Matching Unstructured Product Offers to Structured Product Specifications

Anitha Kannan
Microsoft Research
ankannan@microsoft.com

Inmar E. Givoni[*]
University of Toronto
inmar@psi.utoronto.ca

Rakesh Agrawal
Microsoft Research
rakesha@microsoft.com

Ariel Fuxman
Microsoft Research
arielf@microsoft.com

ABSTRACT

An e-commerce catalog is typically comprised of specifications for millions of products. The search engine receives millions of sales offers from thousands of independent merchants that must be matched to the right products. This problem is hard for several reasons. First, unique identifiers are absent in most offers. Second, although the product specifications are well structured, offers are described in the form of free text. Third, offers mention the values of the attributes without providing the corresponding attribute names. Fourth, values of a large number of attributes are often missing from the offer description. Finally, offers may also contain words other than attribute names and values.

We present an automated technique for matching unstructured offers to structured product descriptions. A novel aspect of our approach is the semantic parsing of offer descriptions using dictionaries built from the structured catalog. Another novelty is that the matching function we learn factors in not only matches but also mismatches of attribute values as well as the missing attribute values. Our approach has been implemented in an experimental search engine and is used to match all the offers received by Bing shopping to the Bing product catalog on a daily basis. We present extensive experimental results from this implementation that demonstrate the effectiveness of the proposed approach.

1. INTRODUCTION

With the increasing widespread use of the Internet, there has been tremendous growth in the amount of commerce conducted over the Web. A recent Comscore study [1] estimates that the yearly retail e-commerce sales in the U.S. alone has topped \$100 Billion. Nearly seven out of ten consumers said that the Internet has become important in providing them with information to help them make buying decisions. More than 70% of consumers said they are likely to shop online before making an offline purchase.

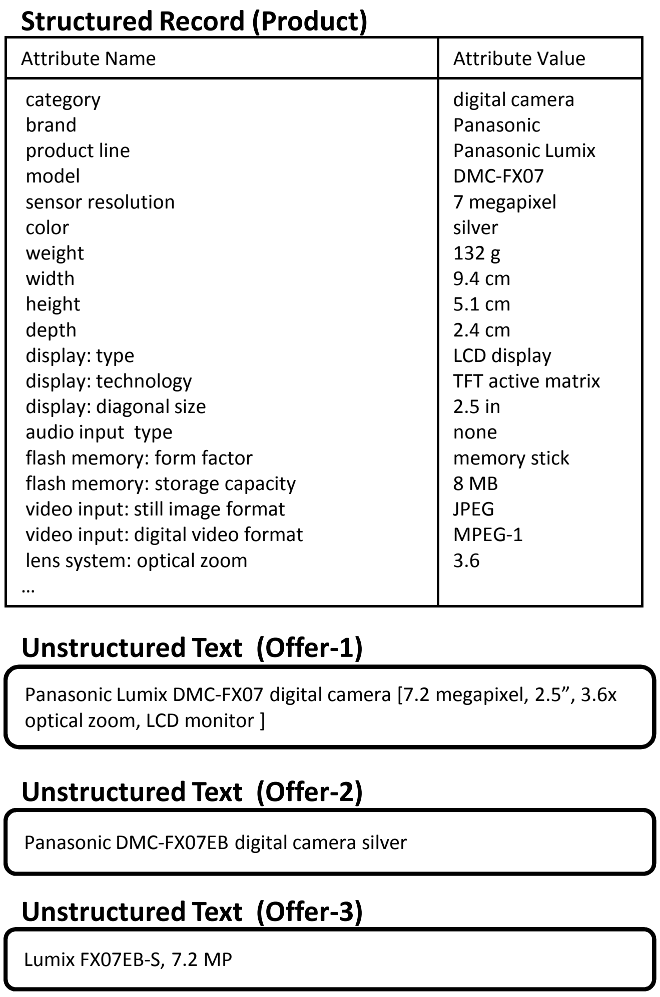
A comprehensive product catalog is a prerequisite for the effectiveness of an e-commerce search service. Such a catalog at web-scale will contain information about every product as well as sales offers from various merchants. For instance, the Bing Shopping catalog (shopping.bing.com) has information on more than five million products and more than ten million offers from upwards of tens of thousands of merchants. The product information consists of various attributes and their corresponding values, stored in a structured record comprised of attribute ⟨name, value⟩ pairs. Many products do not have universally agreed unique identifiers. The product information is obtained from multiple product aggregators (*e.g.,* CNET, PriceGrabber), each of them having only partial but different information. Consequently, the catalog can have multiple data records, each somewhat different from the other, corresponding to the same product.

Similarly, offers come from multiple merchants (*e.g.,* buy.com, gadgettown.com). Generally, there is very little structure in the offers. Typically, an offer consists of a textual description of the product for which the offer is being made. Embedded in the description are some attribute values and sometimes attribute names along with other terms, which the merchant presumes might be sufficient for the offer to be matched to the intended product. Different merchants often use different names for the same attribute. Many offers have no identifier that could be used for matching the offer to the corresponding product. The matching is currently done using rules written by experts – a costly, error-prone, and brittle process. Consequently, many offers are matched incorrectly and millions of offers go unmatched.

Fig. 1 shows part of the structured record for Panasonic DMC-FX07 digital camera as well as three merchant offers for this product as they appear in the Bing Shopping catalog. We make the following observations:

- While Offer-1 is the most detailed one shown, it still contains only a small part of the information in the structured record. The phrase 'Panasonic Lumix' indicates both brand (Panasonic) as well as product line (Panasonic Lumix). Some of the attribute values only match approximately (7.2 megapixel vs. 7 megapixel, LCD monitor vs. LCD display). The only attribute name present in the offer is optical zoom (called 'lens system: optical zoom' in the structured record). The

[*]Work done while author was an intern at Microsoft Research



Structured Record (Product)

Attribute Name	Attribute Value
category	digital camera
brand	Panasonic
product line	Panasonic Lumix
model	DMC-FX07
sensor resolution	7 megapixel
color	silver
weight	132 g
width	9.4 cm
height	5.1 cm
depth	2.4 cm
display: type	LCD display
display: technology	TFT active matrix
display: diagonal size	2.5 in
audio input type	none
flash memory: form factor	memory stick
flash memory: storage capacity	8 MB
video input: still image format	JPEG
video input: digital video format	MPEG-1
lens system: optical zoom	3.6
…	

Unstructured Text (Offer-1)

Panasonic Lumix DMC-FX07 digital camera [7.2 megapixel, 2.5”, 3.6x optical zoom, LCD monitor]

Unstructured Text (Offer-2)

Panasonic DMC-FX07EB digital camera silver

Unstructured Text (Offer-3)

Lumix FX07EB-S, 7.2 MP

Figure 1: Structured product record for ‘Panasonic DMC-FX07 digital camera’ and textual descriptions from three matching offers.

corresponding values for this attribute are 3.6x vs. 3.6.

