 27th VLDB Conference*, 2001.
- [15] Andrew Hogue and David Karger. Thresher: automating the unwrapping of semantic content from the world wide web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, 2005.
- [16] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: A vision based page segmentation algorithm. Microsoft Technical Report (MSR-TR-2003-79), 2003.
- [17] Amit Jaiswal. Visual layout driven information extraction from websites. Master's thesis, Kanwal Rekhi School of Information Technology, Indian Institute of Technology- Bombay, Mumbai, 2005.
- [18] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2D conditional random fields for web information extraction. In *Proc. 22nd International Conf. on Machine Learning*, Bonn, Germany, 2005.
- [19] Sanjiv Kumar. *Models for Learning Spatial Interactions in Natural Images for Context-Based Classification*. PhD thesis, The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2005.

- 
- [20] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR*, 2004.
- [21] Sunita Sarawagi. CRF package. <http://crf.sourceforge.net/>.
- [22] University of Waikato. Weka—Machine learning software in Java. <http://sourceforge.net/projects/weka/>.
- [23] Sunita Sarawagi. Sequence data mining. A Technical Note, IIT-Bombay.
- [24] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 273, Washington, DC, USA, 2003. IEEE Computer Society.
- [25] William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA*, 2004.
- [26] Amit Chandel, P.C. Nagesh, and Sunita Sarawagi. Efficient batch top-k search for dictionary-based entity recognition. In *Proc. of the 22nd IEEE Int'l Conference on Data Engineering (ICDE)*, 2006.



# Acknowledgements

I take this opportunity to express my sincere gratitude for **Prof. Sunita Sarawagi** for her constant support and encouragement. Her excellent guidance has been instrumental in making this project work a success.

I would like to thank Amit Jaiswal for his constant help throughout the project. I would like to thank members of the **Data Mining Research Group** at KReSIT for their valuable suggestions and helpful discussions. I owe special thanks to Kaushal and Sandeep for their constant support and encouragement.

I would also like to thank my **family** and **friends**, who have been a source of encouragement and inspiration throughout the duration of the project. I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.

Last but not the least, I am thankful to the Almighty, with whose grace this project has been a success.

**Sandesh Tawari**

I. I. T. Bombay

July 13<sup>th</sup>, 2006

