SCAD: Collective Discovery of Attribute Values

Anton Bakalov*
U. Mass., Amherst
abakalov@cs.umass.edu

Ariel Fuxman
Microsoft Research
arielf@microsoft.com

Partha Pratim Talukdar†
Microsoft Research

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ac.in

ABSTRACT

Search engines today offer a rich user experience, no longer restricted to "ten blue links". For example, the query "Canon EOS Digital Camera" returns a photo of the digital camera, and a list of suitable merchants and prices. Similar results are offered in other domains like food, entertainment, travel, etc. All these experiences are fueled by the availability of structured data about the entities of interest.

To obtain this structured data, it is necessary to solve the following problem: given a category of entities with its schema, and a set of Web pages that mention and describe entities belonging to the category, build a structured representation for the entity under the given schema. Specifically, collect structured numerical or discrete attributes of the entities.

Most previous approaches regarded this as an information extraction problem on individual documents, and made no special use of numerical attributes. In contrast, we present an end-to-end framework which leverages signals not only from the Web page context, but also from a collective analysis of all the pages corresponding to an entity, and from constraints related to the actual values within the domain.

Our current implementation uses a general and flexible Integer Linear Program (ILP) to integrate all these signals into holistic decisions over all attributes. There is one ILP per entity and it is small enough to be solved in under 38 milliseconds in our experiments.

We apply the new framework to a setting of significant practical importance: catalog expansion for Commerce search engines, using data from Bing Shopping. Finally, we present experiments that validate the effectiveness of the framework and its superiority to local extraction.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

*Work done while at Microsoft Research.
†On contract.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

Keywords

Weak Supervision, Collective Information Extraction, Integer Linear Program, Commerce Search, Attribute Discovery

1. INTRODUCTION

Search engines today offer a rich user experience far beyond "ten blue links". For example, a query such as "canon rebel eos" produces a result that includes a photo of the digital camera, and a list of suitable merchants and prices. Similar results are offered for queries in domains such as entertainment, travel, etc. All these experiences are fueled by the availability of structured data about the entities of interest (restaurants, consumer products, etc.) In some cases, the structured data is obtained via data feeds from specialized providers, and constructed with extensive manual input. In other cases, wrappers are used to extract information from Web pages.

However, these data acquisition models have inherent limitations. Manual techniques cannot scale to the large number of entities that need to be supported, and cannot keep pace with the constant emergence of new entities (e.g., new products released to the market). Wrapper-based techniques [19] are sometimes an effective solution; but they are site-specific, and applicable only to sites whose structure and format is quite predictable. As a result, they tend to be brittle and require continual maintenance [7]. These limitations suggest the need for systems that satisfy the following requirements. First, structured data acquisition should be done in a fully automated way. Second, the techniques must be site-independent and should not make strong assumptions about Web page formatting or structure.

In this paper, we present SCAD[1], a system designed to address these requirements. SCAD learns to acquire structured data for entities of a given category (e.g., digital cameras). For each category, it is given a) a schema (including a set of attribute names), b) a small set of seed entities together with their values for the attributes in the schema, and c) a set of related Web pages (e.g., merchant offer pages for a product). SCAD then learns to automatically extract the values of attributes in the schema for other entities in the category. SCAD imposes very little cognitive burden on the trainer: in our experiments, we show that it suffices to provide seed sets with as few as 20 entities per category.

Our work is related to previous efforts in named entity

[1]SCAD stands for "Structured Collective Attribute Discovery". Discovery from scads of unstructured pages.

recognition (NER) [19]. However, with rare exceptions that we discuss later, NER depends exclusively on modeling the local left and right contexts of mentions of entities in unstructured text. SCAD goes beyond the local context modeling of NER systems, and incorporates various *global* signals into the extraction process. The first signal is that we find entity-level consensus among the candidate attribute values across multiple pages. E.g., even if we have an extraction error from one Web page, we may be able to recover by leveraging extractions for the same attribute from other Web pages. Second, we exploit category-level (i.e., across multiple entities) value distributions to reduce spurious extractions. Third, SCAD can effortlessly incorporate a wide variety of constraints from world knowledge, e.g., a value on a Web page is unlikely to refer to more than one attribute in the schema, or the disk capacity of a laptop is larger than RAM size.

Our current implementation uses a general and flexible integer program to integrate these global signals with local context information from the Web pages. Although it is a combinatorial optimization at an entity level, the number of constraints is modest and optimization takes only dozens of milliseconds per entity.

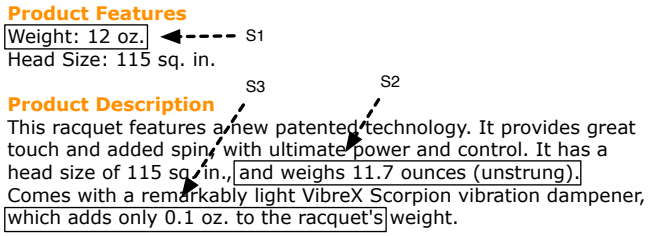

Figure 1: Sample snippets describing the tennis racquet *Wilson BLX Khamsin-Five*.

To illustrate the worth of these signals, consider the following example. Suppose that our goal is to produce structured data for consumer products. (This is a fundamental task in consumer product search engines like Yahoo! Product Search and Bing Shopping.) In this context, an entity is a product, such as the *Wilson BLX Khamsin-Five*, which belongs to the product category *Tennis racquets*. Salient attributes for this category would be "Brand", "Manufacturer", "Head Size", "Strung Weight", "Unstrung Weight", etc. Assume that we are given a set of Web pages that correspond to this racquet. (We will explain later how Commerce search engines typically gather this information.) In Figure 1, we show an excerpt of such a page.

Suppose that the schema contains (among others) the following attributes: "Strung Weight" and "Unstrung Weight"; and that we would like to obtain their values for the racquet *Wilson BLX Khamsin-Five*. SCAD would start by spotting that "12 oz." and "11.7 oz." are values that might be associated to an attribute (for example, because the schema contains numeric attributes). It would then construct snippets of text centered around the values. The snippets for the values "12 oz." (snippet s_1) and "11.7 oz." (snippet s_2) are highlighted with rectangles in Figure 1. A local extractor will produce, for a given snippet, a probability distribution ϕ over the attributes in the input schema. In our example, the local extractor will be highly confident that snippet s_2 is about unstrung weight ($\phi(s_2, \text{"unstrung weight"}) = 1$, say), but it will be less certain about the attribute corresponding to s_1 (both "strung weight" and "unstrung weight" will have comparable probabilities). Since the local model is certain that snippet s_2 should be associated to "unstrung weight", it is reasonable to assume that its value (11.7 oz.) is the right value for this attribute. But the local model is not conclusive about the strung weight of the racquet.