- Information provided in Offer-2 is largely a subset of what is provided in Offer-1. This offer provides the values of category and brand, but the value of the model has an extra suffix. It additionally provides the value of the color attribute.
- Offer-3 is even more interesting. It provides part of the value of the product line (Lumix) and a somewhat different value for sensor resolution (7.2 MP vs. 7 megapixel) as well as model (FX07EB-S vs. DMC-FX07). It neither provides category nor brand information.
- With respect to Offer-3, note further that Panasonic also makes other 7.2 megapixel Lumix digital cameras (*e.g.,* DMC-TZ3K, DMC-LZ6, and DMC-FX12). Moreover, there is also a field controller product with model number FX07.

Clearly, we have on our hands a hard problem of matching unstructured textual descriptions of products to structured records for which it is desirable to have algorithmic solution.

1.1 Problem Description and Highlights of the Solution

We have a large database of product specifications. Each product specification (which we shall interchangeably call ‘product’) consists of a set of attribute ⟨name, value⟩ pairs and is represented in the database as a structured record. Some of the attributes can be numeric, while the others can be categorical. The unstructured offer descriptions (which we shall call ‘offer’, for short) are comprised of free text. The text has embedded in it some of the values and possibly some attribute names corresponding to one of the products. The text may also contain additional words. The attribute names and values in the text may not precisely match those found in the database. The text does not contain an identifier that uniquely identifies the corresponding product. Different textual descriptions may be provided for the same product. An offer may match more than one product as only partial descriptions are provided in the offers and because the same real-world product might have multiple representations in the product database.

Our goal is to enable automated matching of the offers to corresponding products. Highlights of our proposed solution include:

1. Developing semantic understanding of the offers by leveraging structured information in the database. Specifically, we identify and assign attribute names to the values present in the offers. This semantic parsing serves to identify the product the offer corresponds to.
2. Learning a matching function that finds the product which has the largest probability of match to the given offer. This function is designed to have the following properties:
   - It takes into account matches as well as mismatches in attribute values between offer-product pairs.
   - It differentiates between missing attribute values and mismatch of attribute values.
   - It infers the relative importance of different attributes in the matching.
3. Built-in strategies for the solution to work at web scale. These strategies include
   - Avoiding domain-specific features in the matching system.
   - Reducing the candidate set of products that can potentially match a given set of offers.

To evaluate the quality of the proposed solution, we perform large scale experiments using the Bing product catalog and merchant offers. The precision and recall values obtained from these experiments demonstrate the effectiveness of the solution. Our approach has been implemented in a working search engine and is used to match all the offers received by Bing shopping to the Bing product catalog on a daily basis.

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3, we present our approach. The data-sets and metrics we use for evaluating the performance of the algorithm are given in Section 4. We also present experimental results in this section. We conclude with a summary and directions for future work in Section 5.

Algorithm 1 Off-line Training

```
**Input:**
$\mathcal{U} = \{u_1 \dots u_N\}$ - a set of offers
$\mathcal{S} = \{s_1 \dots s_M\}$ - a set of structured product descriptions, $M >> N$
$\mathcal{M} = \{\langle u_i, s_j \rangle_i\}_{i=1}^N, (u_i \in \mathcal{U}, s_j \in \mathcal{S})$ - pairs of correctly matched records, one for every $u_i$.
$\mathcal{N} = \{\langle u_i, s_k \rangle_i\}_{i=1}^N$ - similarly, pairs of mismatched records.
**Output:**
$\mathcal{D}$ - dictionaries,
$\mathbf{w}$ - algorithm parameters
**Preprocess:**
$\mathcal{D} \Leftarrow$ CreateAttributeDictionaries($\mathcal{S}$) - build dictionaries of attributes and their values using $\mathcal{S}$
**Train:**
**for all** $u \in \mathcal{U}$ **do**
    $u \Leftarrow$ SemanticParsing($u, \mathcal{D}$) - *Extract plausible parses* (Sec. 3.2)
**end for**
**for all** pairs $\in \mathcal{M}$ and pairs $\in \mathcal{N}$ **do**
    $\mathbf{f}_i^{\mathcal{M}} \Leftarrow$ ExtractSimFeatures(pair$_i$)
    $\mathbf{f}_j^{\mathcal{N}} \Leftarrow$ ExtractSimFeatures(pair$_j$) - *Construct similarity feature vector for matched and mismatched pairs* (Sec. 3.3)
**end for**
$\mathbf{w} \Leftarrow \arg\max_\omega$ LearnToMatch($\mathcal{F}(\omega, \mathbf{f}), \{\mathbf{f}_i^{\mathcal{M}}\}, \{\mathbf{f}_j^{\mathcal{N}}\}$) - *Train a function that maps feature vectors to match probability,* $\mathcal{F}(\omega, \mathbf{f}) : \mathbf{f} \rightarrow [0, 1]$ (Sec. 3.4)
**Return:** $\mathbf{w}, \mathcal{D}$
```

Algorithm 2 Online Matching

```
**Input:**
$u$ - offer
$\mathcal{S}, \mathcal{D}, \mathbf{w}$
**Output:** $s^*$ - best matching $s \in \mathcal{S}$
$u \Leftarrow$ SemanticParsing($u, \mathcal{D}$) - (Sec. 3.2)
**Blocking:**
$[k_i] \Leftarrow$ Top attributes with largest weights in $\mathbf{w}$
$\mathcal{S}^* \Leftarrow$ Subset of $\mathcal{S}$ with $\cup_i(u.val(k_i) = s.val(k_i)$
**for all** $s_i \in \mathcal{S}^*$ **do**
    $\mathbf{f}_i \Leftarrow$ ExtractSimFeaturs($\langle \hat{u}, s_i \rangle, \mathcal{K}$) - (Sec. 3.3)
    $P(\text{match}(s_i, u)) \Leftarrow \mathcal{F}(\mathbf{w}, \mathbf{f}_i)$ - *Matching score of a pair* (Sec. 3.5)
**end for**
**Return:** $s^* = \arg\max_{s_i} P(\text{match}(u, s_i))$ - *Best Matching score of all pairs* (Sec. 3.5)
```

2. RELATED WORK

The problem of matching records has been studied under various topics including record linkage [2, 3, 4, 5], duplicate detection [6, 7], entity resolution [8, 9, 10], and merge/purge [11]. While our work continues this rich lineage of work, there are distinguishing traits in our setting that call for fresh approaches and techniques. For instance, while the work of Newcombe [4] (later formalized by Fellegi and Sunter in [3]) pioneered the probabilistic approach to matching, their work (and much of the subsequent record linkage literature) tacitly assumes that the data to be matched consists of properly structured records with a well-defined schema. The work on duplicate detection, merge/purge, and entity resolution is also targeted at structured and properly segmented records. At the other end of the spectrum, the work in the natural language processing[12] focuses on the detection of mentions of the same entity in free text. In contrast, in matching offers to products, there are components from both bodies of work: the offers consist of only free text, while the products are properly structured under a given schema.

Much of the prior work has relied on presence of values for all attributes in the data records, and the goal has been to design similarity metric either at the entire record level [13, 14] or at the attribute level that are subsequently combined to measure record level match [15, 16]. This assumption is not valid in our setting. Since offers are free text, their tokens need to be mapped to attributes. However, not all tokens may map to any attribute (*e.g.,* the token 'monitor' in Offer-1 of Fig. 1), and when they do map, they can be ambiguously mapped to multiple attributes (*e.g.,* the token 'Panasonic' in Offer-1 of Fig. 1). So, unlike in previous settings, matching algorithm needs to disambiguate among the multiple possible interpretations of the offers. These problems also arise in the context of understanding queries [17, 18]. In [17], a probabilistic model is introduced to identify the annotation of a query which corresponds to best explanation of that query. In our work, we propose the notion of optimal parse which is defined with respect to a product the offer will be matched against.

While prior work has focused primarily on the computation of weights for value matches and mismatches for the different fields of a record [15, 16], the explicit modeling of missing values has not received much attention. An exception is in [19], wherein a comparison feature vector is augmented to encode presence/absence of values. However, this approach does not explicitly penalize mismatches at the attribute level, and therefore does not leverage a strong signal for matching offers to products.

Specifically in the Commerce domain, Bilenko et al. [20] proposed techniques for clustering merchant offers, but assume that the offers have structured information. The challenges of matching unstructured offers to structured product specifications are not present when clustering structured offers.

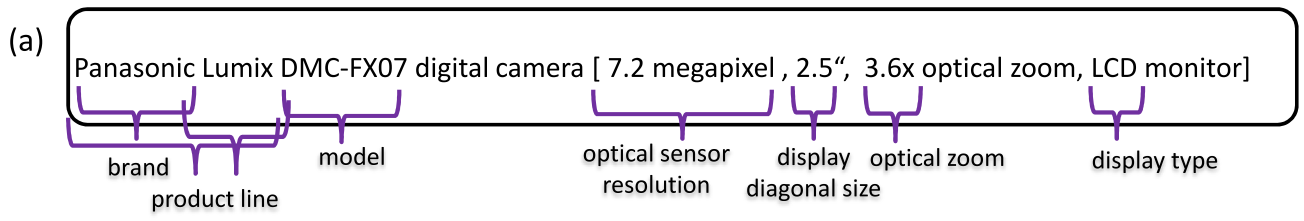
3. METHODOLOGY

3.1 Problem Statement

We have a database $\mathcal{S}$ of product descriptions, represented as structured records. Every structured record $s \in \mathcal{S}$ consists of a set of attribute ⟨name, value⟩ pairs. The attributes can be numeric or categorical. We receive an unstructured offer u as input, which is a concise free-text description that specifies values for a subset of the attributes in $\mathcal{S}$ in an arbitrary manner. The text may also contain additional words. Our objective is to match u to one or more structured records in $\mathcal{S}$. We use the metric of precision and recall for judging the quality of the matching system.

We take a probabilistic approach and find the product $s \in \mathcal{S}$ that has the largest probability of match to the given offer, u. Our matcher is learned in an offline stage (Algorithm 1). For this, we postulate a small training set $\mathcal{U}$ of unstructured offers. Each $u \in \mathcal{U}$ has been matched to one structured record in $\mathcal{S}$ (set $\mathcal{M}$). We also have mismatched records

(a)


Panasonic Lumix DMC-FX07 digital camera [7.2 megapixel , 2.5", 3.6x optical zoom, LCD monitor]
brand
product line
model
optical sensor resolution
display diagonal size
optical zoom
display type

(b)

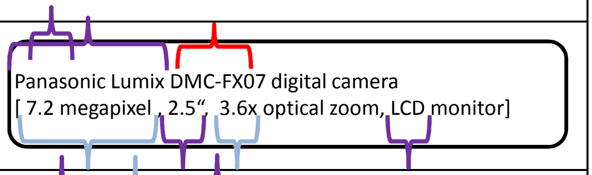
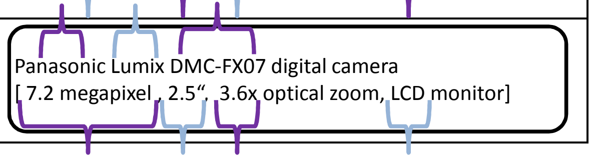
	Attribute Type	Attribute Value	Maximal Parsing of Unstructured Record (a)
s1	**brand** **product line** **model** **display diagonal size** display type	Panasonic Panasonic Lumix DMC-FX05 2.5 in LCD	 Panasonic Lumix DMC-FX07 digital camera [7.2 megapixel , 2.5", 3.6x optical zoom, LCD monitor]
s2	**brand** **model** **optical sensor resolution** optical zoom	Panasonic DMC-FX07 7.2 megapixel 3.6x	 Panasonic Lumix DMC-FX07 digital camera [7.2 megapixel , 2.5", 3.6x optical zoom, LCD monitor]

Figure 2: (a) An offer u (b) Two products $s1$ and $s2$ and the optimal parses of u.

from $\mathcal{S}$, one for every $u \in \mathcal{U}$ (set $\mathcal{N}$).

In the subsequent online stage (Algorithm 2), new offers are matched one at a time. We first do candidate selection, and then choose the best matched product amongst the candidates by applying the learned model.