Now, let us show how the entity and category-level constraints help to reduce the uncertainty of the local model. Consider two of the entity-level constraints of our framework: (C1) each snippet should be assigned exactly one attribute; (C2) each attribute should be assigned at most one value. Since s_2 has been assigned "unstrung weight", due to constraint C1, it cannot be assigned any other attribute. Furthermore, due to constraint C2, "unstrung weight" cannot be assigned the value "12oz". (Otherwise, the same attribute would have multiple values.) Thus, s_1 cannot be assigned to the attribute "unstrung weight". Since the only other plausible candidate for s_1 is "strung weight" and its value is 12 oz., we can now conclude that the value of "strung weight" should be 12 oz.

Notice that there is some chance that 12 vs 11.7 oz. is just precision variation, and both are unstrung weights. This is a valid configuration considered by SCAD, but in this case it is ruled out due to the fact that there is signal from the local model ϕ about the fact that 11.7 oz. is associated to "unstrung weight". Another example of an entity-level constraint that can help us decide how to assign attributes to values is that "strung weight" should be greater than "unstrung weight." Thus, assigning "11.7 oz." and "12 oz." to "strung weight" and "unstrung weight", respectively, is not allowed. In the paper, we refer to these constraints as inter-attribute constraints.

To understand how the category-level constraints work, consider the snippet s_3 shown in Figure 1 – "which adds only 0.1 oz. to the racquet's". From the textual content of s_3, the local model might assign it some probability of being associated to "strung weight". However, from an understanding of the value domain for this attribute, it is obvious that the quantity 0.1 oz. cannot be associated to the weight of a tennis racquet.

The rest of the paper is organized as follows. In Section 2, we present related work; and in Section 3 we give an overview of our framework. In Section 4, we provide the details of the global assignment optimization problem solved by SCAD, and in Section 5 we present SCAD's local model. The experimental results are presented in Section 6.

2. RELATED WORK

Creating a structured entity description involves locating a mention of the entity, mentions of structured attributes, and evidence of suitable associations between the entity and attributes. Detecting mentions of entities is a well-developed field [19], as is detecting mentions of relations (not necessarily attribute relations) between entities [3]. However, there is relatively little work on attribute extraction that is collective across attribute mentions in a document and that aggregates evidence over multiple documents. (The quantity consensus query system [1] does only the latter.)

Several methods for incorporating non-local information for Information Extraction have been previously proposed [2, 10, 20]. These methods try to enforce label consistency

where the same token mentioned multiple times in a document is assumed to have the same label. Unfortunately, this assumption is not valid in our application setting. For example, in a document from the Washing Machines category with multiple mentions of the token *3*, one mention may correspond to the "number of cycles" attribute, while another to the "number of doors".

As noted earlier, wrapper induction techniques [19] require sites (or documents) to be template-based [11], and they are expensive to maintain [7]. SCAD does not induce wrappers and it does not require documents to be template-based, and hence it is applicable more generally.

Whereas SCAD seeks to collectively extract attributes of entities, a large body of work, including KnowItAll [9] discover instances of types starting with a few seed instances. These are very different problems, and the building blocks from entity discovery (e.g., PMI [21]) are not necessarily applicable to our scenario. The work that is closest to ours is Kylin [23], since it also focuses on building structured representations (in their case, for Wikipedia infoboxes). Kylin extracts each attribute value in isolation, compared to the collective extraction in SCAD. One of the critical differences between SCAD and Kylin is that SCAD aggregates attribute evidence across multiple documents, while Kylin does not. As demonstrated through experiments in Section 6, we find such aggregation to be very effective in practice.

A semi-supervised method for extraction of attribute-value pairs from product descriptions is presented in [17]. In contrast to that work, SCAD not only assigns attributes to values in context, but it also constructs an entity-specific record consisting of attribute-value pairs, with one (or more) value(s) per attribute. We note that this is a harder task as it involves an additional step of resolving value ambiguity for a given attribute. While attribute extraction itself (without value) is the goal of [16], SCAD focuses on the extraction of *values* of attributes. Field compabilities learned by [22] can be used as Entity-level constraints in SCAD (Section 4.3.1). Unlike [22], we do not assume the availability of quantity-attribute mapping as input, and instead estimate it from the data by using a local model, as described in Section 4.4.

Some earlier work [1, 8, 15, 24] focused on quantity queries. Some of these exploited associative reinforcement between similar quantities, but not simultaneously the dissociative nature of quantities within a discourse/page (i.e., that two quantities within a discourse tend to pertain to *different* attributes).

As a global assignment problem between attributes and values, it is natural to express our problem using integer programs, and this has been done in the past, e.g., in the CCM framework [4] for NLP tasks. However, existing instantiations of the CCM framework [4, 18] appear not to handle all the constraints we would like to impose, including consensus and constraints involving quantitative attributes. For example, we cannot see how to express in CCM that assigning 0.3 pounds to battery weight forbids us from assigning 0.2 pounds to gross weight. Therefore, we use additional special variables in SCAD's integer program (see Section 4.2).

Like SCAD, most collective information extraction algorithms must reconcile local and global evidence, and aggregate them. EntityRank [5] extracts records with structured fields (e.g., email and phone number of a researcher) from multiple pages, and ranks tuples using a majority vote from source pages weighted by their PageRank. However, EntityRank does not exploit constraints involving value distributions or involving multiple attributes.

3. FRAMEWORK OVERVIEW

Attribute inference can be modeled as filling in an incomplete table. Each row corresponds to an **entity**, and the whole table is expected to pertain to a coherent **category** of entities. While the framework allows any kind of entities, in this paper we will usually draw examples from Commerce search, so entities will often be consumer products. Tennis racquets and laptop computers are examples of categories. The Wilson BLX Khamsin-Five and the Lenovo X300 Thinkpad are entities belonging to those respective categories. Entities have **attributes**, such as string tension, strung weight, and battery life. These attributes are represented as columns in the table for the category.

We will regard the cell values of the tables as structured data, either categorical or quantitative attributes of the entity represented by the row. While our framework can support both categorical and numeric attributes, our experiments will often focus on the latter. For simplicity we will also assume that there are no multivalued attributes.

3.1 Training phase overview

In the beginning, a few rows in the table have all columns filled out. In addition, each of these rows has associated with it a set of **contexts**. A context is an extent of text that provides evidence of the attribute values for the entity. For example, in the domain of consumer products, it is often easy to find Web pages that are dedicated to a product and come from merchants, manufacturers, reviewers, users, etc. (For sources that are grossly heterogeneous or have a lot of site boilerplate, we will assume that suitable page segmentation and key content panel extraction techniques exist.)

As Figure 2 shows, we collected contexts (where entities are mentioned together with attributes) from two sources: Web pages and offer pages. We used suitable keyword queries and the Bing Web search API[2] to collect Web pages, and offer pages were collected from online merchants. (More details on the collection of both kinds of pages in Section 6). The pages are first passed through a snippet extractor that locates text around attributes of desired types.

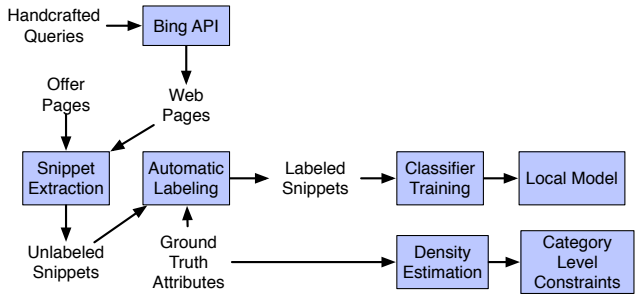

Figure 2: Training phase.

Our framework uses a *local model* of compatibility between a snippet and an attribute. The local model has to be trained using ground truth to associate snippets with attributes. The ground truth consists of structured, correct

[2]http://www.bing.com/developers

values for attributes. We look for these values in the collected snippets (see Section 5.2) to automatically generate training data. Similar mechanisms have been used in Kylin [23], and other systems. One novelty of our framework is that we also use the "gold data" to build density estimates (Section 4.3.3) of attributes, which is then used in our *global* optimization.

3.2 Deployment phase overview

Continuing with the incomplete table metaphor, the remaining rows have available unstructured contexts, but all of their cells are empty. Our task during the deployment phase, sketched in Figure 3, is to collectively mine the contexts in order to fill in the empty cells.

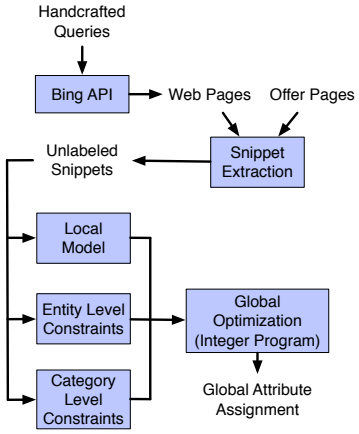

Figure 3: Deployement phase.

Context Web pages are obtained and snippets extracted as before. However, the local model now provides only one form of input to a global optimizer. Other inputs come from entity-level assignment constraints, as well as the densities estimated during the training phase.

In our current implementation of the framework, all these signals and constraints are input into a flexible and general integer linear program (ILP), which is then solved using the Microsoft Solver Foundation [3].

4. GLOBAL ASSIGNMENT OPTIMIZATION

In this section, we explain in detail the global optimization problem solved by SCAD. We first present terminology and notation, and then discuss the different constraints used for collective extraction. Finally, we introduce the objective function, and illustrate it with an example.

4.1 Terminology and notation

A **context** c is an extent of text that pertains to an entity and provides evidence for its attribute values. A context may span a page or a region of a page. A **snippet** $c.s$ is a segment of text in a context c which contains a **value** $c.s.v$ that is a candidate to be associated with an attribute of the entity. In general, a context can contain multiple snippets.

For example, the Web page region shown in Figure 1 is a context. In that Figure, we can see snippets such as "Weight: 12 oz." and "and weighs 11.7 ounces (unstrung)". The value for the former snippet is 12 ounces, while the value for the latter is 11.7 ounces. (Note that these are no longer considered as strings, but are converted into numbers with associated units; in particular, units can be converted.) We use the following rules to create snippets. Once we detect an attribute value, we consider the six words to the left and right of it. If there is a number, or some distinctive HTML tags (such as `<tr>, <li>, <title>`), then we include all the words up until that token. If we come across a mention of an attribute name that we know about, then we include it in the snippet and disregard everything beyond it.

Each category is associated to a table. The table has a **schema**, consisting of a set of typed attributes. We will denote an attribute with a. The **types** of the attributes are domain-specific, and have **units** associated with them. For example, "strung weight" and "head size" are attributes of the Tennis Racquets category, whose types are "weight" and "area", respectively. Example of a units for weight and area are "lbs." and "square inches", respectively. Strictly speaking, an attribute is just an unambiguous column ID in the table, but it is described by one (or more, synonymous) names.

Let e be an entity and $a_1, \ldots, a_n$ be the attributes of its category's schema. The goal of SCAD is to produce a tuple $\langle v_1, \ldots, v_n \rangle$ such that each v_i is associated to its corresponding a_i. We will talk about a snippet $c.s$ *being assigned* to an attribute a, or vice versa. This will mean that $c.s.v$ is a *candidate value* for attribute a. The reported value/s of an attribute depend on candidate values and other considerations such as global (exact or approximate) agreement.

Not all mentioned quantities will map to attributes in the schema. For those, we create a special **background/no attribute** NA (similar to "no assignment" [12]).

Our approach is to encode assignment decisions as 0/1 variables in a suitable integer linear program (ILP). Constraints for the ILP will come from some natural snippet and value assignment considerations. The objective will be designed using a measure of local compatibility between a snippet and an attribute.

4.2 Decision variables

We use two types of variables. The first type models the assignment of snippets to attributes. The assignment of snippet $c.s$ to attribute a is recorded in the binary variable $x(c.s, a) \in \{0, 1\}$. It is equal to 1 if attribute a is assigned snippet $c.s$; and 0 otherwise. Here a ranges over all attributes, including NA.

While the x variables associate snippets to attributes, the actual goal of catalog expansion is to associate *values* to attributes. This is denoted with the second type of decision variables: $z(v', a) \in \{0, 1\}$. It is set to 1 if attribute a is assigned any context $c.s$ with value $c.s.v = v'$, and to 0, otherwise. Because NA has no associated value, here a ranges over all attributes except NA. In contrast to current instantiations of the CCM framework [4, 18], these additional z variables enable SCAD to be more expressive and enforce additional domain-specific constraints.