We next describe the key components required for the two algorithms corresponding to learning the matcher in the offline stage and matching the offers in the online stage:

3.2 Semantic Parsing

Our matching algorithm is based on understanding the semantics in the offer descriptions and using that semantics to aid in matching. Thus, the first step in matching is the semantic parsing step in which the semantics present in the offer is understood. This is operationalized in a three stage process consisting of *tagging* the offer with attributes, identifying *plausible parsings* based on the tags and finally obtaining an *optimal parse*. We describe each of these steps below.

Tagging: The tagging step identifies attribute names present in the offer and associates all strings in the offer that can be assigned to them. Let $\mathcal{A}$ represent all the attributes present in structured data available in product descriptions $\mathcal{S}$. We first build an inverted index $\mathcal{D}$ from $\mathcal{S}$ such that $\mathcal{D}(v)$ returns the attribute name $a \in \mathcal{A}$ associated with string v. Given an offer u, let $\mathcal{Z}_u$ represent the set of all n-grams (up to n = 4) present in u. Then, the tagging step identifies the set of all attribute ⟨name, value⟩ pairs in u:

$$\mathcal{R}_u = \Big\{ \langle a, \{v | v \in \mathcal{Z}_u, \mathcal{D}(v) = a|\}\rangle | a \in \mathcal{A} \Big\} \qquad (1)$$

Fig. 2(a) shows an offer for digital camera, and the output of the tagging step. A portion of the output from this step is { {brand, {'panasonic'}}, {product line, {'lumix', 'panasonic lumix'}}, where brand and product line are the two of the identified attributes, with brand having a single value 'panasonic' and product line having a set of two values {'lumix', 'panasonic lumix'}.

Plausible parse: Given the tagging, a plausible parse of an offer is defined to be a particular combination of all attributes identified in the tagging step such that each attribute is associated with exactly one value. Thus, if each attribute a has k_a values, then there are exactly $\prod_a k_a$ plausible parses in the offer. Typically, k_a is small and thus, only a small number of parses are plausible.

The example in Fig. 2(a) has a single value for six of the seven identified attributes and the 'product line' attribute has two values. Thus, this offer has two plausible parses, one parse in which 'lumix' is the 'product line' and other in which 'panasonic lumix' is the 'product line', while the values do not change for other attributes between parses.

Multiple plausible parses arise because of ambiguities in the data. Therefore, we maintain these plausible parses until the offer is paired with a product which gives rise to the optimal parse of the offer with respect to that product.

Optimal parse: When an offer is paired with a product $\langle u, s\rangle \in \mathcal{M}$ (also for pairs in $\mathcal{N}$), we use the parse of the offer that is best in agreement with the product. By best in agreement we mean, the parse in which the maximum number of attributes agree in their values in u and s. We call this plausible parse as *the optimal parse* of the offer to the product. Note that optimal parse is defined only with respect to the product. Different products can give rise to a different choice of plausible parse to be optimal.

Continuing with our example using Fig.2b, the optimal parse of u corresponding to product $s1$ is the plausible parse with 'Panasonic Lumix' as the product line. When u is paired with $s2$, both plausible parses are optimal since $s2$ does not have product line specified.

3.3 Similarity Feature Vectors

Similarity between an offer and a product is measured in terms of their similarity on the values of the attributes present in them. Since products have a large number of attributes, we choose a subset of these attributes that are present in offers. In particular, using Eqn. 1, we select attributes, $\mathcal{K}$ such that

$$\frac{\sum_{u \in \mathcal{U}} \mathrm{I}[\mathcal{R}_u(k) \neq \emptyset]}{|\mathcal{U}|} > 0 \tag{2}$$

where $\mathcal{R}_u(k)$ represents the values of the attribute k found in $\mathcal{R}_u$ (according to Eqn. 1). $\mathrm{I}[t]$ is the indicator function and the expression $[\mathcal{R}_u(k) \neq \emptyset]$ is defined to be true if $\mathcal{R}_u(k)$ has one valid value for attribute k is found in u.

We would like the similarity function defined over $\mathcal{K}$ to take into account not only the match in values of certain attributes, but also reflect mismatches or missing values in either products or offers. The function should penalize mismatches differently from missing values; In fact, a mismatching value is a stronger indicator of the corresponding offer and product mismatching. In addition, an attribute that is frequently missing reflects its lower importance for matching.

With these design considerations, we define the similarity feature vector as follows: Let $\hat{u}$ represent the optimal parse of offer u with respect to product s. Then, for the pair $\langle \hat{u}, s \rangle$, we compute a similarity feature vector $\mathbf{f}$ by determining similarity levels between $\hat{u}$ and s for each attribute. Let $\hat{u}.val(k)$ and $s.val(k)$ represent the value of some attribute k from $\hat{u}$ and s, respectively. The similarity between $\hat{u}$ and s for attribute k is defined to be:

$$f_j = \begin{cases} 0 & \text{if } \hat{u}.val(k) = \emptyset \text{ OR } s.val(k) = \emptyset \\ (-1)^{\mathrm{I}[\frac{|\hat{u}.val(k) - s.val(k)|}{\max(\hat{u}.val(k), s.val(k))} > \lambda]} & \text{if } k_j \text{ is numeric attribute} \\ (-1)^{\mathrm{I}[\hat{u}.val(k) = s.val(k)]} & \text{otherwise} \end{cases} \tag{3}$$

where I[z] is the indicator function. When either the optimal parse of the offer or the product has a missing value for an attribute, the corresponding feature value is 0, unlike when the values mismatch whence the value is -1. This enables penalizing the matching score differently when $\hat{u}$ or s is missing an attribute value than if they disagree on that attribute.

For categorical attributes, we use binary loss since the offer descriptions typically do not have typographical errors (perhaps due to the fact that they are shown on merchants' websites). However, numeric attributes frequently have imprecise values because of round-off errors (*e.g.* 7 MP vs. 7.2 MP) or difference in conversion factors (1GB = 1000 MB or 1GB =1024 MB). After some experimentation, we set λ to .1 to provide a less sensitive measure of similarity than that of the binary loss. This parameter is held at .1 across all categories. If desired, it can be learned using cross validation. Another possibility, is to set λ to zero, requiring the stringent condition that numeric attribute values should also match exactly.

3.4 Matching Function

We would like a matching function that can provide a probabilistic score of match between an offer and a product so that the best matching product to an offer is the one that has the largest probability. In addition, as the number of attributes in $\mathcal{S}$ is large, and not all attributes are present in the offers, the function needs to automatically infer attributes that are required to be matched, and also learn the relative importance between them.