4.3 Constraints

We introduce two kinds of constraints: *entity-level constraints* that keep the assignment of values to attributes globally consistent, and *category-level constraints* that prevent gross outlier values from being assigned to an attribute.

[3] `http://code.msdn.microsoft.com/solverfoundation`

Entity-level constraints themselves come in two flavors: *consensus constraints* and *inter-attribute constraints*. The former model the agreement between multiple sources of evidence for the same attribute, and they are generic in the sense that the same constraints are used for all attributes and categories. The latter model relationships that can be enforced between values of different attributes using domain knowledge.

4.3.1 Entity-level consensus constraints

The following are the consensus constraints we use.

$$\sum_a x(c.s, a) = 1 \quad \forall c\, \forall q \qquad \text{(OneTargetAttr)}$$

This constraint ensures that each snippet $c.s$ is assigned at most one attribute from the schema. We have a strict equality because the sum is over all attributes, including NA. The intuition is that when the author of a Web page writes a value, she has a single attribute in mind. Even if the same *value* appears multiple times on a page, each mention (*snippet*) is associated to a different attribute. For example, suppose that a table (furniture) is square and both its depth and width are 3 feet. The value 3 feet might appear multiple times on the page. However, a snippet "Depth: 3 feet" is associated exclusively to the depth of the table.

For many attributes we can also assert:

$$\sum_v z(v, a) \leq 1 \quad \forall a \qquad \text{(NoValueConflict)}$$

This constraint says that an attribute should be assigned at most one distinct value. (NoValueConflict) will not be appropriate if the product ID is insufficiently differentiated, as in a DVD player available in different colors, all of which are assigned the same product ID. In such cases, we can replace 1 on the rhs with a suitable upper bound to the number of distinct values we expect.

Snippet assignments induce value assignments, which we model as follows:

$$z(v', a) \geq x(c.s, a) \quad \forall a,\ \forall v',\ \forall c.s :\ c.s.v = v' \qquad \text{(ValueAssignment)}$$

This constraint ensures that if snippet $c.s$ is assigned to attribute a, then this attribute should be assigned the value $c.s.v$ embedded in the snippet. Essentially, it transfers the agreement from the snippets to the attributes.

Finally, we need to make sure that the assignments of values to attributes are sound: they must be backed up by at least one evidence snippet. This is done with the following constraint:

$$z(v', a) \leq \sum_{c.s : c.s.v = v'} x(c.s, a) \quad \forall a, v' \qquad \text{(ValueEvidence)}$$

This constraint ensures that if attribute a is assigned value v', then attribute a should be assigned at least one snippet $c.s$ whose value $c.s.v$ is equal to v'.

4.3.2 Entity-level inter-attribute constraints

While the consensus constraints are generic, the inter-attribute constraints are assumed to be given together with the schema. For example, a reasonable constraint for the category *Tennis Racquets* is that the strung weight is always larger than the unstrung weight. Similarly, the hard disk capacity of a laptop is almost always (much) larger than the RAM size. Such domain knowledge can be modeled as follows.

$$\sum_v v \cdot z(v, \text{“strung weight”}) \geq \sum_v v \cdot z(v, \text{“unstrung weight”}) \qquad \text{(AttribValueComparison)}$$

This is just one example of a very general constraint template. If we know that hard disks are at least 20 times larger than RAM sizes, or that the battery life of a laptop is at least 2 hours, or the weight of a laptop is at least one pound more than the battery weight, (AttribValueComparison) can be easily adapted to represent such domain knowledge. Any such constraint has the potential to eliminate spurious assignments and improve accuracy.

Inequality (AttribValueComparison) has a problem as posed above. For brevity, let a_b and a_s be the "big" and "small" attributes, with decision variables $z(a_b, v)$ and $z(a_s, v)$, where we wanted $\sum_v vz(a_b, v) \geq \sum_v vz(a_s, v)$. The problem arises when a_b is not assigned, i.e., all $z(a_b, v) = 0$. Then, the lhs is 0 and so the rhs is also forced to be 0. Consequently, a_s is not assigned either.

To address this problem we add more constraints:

$$v_{\max}[1 - \sum_v z(a_b, v)]_+ + \sum_v vz(a_b, v) \geq \sum_v vz(a_s, v),$$

where $[C]_+ = \max\{0, C\}$ and $v_{\max}$ is the maximum candidate value for a_s. So if all $z(a_b, v) = 0$, the first term kicks in and allows a_s to be assigned. The extra constraint can be returned to benign linear form using another variable $c \in \{0, 1\}$, $c \geq 1 - \sum_v z(a_b, v)$ for all v, etc. It would be of interest to compile a catalog of such tricks for all common inter-attribute constraints.

4.3.3 Category-level constraints

Suppose we are expanding a catalog of steel nuts and bolts. The typical weight of a bolt, as seen in the training examples, may be between 1 and 5 grams. A human would unconsciously note this range. Faced with a page that offers a nut that can hold up a weight up to 1000 kilograms, a human would never be misled into interpreting 1000 kilograms as the weight of the nut itself. We can "fake" this form of intelligence without deep language understanding, by instead modeling a distribution of values over training data for each attribute, and then incorporating this distribution into the earlier integer program.

For a given attribute a, let $V_a = \{v_{a1}, v_{a2}, \ldots v_{an}\}$ ($|V_a| = n$) be the set of values of attribute a corresponding to products in the training data. Now consider a test snippet $c.s$ with its value $c.s.v$. Intuitively, if $c.s.v$ is "close" to (the distribution estimated from) V_a, then we should feel more confident assigning this value to a.

We could measure the number of standard deviations by which $c.s.v$ differs from the average of values in V_a. This policy is not very robust in the face of multimodal distributions.

A more robust approach is to model a (Gaussian) kernel density on V_a, and then express the support at $c.s.v$ as the density at that point:

$$S(c.s.v, V_a) = \frac{1}{Z} \sum_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{ak}^2}} \exp\left(-\frac{(c.s.v - v_{ak})^2}{2\sigma_{ak}^2}\right),$$

where σ_{ak}^2 is the variance of the k^{th} Gaussian, and Z is a constant to make sure $S(c.s.v, V_a) \in [0, 1]$. This is not necessarily desirable for multimodal data, so, for simplicity, we just set $Z = 1$ and call it *unnormalized support*. Also, we set $\sigma_{a1}^2 = \sigma_{a2}^2 = \cdots = \sigma_a^2$, to a value computed using Lashkari and Golland's [13] estimator.

We want an assignment of $c.s$ to a to be possible only if $c.s.v$ has adequate support from (the density of) V_a. We estimate an attribute-specific lower threshold τ_a of support below which an assignment is not allowed. This is implemented using the following simple added set of constraints:

$$(S(c.s.v, V_a) - \tau_a)\, x(c.s, a) \geq 0.$$

The threshold τ_a is estimated as follows:

$$\tau_a = \min_i \{S(v_{ai} - \alpha\sigma_a, V_a)\}$$

where $\alpha \geq 0$ is a constant, with $\alpha = 4$ in the current set of experiments. Roughly speaking, we allow values that are at most α standard deviations away from one of the values in V_a.

4.4 Objective

Multiple assignments will generally be feasible while satisfying all the above constraints. Which one should we prefer? Input to this decision comes from a *local model* for compatibility between a snippet $c.s$ and an attribute a, represented as a conditional probability model $\Pr(a|c.s)$ that we learn through a training process. We call this a *local* model because the strength of association depends exclusively on the content of the snippet, without taking other snippets (or contexts) into account.

The object of the ILP is to maximize local compatibility subject to all the constraints already discussed. We thus define the following objective function over choices of the binary decision variables (the sum over a includes NA):

$$\sum_{c.s,a} x(c.s, a) \log \Pr(a|c.s). \qquad \text{(Objective)}$$

We will sometimes shorthand $\phi(c.s, a) = \Pr(a|c.s)$.

4.5 Example

Now that we have introduced all the elements of the global optimization problem, we can illustrate it using the example presented in the Introduction. In that example, there was a snippet s_2 such that $\phi(s_2, \text{“unstrung weight”})$ had a high value of 1. Given (Objective), the ILP would have an incentive to set $x(s_2, \text{“unstrung weight”})$ to 1. Once this is done, (OneTargetAttr) precludes any other attribute from being associated to snippet s_2. That is, $x(s_2, a) = 0$, for every attribute a such that $a \neq$ “unstrung weight”. Since $x(s_2, \text{“unstrung weight”}) = 1$ and $s_2.v =$ “11.7$oz.''$, by constraint (ValueAssignment) $z(\text{“unstrung weight”}, \textit{“11.7 oz.”}) = 1$. Recall that, according to the local model, snippet s_1 could be associated either with “unstrung weight” or “strung weight”. However, by constraint (NoValueConflict), each attribute is assigned exactly one value. Thus, $z(\text{“unstrung weight”}, \textit{“12 oz.”}) = 0$. By the (ValueAssignment) constraint, $x(s_1, \text{“unstrung weight”}) = 0$ (it would otherwise set $z(\text{“unstrung weight”}, \textit{“12 oz.”})$ to 1 and reach a contradiction). Since snippet s_1 cannot be associated with “unstrung weight”, the objective function now has a strong incentive to set $x(s_1, \text{“strung weight”})$ to 1. By (ValueAssignment), $z(\text{“strung weight”}, \textit{“12 oz.”}) = 1$.

5. LOCAL MODEL

In this section, we present the details of the local model used by SCAD. In Section 5.1, we present the features used by the classifier; and in Section 5.2 we described the approach used for training it.

5.1 Features

In Section 4.1, we explained how the snippets are created. In order to associate a snippet to an attribute, the model takes into consideration various measures of matches between the text in the snippet and various elements such as word distributions in previously seen snippets, and attribute metadata (types, attribute name synonyms, etc.).

Each such match signal is called a **feature** and becomes an element in a feature vector $f(c.s, a) \in \mathbb{R}^d$ for some suitable vector space of d dimension. The overall compatibility between the context $c.s$ and the attribute a is obtained by a linear combination of the features through a **local model** $w \in \mathbb{R}^d$, and written as the dot/inner product $w^\top f(c.s, a)$.

To combine local compatibility of various $(c.s, a)$ pairs, we calibrate these to probabilities using a multiclass logistic regression

$$\Pr(a|c.s) = \frac{\exp(w^\top f(c.s, a))}{\sum_{a'} \exp(w^\top f(c.s, a'))}, \qquad (1)$$

where a is regarded as the class label. Often, feature elements are non-negative. We say a specific element of f “fires” if its value is positive, as against zero. We now describe the actual features used in our system.

Features based on attribute names.

For some attributes, we may know a priori some of the ways that they may be referred to in Web documents. For example, the attribute “Model Part Number” can appear as “MPN” and “Manufacturer Part Number”.[4] For each attribute a, we create a feature as follows. Let Names(a) be the set of possible names that we have identified for attribute a. We then create a feature function that returns 1 if the input snippet contains an element from Names(a), and 0 otherwise.

Features based on word distributions.

The features based on attribute names need to be complemented with other features because it might be impossible to determine a priori all the different ways in which an attribute can be mentioned. Furthermore, some attribute values appear without any mention of an attribute name. Thus, we also employ softer measures of textual similarity between the snippet of interest $c.s$; and snippets from training contexts already known to be assigned to attribute a for other entities.

We use two type of features based on word distributions, using TF-IDF [14] and Jensen Shannon divergence [6] as measures. For the TF-IDF measure, we can consider all the snippet in the training data for an attribute a as a “document” for a. Then, we can find the tf-idf score of a word. Given a snippet $c.s$, we compute the tf-idf value corresponding to attribute a by summing over the scores of the words

[4]In our system, we use a method for detecting these attribute name synonyms based on distributional similarity, which is outside the scope of this paper.

w_i in $c.s$. That is,

$$\text{tf-idf}(c.s, a) = \sum_{w \in c.s} \text{tf-idf}(w, a),$$

where tf-idf(w, a) is computed with respect to the training snippets for attribute a.

To compute the features based on Jensen Shannon divergence, we do the following. For each attribute a, collect all training snippets associated to a in the training data into one word bag and represent it as a distribution P_a over words (unigrams). In particular, for a word w, $P_a(w)$ is computed as follows. Let x be the number of times that x appears in the training snippets for attribute a. Let y be the total number of tokens in those snippets. Then $P_a(w) = \frac{x}{y}$. Similarly, we compute distributions for the snippet of interest $c.s$ and denote it $P_{c.s}$. The feature is then computed as $JS(P_a||P_{c.s})$.

Features based on units.

Additional strong local clues come from the units of quantity attributes. Most quantities of interest in a catalog have units like money amount, linear dimension, duration, and weight. As additional evidence for matching up $c.s$ with a, we add features that detect the typical units in which snippet $c.s$ is written.

We start with some basic features that associated to the value $c.s.v$:

- Is the token a (decimal) integer?
- Does the number represented by the token have a fractional part?
- Is the token alphanumeric? Note that some attribute values like *802.11n* may not be purely numerical.