We find that binary logistic regression conveniently lends itself to satisfy these two criteria. Given some labeled data of good and bad matches, and features that measure similarity between the attributes, it can automatically learn the relative importance between the attributes, and in turn provide a function that measures the match between an offer and product in terms of a probabilistic score. Hence, we use binary logistic regression of the form:

$$\mathcal{F}(w, \mathbf{f}) = P(y = 1|\mathbf{f}, \mathbf{w}) = \frac{1}{1 + \exp\{-(b + \mathbf{f}^T \mathbf{w})\}} \tag{4}$$

The logistic regression learns a mapping from the similarity feature vector $\mathbf{f}$ to a binary label y, through the logistic function. The parameter $\mathbf{w}$ is the weight vector wherein each component w_j measures the relative importance of the feature f_j for predicting the label y.

We have with us all matched and mismatched training pairs, and let $\mathbf{f}_i = [f_{i1}, f_{i2}, \ldots, f_{i|\mathcal{K}|}]$ be the feature vector for pair i. Let $\{\mathbf{F}, \mathbf{Y}\} = \{(\mathbf{f}_1, y_1), \ldots, (\mathbf{f}_N, y_N)\}$ be the set of feature vectors along with their corresponding binary labels. Here, $y_i = 1$ indicates that the ith pair is a match, otherwise $y_i = 0$. Logistic regression maximizes an objective function which is the conditional log-likelihood of the training data $P(\mathbf{Y}|\mathbf{F}, \mathbf{w})$:

$$\arg\max_{\mathbf{w}} \log P(\mathbf{Y}|\mathbf{F}, \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log P(y_i|\mathbf{f}_i, \mathbf{w}), \tag{5}$$

where $P(y_i = 1|\mathbf{f}_i, \mathbf{w})$ is defined by Eq. 4. Note that a feature with positive weight will affect the score by increasing the probability of match for a pair with agreement on the feature, by decreasing the score in the case of a mismatch, and by leaving the score unaffected in the case of a missing value.

3.5 Online Matching

During the online phase, we are given a previously unseen offer u, and the goal is to identify the best matching product $s \in \mathcal{S}$.

The scoring function learned during the offline phase provides the probability of match for a pair $\langle u, s \rangle$. Naively, we can find the best match by pairing u with every $s \in \mathcal{S}$, calculating the pair match score, and choosing the s^* that results in the highest score. However, such naive pairing will cost $O(|\mathcal{S}|)$ operation for each offer.

Instead, building upon the work in record linkage [2, 21] and merge/purge [11], we design a staged blocking strategy. We note that the products are usually categorized into a taxonomy. Therefore, in the first stage, we use a classifier to categorize the given offer into a category node in this taxonomy. This reduces the candidate set to only those products that belong to the offer category.

Further, within the category, we would like to reduce the number of candidate products to match agaisnt the offer. For that, we make the following observation. The goal of the matching process is to match the offer to the product that has the largest matching score. To obtain this large score, a product needs to agree, especially on the values of the attributes that contribute large weights to the matching function. Using this insight, in the second stage, we further reduce the candidate set by identifying those top weighing attributes that can potentially give a matching score of at least θ in Equation 4. To describe in detail, after identifying attributes in the offer using method in Sec. 3.2, we choose attributes in the descending order of weights until the following condition is satisfied:

$$\sum_j f_j w_j \geq \log \frac{\theta}{1 - \theta} - b \tag{6}$$

where weights are ordered so that $w_j >= w_{j+1} \forall j$.

This equation can be derived in straightforward way from Equation. 4 by rearranging terms and taking the log. Here, θ corresponds $P(y = 1|\mathbf{f}, \mathbf{w})$. This set of attributes $\{k_j\}$ are then used to retrieve products such that the retrieved products match on value of at lease one of the attributes in $\{k_j\}$. The union of all these products becomes the candidate set of products. Note that, this candidate set is a superset of products that can potentially match to offer since we consider all products that have at least one matching attribute value, within this attribute set. In our experiments, we use $\theta = 0.5$ as it is mid point on the probability scale.

4. EVALUATION

The matching system described in the previous section has been implemented in a working experimental search engine and is used to match all the offers received by Bing shopping to the Bing product catalog on a daily basis. In this section, we present performance results from experiments using this implementation.

4.1 Algorithms used in the Study

Variants of our Algorithm

We define the following variants of our matching algorithm in order to study its various characteristics:

1. **Equal Weights (EW):** This is the simplest version where the number of agreeing attributes between the offer u and the product s is used as the predictor for matching. The product s having the largest number of attributes in agreement with u is taken to be the best match.
2. **Learned Weights (LW):** In this version, we learn the relative importance between the attributes, but we treat missing and mismatched attributes as equals. For attribute k, the value of feature $\mathbf{f}_k$ is -1 when either the values are missing or when the values are mismatched.
3. **LW with distinction between Mismatched and Missing (LWMM):** Here, relative importance between attributes is learned taking into account whether attribute values are mismatched or missing. For attribute k, the value of feature $\mathbf{f}_j$ for missing values is 0, and it is -1 for mismatched values. This version implements the full functionality of Algorithm 1.

Baseline Algorithms

In the absence of an algorithm, directly applicable to our problem formulation, we define two baselines, COSINE and TFIDF for comparison. They are inspired by the work in record linkage for measuring token-level similarity ([13, 14, 20]).

- **COSINE:** This baseline uses the cosine similarity as the measure of agreement between the offer and product. Similarity is measured between the frequency distribution of tokens in u and s [20]. As s is structured, it is first converted into a string by concatenating the content of the record. The s having the largest cosine similarity with u is taken as the best match.
- **TFIDF:** This baseline uses the tf-idf weighted cosine similarity as the measure of agreement between the offer and product. Each token is associated with the tf-idf score defined by log(TF(token)+1) log(IDF(token)) [14]. Here, TF(token) is the frequency of the token in the offer/product and IDF(token) is its corresponding inverse document frequency; IDF(token) is computed across all the products in the category. Similarity is measured between the normalized tf-idf score of tokens in u and s.

Note that while COSINE treats all tokens as equally useful, TFIDF will weigh them inversely to their popularity. Thus, a token such as '40d' (corresponding to model number of a canon eos 40d digital camera) will have higher tf-idf score than 'digital camera' that is ubiquitous in digital camera category. Typically, tokens that are unique such as model numbers and brands are those that are useful in matching. Since TFIDF can choose such unique tokens, it can serve as a good baseline to our approach.

4.2 Performance Metrics

For evaluation purposes, we have access to a test set of offers, $u \in \mathcal{T}$. We also know the correctly matched product s^* for every u. The matcher has no knowledge about the matching product, but instead predicts the best matched product $\tilde{s}$ with probabilistic score $\gamma_{u,\tilde{s}}$. By best matched we mean that there is no other s that can match u with a higher score. Thus, instead of a standard classification task, we are evaluating performance on the harder task of the matcher finding the best matching s for every offer u. We require the match score to be at least some given level $\theta \in [0, 1]$ before calling out a match.

We define precision and recall at threshold level θ as:

$$\text{Precision}(\theta) = \frac{\sum_{u \in \mathcal{U}} I[(\gamma_{u,\tilde{s}} > \theta) \text{ AND } (s^* = \tilde{s})]}{\sum_{u \in \mathcal{U}} I[\gamma_{u,\tilde{s}} > \theta]} \tag{7}$$

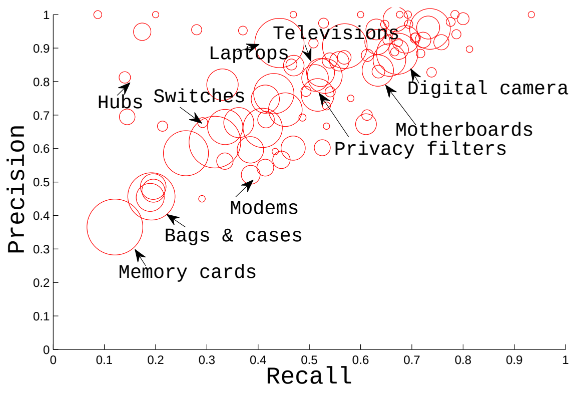
$$\text{Recall}(\theta) = \frac{\sum_{u \in \mathcal{U}} I[(\gamma_{u,\tilde{s}} > \theta) \text{ AND } (s^* = \tilde{s})]}{|\mathcal{T}|}, \tag{8}$$

where I[z] is the indicator function. Unless stated otherwise, θ is set to 0.5. We also combine precision and recall values into the well-known F-measure, defined as:

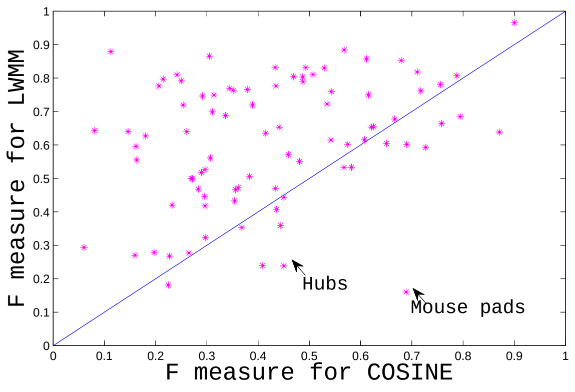
$$\text{F-measure}(\theta) = \frac{2 * \text{Precision}(\theta)\text{Recall}(\theta)}{\text{Precision}(\theta) + \text{Recall}(\theta)}. \tag{9}$$

4.3 Data Sets

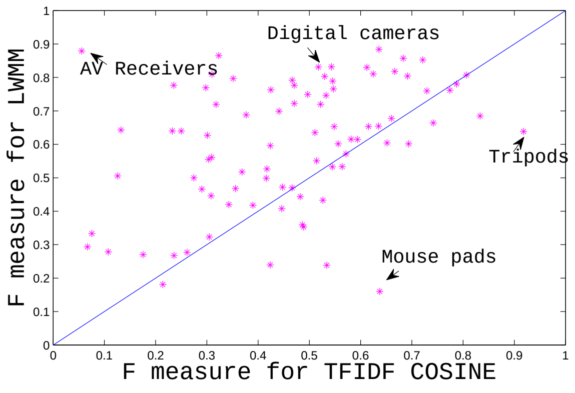
We use a subset of the Bing Shopping catalog in our evaluation. The products belong to 87 categories related to electronics (*e.g.,* televisions, mp3 players), computing (*e.g.,* desktop computers, laptops) and cameras and accessories (*e.g.,* digital cameras, camera accessories). We had a labeled set of 40,000 offers from these categories, each labeled with the corresponding matched products. There were on average 460 offers/category; the smallest category had 50 offers. For each category, we randomly sampled 100 offers (20 if the number of offers is less than 200) and used them for training the matcher. We used whatever offers were left within a category as the test set for that category. Thus, the training set for each category had at most 100 samples. The test set size varied from category to category as we made use of all the available samples, but in no case the test set was smaller than 30 samples. Results are shown using 5-fold cross validation.



(a) Performance of LWMM



(b) LWMM vs. COSINE



(c) LWMM vs. TF-IDF

Figure 3: LWMM Performance

Offer Description	Products
imation 4gb nano usb 2.0 flash drive - 4gb - usb - external	imation nano flash drive usb flash drive - 4 gb imation 2gb usb 2.0 clip flash drive

Table 1: Sample offer and two products from the memory cards category.

Note that other than using a separate training set for each category, we do not use any category-specific feature and the same code is used for different categories. We use small training set (100 offers per category), which can be curated easily from successful matches (*e.g.,* high click throughs) in a running system. We also do not have any parameters that needs to be tuned. These characteristics are critical for a solution to work at web scale.

4.4 Experiment 1: LWMM Performance

We start by presenting precision and recall values for different categories. Fig. 3(a) shows the scatter plot of precision-recall values that the LWMM algorithm exhibits. Each circle corresponds to a category and the area of the circle is proportional to the test set size. We have labeled some of the categories. The macro average precision is 80% while the macro average recall is 50%.

By way of comparison, Fig. 3(b) (resp. Fig. 3(c)) gives the ratio of the F-measure achieved by LWMM to the F-measure achieved by COSINE (resp. TFIDF). The macro average precision and recall for COSINE are 50% and 37%, respectively. The corresponding numbers for TFIDF are 54% and 42%.

We observe that LWMM has high performance over a range of categories and performs better than COSINE as well as TFIDF. The next three subsections present further findings.

Difference in performance of LWMM over various categories

We see in Fig. 3(a) that LWMM has very high performance on categories such as digital cameras and televisions, but the performance becomes lower on categories such as bags & cases and memory cards. This difference can be understood by examining how complete the information is for the products in the corresponding categories.