Then we add features that draw upon the typical units in which a specific attribute a is expressed. Consider the dimensions of cameras, specifically, width. It can be expressed in millimeters, centimeters, or inches. For larger objects like furniture, feet, meters or yards may be used. Some attributes, like the number of USB ports in a motherboard, are unitless counts. To capture both classes, we use these features:

- Does the snippet contain any unit?
- For each unit u, does the snippet contain u?

5.2 Automated Training

As a training set, SCAD is given a set of "gold" entities, together with their corresponding contexts. In order to train the local model classifier, it is also necessary to assign attribute labels to the snippets in the contexts. Doing so in a manual way would be overly laborious. Thus, SCAD does it an automated fashion. In particular, for each document (context), SCAD looks at the occurrences of values, and tries to relate them to the gold data that it has available for the corresponding entity. Let us say that we have the snippet $c.s$ associated to entity e. SCAD considers three cases:

1. The snippet has a numerical value $c.s.v$ whose unit is u. If the gold data for entity e contains an attribute value pair $\langle a, v' \rangle$ such that $v' = c.s.v$ and the unit of v' is u, then the snippet $c.s$ is labeled with attribute a.

Category	Attributes
Tennis Racquets	Head Size, Length, Beam Width, Unstrung Weight, Strung Weight
Digital Cameras	Depth, Digital Zoom, Effective Camera Resolution, Flash Memory, Height, Interpolated Resolution, Max Focal Length, Min Focal Length, Optical Zoom, Screen Size, UPC, Weight, Width
Washing Machines	Number of Cycles, Number of Temperature Settings, Tub Capacity, Energy Used, Depth, Height, Width, UPC, Weight, Water Used, Max Spin Speed
Computer Memory	RAM Storage Capacity, RAM Memory Speed

Table 1: Representative set of product attributes used in the experiments.

2. If $c.s.v$ is not associated to any unit, SCAD also looks for occurrence of $c.s.v$ in the gold data for e. But as an extra assurance SCAD looks for suitable attribute names in the text of snippet $c.s$. More specifically, suppose that there is an attribute value pair $\langle a, v' \rangle$ in the gold data for e such that $v' = c.s.v$. The extra assurance is that SCAD checks whether one of the known names for a appears consecutive to the value $c.s.v$ in the text for snippet $c.s$. If that is the case, snippet $c.s$ is labeled with attribute a.
3. If none of the two cases above hold, then the snippet is associated to the the background class NA.

Since the number of snippets labeled as background usually dominate, SCAD downsamples them, in such a way that, in the end, the training set contains as many background snippets as the number the snippets for all non-background attributes.

6. EXPERIMENTS

6.1 Experimental Framework

The experiments were conducted in the Commerce search domain, with the goal of building structured descriptions for commercial products. Each description becomes a record in the catalog maintained by the Commerce search engine. All the experiments were done using data from Bing Shopping.

We considered seven product categories: TVs, washing machines, microwave ovens, refrigerators, computer memory, digital cameras, and tennis racquets. A representative set of the attributes that we focused on is shown in Table 1.

Recall that in SCAD each entity is associated to a *context*. In order to obtain the contexts for each entity (product), we considered two scenarios of significant practical importance in Commerce search, described next.

6.1.1 *Contexts from merchant offers*

In this scenario, we make use of offer feeds provided by merchants to Bing Shopping. For each product, there is a set of merchant offers; and each offer is associated to a Web page where the product can be bought. The Web pages pointed to by the links become the contexts for the entities used in SCAD.

To understand why the offers and merchant Web pages are readily available, consider the business model of Commerce search engines. In this model, users search for a product, and once they find it, they are presented with a list of links to

merchant Web pages where they can buy it. The Commerce search engine is paid by the merchants for each click to their sites, and the merchants benefit by increasing the visits to their site. Thus, both the Commerce search engine and the merchants have an strong incentive to ensure that products have associated offers.

6.1.2 Contexts from Web search

In this scenario, the Commerce Search engine wishes to build product descriptions even before the merchants provide offers for the corresponding products. One reasonable way of doing so is by leveraging the results of a general Web search engine. For example, if we want to create a description for the racquet Wilson BLX Khamsin-Five, we can issue the query "wilson blx khamsin-five" to the search engine, which would return a set of results related to the product. We can then fetch the pages corresponding to the results, and consider them as the contexts for the entity. Notice that the quality of the results depends on how effective the query is in retrieving documents that are relevant to the product. For example, the query "wilson racquets" would clearly be less effective than the query "wilson blx khamsin-five". In our experiments, we always construct the queries using the name for the product, which includes identifiers such as brand and model. This approach proved to be effective in our experiments; we leave the study of other query constructions for context retrieval as future work.

In SCAD, we assume the availability of a training set of products within a category. In all the experiments, we used a training set of 20 products per category. For six categories (TVs, washing machines, microwave ovens, refrigerators, computer memory, digital cameras), the attribute-value pairs for the training products were obtained from data existing in the Bing Shopping catalog. To stress the fact that the training data is easy to gather, for an additional category (tennis racquets) we obtained the training data directly from a site that specializes on racquets (`www.tennisracquets.com`). In the experiments, we report results for test sets of 60 products for each category, except for refrigerators and washing machines for which we used 45 and 35 products[5], respectively. When using merchant offers, we consider products that have at least eight offers associated to them. For Web Search results, we use the Bing search API to fetch top-50 results for each product/query.

6.1.3 Metrics

We measure the quality of the results by using the standard precision, recall, and $F1$ measure [14]. As a parametric knob, we use a threshold θ applied to the scores of the local model classifier. The metrics at each value of θ are then computed as follows. Let E be a set of entities (products). Let X be the set of attribute-value pairs predicted by a system (either SCAD or a baseline) for the products in E, when the local model outputs only the predictions whose score is above θ. Let Y be the ground truth attribute-value pairs for the products in E. Then, we define precision (P), recall (R) and F1 at θ as follows:

$$\mathrm{P}_\theta = \frac{|X \cap Y|}{|X|}, \quad \mathrm{R}_\theta = \frac{|X \cap Y|}{|Y|}, \quad \mathrm{F1}_\theta = \frac{2 \times \mathrm{P}_\theta \times \mathrm{R}_\theta}{\mathrm{P}_\theta + \mathrm{R}_\theta}$$

We note that the local model's prediction scores are not calibrated to confidence and to the diverse number of attribute options a snippet can have, and this may result in P-R plots that do not always show smooth inverse relations. For presentation purposes, we sometimes report the highest $F1$ measure that is obtained across values of θ (i.e., the highest $F1_\theta$, for $0 \leq \theta \leq 1$).