As described in Section. 3.2, an important component of our matching system is semantic parsing which makes use of attribute dictionaries compiled from attribute data for the products. This means that the quality of the attribute directories is dependent upon the quality of the product data. For the categories such as digital cameras and television the product data is nearly complete that enables high quality semantic parsing leading to LWMM's high performance.

In comparison, note that memory cards category has low recall and precision. Table. 1 shows an offer and two products from this category. The distinguishing attribute for correct matching in this category is the capacity of the memory card. However, in our product catalog, the value for this attribute is missing in all but two products (out of 7500 products).

Improvement in LWMM over COSINE

In Fig. 3(b), COSINE's F-measure is much lower than LWMM for most of the categories. By examining instances where the matches differ, we found that the main reason is that there are many tokens in the offers that are not distinguishing attribute names or values, but generic terms. Since, the cosine similarity weighs all tokens equally, it is unable to perform effective matching. This demonstrates the importance of performing semantic parsing of the offer description and using only the relevant tokens (attribute names and values) in the matching.

Improvement in LWMM over TFIDF

TFIDF weighs the relative importance of the tokens, but only as measured by the frequency of their presence in the product collection. It is not cognizant of what tokens are semantically important. Thus, the token 'canon' gets down-weighted because there are very many canon cameras. However, 'canon' as the brand is a very good indicator of a product identity. Hence, it is important to identify the attributes that are present in the offers and their relative importance for better matching.

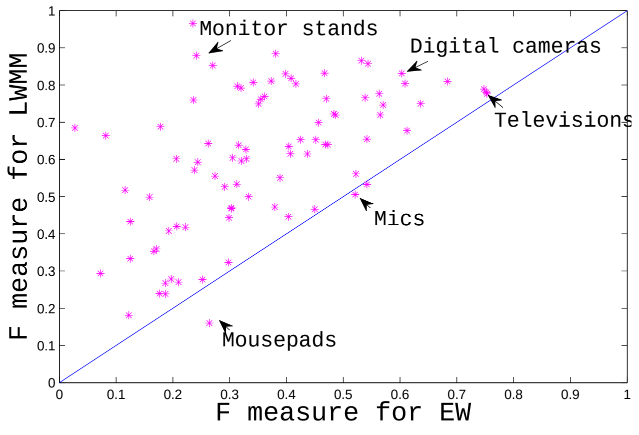
We investigated further the categories where the COSINE and TFIDF do better than LWMM. These categories include hubs, mouse pads and tripods. We found that the primary reason is that these categories have very few structured attributes. For example, tripods have only two attributes: brand and height. Additionally, many tripods have the same value for these attributes. Our matching function thus finds that it is matching too many products for an offer and refuses to return a product. LWMM thus ends up with 100% precision and 8% recall for tripods. COSINE and TFIDF, on the other hand, take chance on the name of the product and have higher recall (and a higher score for F-measure). Similar observations hold for hubs and mouse pads.

Two points are in order here. First, our deployed system is designed to be conservative; it is willing to sacrifice recall for precision. More importantly, this analysis points to selectively adopting a hybrid strategy. When a category is impoverished with respect to the number of the attributes required to explain its products, a hybrid scheme that uses both structured information as well as token representation should be used.

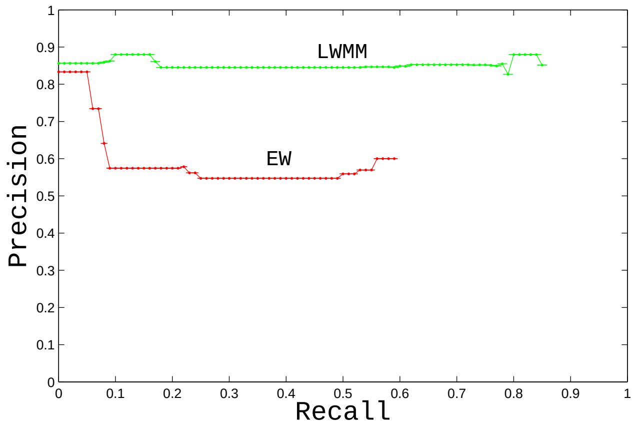
4.5 Experiment 2: Importance of learning weights

Fig. 4(a) shows the scatter plot of F-measure of LWMM over EW's F-measure. Clearly, learning weights makes matching better. We notice that the gains are much larger for some categories than others. To understand this difference, we drill down on two categories – digital cameras and televisions.

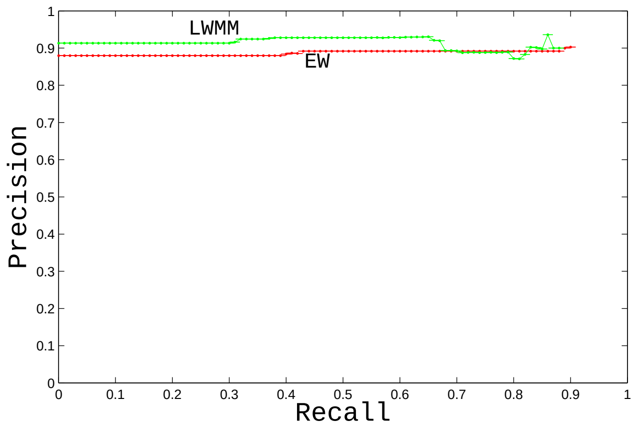
Fig. 4(b) shows the precision-recall values for digital cameras. For this category, there were seven attributes present in at least one offer during the offline training phase. These attributes were brand, model, product line, color, resolution, optical zoom, viewfinder type and video input type). **EW** weighs these attributes equally. At low recall, **EW** insists on all key attributes to agree on their values, and hence the precision becomes high. However, as we increase recall by reducing the number of agreements in attributes, precision drops. It is as expected since certain combinations of attributes provide spurious matches (*eg.* agreement on color and resolution of a camera). On the other hand, by



(a) LWMM vs. EW



(b) LWMM vs. EW (Digital Cameras)



(c) LWMM vs. EW (Televisions)