[5]This is due to the limited coverage of the catalog for the refrigerators and washing machines categories.

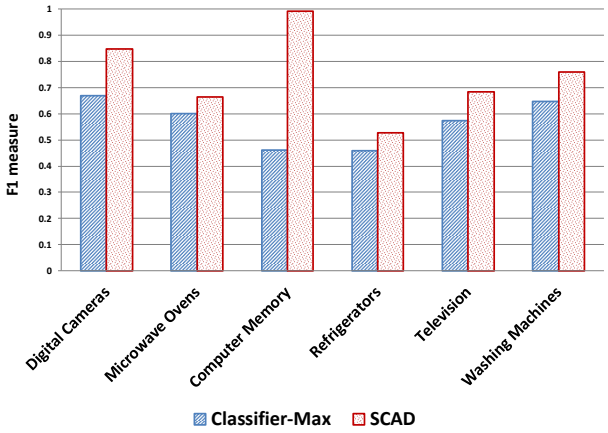

Figure 4: Comparison with Classifier-Max on all categories using merchant offers.

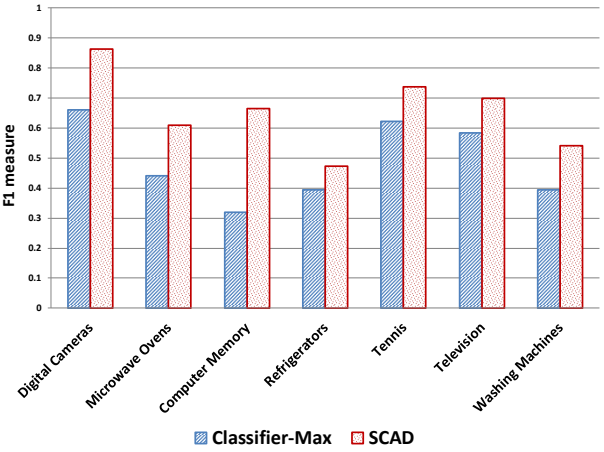

Figure 5: Comparison with Classifier-Max on all categories using Web search results.

6.2 Benefits of Collective Extraction

We now demonstrate the benefits of collective extraction by comparing SCAD against a baseline that makes assignments without requiring consensus among snippets. We call this baseline **Classifier-Max** because it uses the same local model (classifier) as SCAD, but instead of making a collective assignment of attribute values, it chooses for each attribute the value in the snippet with the the highest score according to the local model. More specifically, for each entity (product) e and attribute a, let C be the set of contexts associated to e. Then, **Classifier-Max** chooses a value $c.s.v$ such that $c \in C$ and there is no $c'.s'.v'$ such that $\phi(c'.s', a) > \phi(c.s, a)$. This baseline is inspired by the Kylin system [23], where the same approach is taken to com-

Dataset	NameMatch-Majority	Classifier-Majority	SCAD
Web search contexts			
Microwave Ovens	0.39	0.58	**0.60**
Computer Memory	0.24	0.65	**0.66**
Refrigerators	0.31	0.41	**0.47**
Digital Cameras	0.82	**0.86**	**0.86**
Tennis	0.59	0.61	**0.73**
Television	0.67	0.68	**0.69**
Washing Machines	0.51	0.52	**0.54**
Merchant offer contexts			
Microwave Ovens	0.36	0.64	**0.66**
Computer Memory	0.06	0.98	**0.99**
Refrigerators	0.50	0.49	**0.52**
Digital Cameras	0.82	**0.86**	0.84
Television	**0.68**	**0.68**	**0.68**
Washing Machines	0.67	0.70	**0.75**

Table 2: F1 comparison of SCAD with NameMatch-Majority and Classifier-Majority on all configurations. Best performance is marked in bold.

bine scores from a local model (although their local model is different from ours).

In Figures 4 and 5, we compare SCAD and **Classifier-Max** on contexts from merchant offers and Web search results, respectively. For each category, we show the highest $F1$ obtained across values of parametric knob θ. Notice that our system performs significantly better than **Classifier-Max** across all categories. The largest improvement is in the category Computer Memory on merchant offers, where the $F1$ increases by 0.52. For 10 out of the 13 category/source considered configurations, the increase is larger than 0.10; and in all cases the increase is above 0.06.

Except for Computer Memory, in all other categories the $F1$ increase on Web search results is even more pronounced than in merchant offers. For example, for Microwave Ovens, the increase on merchant offers is 0.06; whereas it is 0.17 on Web search results. The reason is that pages from Web search results are generally noisier than merchant pages: the latter tend to describe exactly one product and have prominent product descriptions. When the pages are noisier, the local model makes more mistakes, and provides greater room for improvement to the collective constraints.

To understand the benefits of SCAD in terms of precision and recall, see Figures 6 and 7. These figures present the precision/recall curves for Tennis Racquets, and Washing Machines on offers and Web search results, respectively. Notice that SCAD consistently outperforms **Classifier-Max** across values of the parametric knob. The reason for this is that **Classifier-Max** is rather brittle: a wrong local model prediction with high confidence leads to an error. On the other hand, SCAD is able to exploit redundancy in the contexts to recover from local model errors.

6.3 Justification of design decisions

We now compare SCAD against a number of baselines, where we aim for the following:

1. Evaluation of the Local Model
2. Understand the benefits of our ILP-based Global Optimization

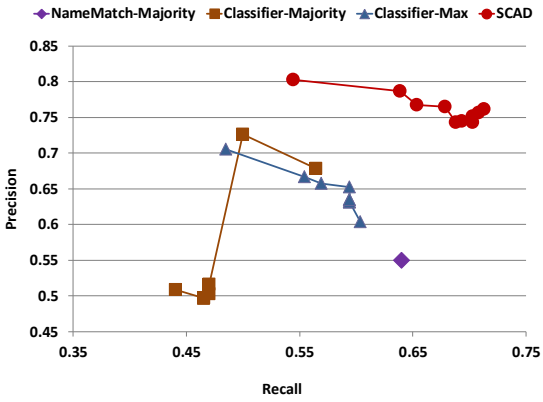

Figure 6: Precision and recall results for Tennis Racquets category.

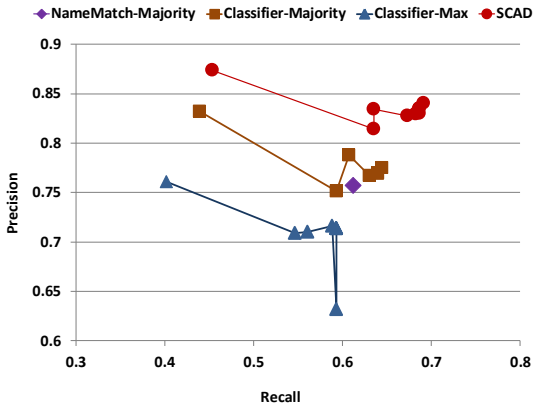

Figure 7: Precision and recall results for Washing Machines category on merchant offers.

3. Perform a constraint ablation study to show the need for the actual constraints used in the linear program
4. Understand the running time of the ILP

6.3.1 Evaluation of Local Model

While in SCAD the results of the local model are adjusted by the collective constraints, we must ensure the local model produces reasonable results. We show that this is the case by comparing against a system where the local model is not classification-based: rather, it associates value $c.s.v$ to attribute a if the name of attribute a appears in the text of snippet $c.s$. Furthermore, it performs aggregation by majority voting. We call this implementation **NameMatch-Majority**.

In Table 2, we show the highest $F1$ on all categories for SCAD and **NameMatch-Majority**. We can observe that SCAD consistently outperforms **NameMatch-Majority**. By examining the data, we observed that the improvement is more pronounced for categories where the attribute names do not appear explicitly on the Web pages, but SCAD's local model is able to get the signal from other features. For example, for Computer Memory, the attributes names are rare on Web pages (e.g., there is an attribute "RAM Memory Speed" but the word "RAM" never appears next to the

attribute values). However, these attributes are easy to detect by the local model by using the distributional similarity features with respect to previously seen snippets (e.g., the term "speed" tends to appear near the memory speed) and features related to the units (e.g., memory speed is the only attribute whose unit is in Mhz.)

6.3.2 Benefits of our ILP-based Global Optimization

To show that the aggregation approach of SCAD is effective, we compare against an implementation that uses the same local model as in our system (i.e., classifier-based) but does aggregation by majority voting (unlike our system, which uses an ILP). We call this implementation **Classifier-Majority**. In Table 2, we show the $F1$ measure on all categories for our system and **Classifier-Majority**. Notice that with the exception of one case (Cameras on merchant offers), our system consistently outperforms **Classifier-Majority**. The largest improvement is on the Tennis Racquets category, where the $F1$ measure increases from 0.61 to 0.73. On Figures 6, and 7, we show the precision and recall for some of the categories (Tennis Racquets, Washing machines on Offers, and Washing Machines on Web search results). It can be seen that SCAD outperforms **Classifier-Majority** across all values of the parametric knob.

6.3.3 Constraint Ablation Study

In order to justify the need for the actual constraints used in the linear program, we considered an ablation of SCAD's constraints, where only the consensus constraints are used, but not the category-level constraints. We observed that for 12 out of the 13 category/source configurations, SCAD outperforms the ablated system. Furthermore, the ablated system outperforms **Classifier-Majority**. The differences are significant in some cases. For example, for Tennis Racquets, the ablated system has an F1 of 0.68 while SCAD has an F1 of 0.73. This shows that category-level constraints are useful. Furthermore, the ablated system outperforms **Classifier-Majority**, which has an F1 of 0.62. This shows that SCAD's consensus constraints help. As another example, for Washing Machines on merchant offers, the F_1 for **Classifier-Majority** is 0.70. When we use an ILP with consensus constraints, it jumps to 0.72. If we use the entire SCAD system, the F_1 measure is 0.75.

6.3.4 Running time of ILP

Recall that in SCAD, each entity has an associated ILP. That is, the system solves n ILPs, where n is the number of entities to be processed. In the different experiments, the average running time to solve an individual ILP was 37.5 ms. per entity. The reason we do not incur a higher cost is that we have one ILP per entity and the total number of variables per ILP is only 1323 on average. All this implies that use of ILP is efficient and practical in our framework.

7. CONCLUSIONS

We have presented SCAD, a general and flexible optimization-based framework for collective extraction of attributes from unstructured text. An important motivation for the problem is to expand and complete product catalogs in e-commerce sites. SCAD requires limited amount of supervision and can be bootstrapped quickly, even in the absence of data providers. Experimental results with several product categories from Bing Shopping demonstrate the merits of our approach. As part of future work, we plan to apply SCAD on more categories, exploiting different category-specific constraints, and exploring how such constraints can be estimated reliably from limited training data.

8. REFERENCES

[1] S. Banerjee, S. Chakrabarti, and G. Ramakrishnan. Learning to rank for quantity consensus queries. In *SIGIR Conference*, 2009.

[2] R. Bunescu and R. Mooney. Collective information extraction with relational Markov networks. In *ACL*, 2004.

[3] R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *EMNLP Conference*, pages 724–731. ACL, 2005.

[4] M. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *AAAI*, 2008.

[5] T. Cheng, X. Yan, and K. Chang. EntityRank: searching entities directly and holistically. In *VLDB*, 2007.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[7] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD*, 2009.

[8] D. Davidov and A. Rappoport. Extraction and Approximation of Numerical Attributes from the Web. In *ACL*, 2010.

[9] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *WWW*, 2004.

[10] J. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[11] P. Gulhane, R. Rastogi, S. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. In *WWW*, 2010.

[12] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *SIGKDD Conference*, 2009.

[13] D. Lashkari and P. Golland. Convex clustering with exemplar-based models. In *NIPS Conference*, 2007.

[14] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] V. Moriceau. Numerical data integration for cooperative question-answering. In *EACL Workshop on Knowledge and Reasoning for Language Processing*, pages 42–49, 2006.

[16] M. Paşca. Organizing and searching the world wide web of facts–step two: harnessing the wisdom of the crowds. In *WWW*, 2007.

[17] K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI*, 2007.

[18] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *CoNLL*, 2004.

[19] S. Sarawagi. Information extraction. *FnT Databases*, 1(3), 2008.

[20] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*, 2004.

[21] P. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *ECML*, 2001.

[22] M. Wick, A. Culotta, and A. McCallum. Learning field compatibilities to extract database records from unstructured text. In *EMNLP*, 2006.

[23] F. Wu and D. S. Weld. Automatically semantifying Wikipedia. In *CIKM*, pages 41–50, 2007.

[24] M. Wu and A. Marian. Corroborating answers from multiple web sources. In *WebDB*, 2007.