Figure 4: Importance of learning weights

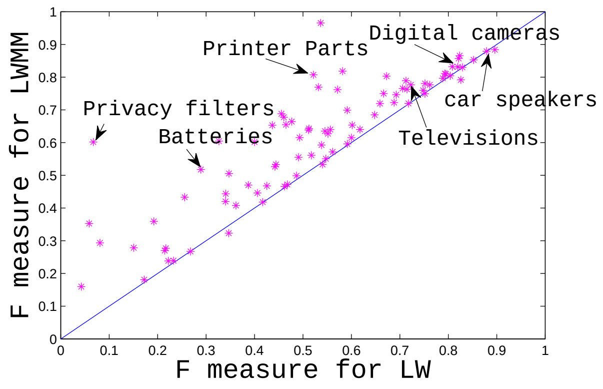


Figure 5: Treating mismatching attributes differently from missing attributes

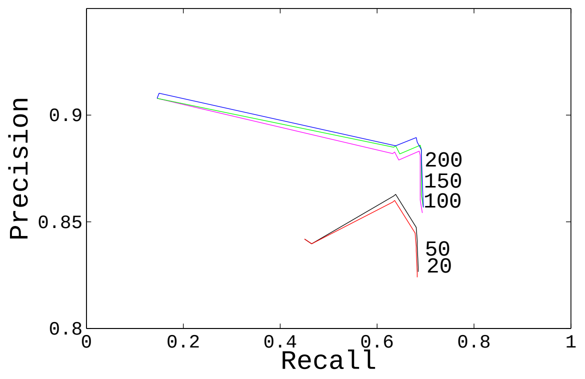


Figure 6: Effect of training set size on test set performance

learning the relative importance of the attributes, **LWMM** matches a larger fraction of offers without sacrificing much of precision.

The precision-recall values for televisions are shown in Fig. 4(c). For this category, only five attributes (brand, model, product line, diagonal screen size, projection display technology) are present in the offers, during the offline training phase. We found that these attributes were often present in the data completely and the values were generally correct. Hence, the performance of **EW** is quite high and it performs almost as well as its learned counterpart, **LWMM**.

4.6 Experiment 3: Importance of treating mismatching attributes differently from missing attributes

Fig. 5 shows the scatter plot comparing the F-measures between LWMM and LW. We can see that there is a gain in F-measure for all categories, when we treat missing attributes differently from mismatched values. The gain is less pronounced for high economic value categories such as digital cameras than for low economic value categories such as batteries. The reason is that in the case of the former, the merchants have the incentive to provide better offer descriptions containing all the necessary attributes needed for matching. Hence, missing attributes becomes less of an issue.

4.7 Experiment 4: Scalability

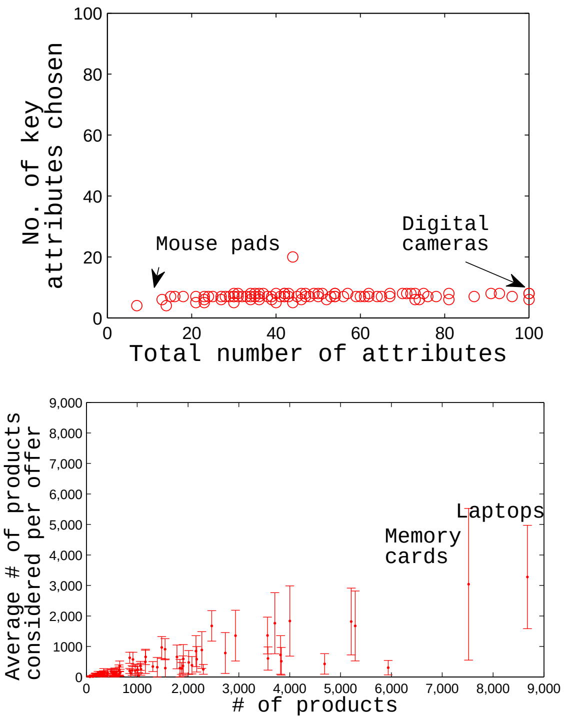


Figure 7: Scalability contributors during online matching

In this section, we give data on factors that contributed significantly for our solution to work at web scale.

Off-Line Training Phase

In addition to the insistence on not having any custom-designed features, we limit the training sets to small sizes. Fig. 6 plots the macro average precision and recall values for average number of samples per category in the training set. We see that the number of samples needed is small. It is because the matcher has to learn only a small number of parameters as it is trained only on attributes present in the offers. In our evaluation, for many categories having a small number of matched offers, we used only 20 samples for training. For larger categories, we used 100.

Online Matching Phase

As discussed in Section. 3.3, we consider only those attributes that are present in offers. Fig. 7(a) shows that such selection on attributes prunes away a large number of attributes. Every point in this scatter plot corresponds to a category and we can see that only 7-8 attributes are required for matching even though the number of attributes generally ranged from 40 to 50.

Additionally, we do staged blocking at the time of matching. In the first stage, offers are classified into categories, and then they are matched to products within that category. This substantially reduces the number of products to compare against. Subsequently, as described in Section. 3.5 only a viable subset of products is selected to be matched against the given offer. Fig. 7(b) shows the number of candidates considered for each category. Each stem corresponds to a particular category; the length of the stem is one standard deviation variation of the average of number of candidates across the offers in that category. We can see that the candidate set size is much smaller than the number of products in most of the categories.

5. CONCLUSIONS AND FUTURE WORK

We studied the problem of matching unstructured textual description to structured data records that arises in the context of matching sales offers to product specifications in the e-commerce websites. The product specifications include both numeric and categorical data. The key distinguishing characteristics of our solution are:

- Semantic understanding of offer descriptions using automatically built attribute dictionaries from structured product specifications contained in the product catalog
- A matching function that considers not only matches but also mismatches of attribute values and the missing attribute values. The function learns the relative importance between the attributes
- Avoidance of domain-specific features and use of staged blocking strategies for the solution to work at web scale

We performed extensive experiments using Bing Shopping catalog to understand the performance characteristics of the proposed solution. The experimental results show that the proposed approach scores high on F-measure and consistently beats baseline approaches for product categories that have reasonably rich attribute structure and good data. They also point to the desirability of hybrid solutions that additionally make use of classical text matching techniques for attribute impoverished product categories. The methodology we employed for analyzing the experimental results might also be of interest to those building and analyzing web scale systems.

There are a number of potential future research directions. Currently, the training data we used has positive examples (matched pairs), and we obtained negative examples by randomly pairing offers with non-matched products. Ideally, we would like negative examples that are similar to positive examples, yet mismatched so that the learning algorithm can tease out subtle nuances between matching and non-matching pairs. One possibility would be to obtain such negative examples by using active learning, taking cues from [7]. Another research direction is to learn to infer inter-attribute correlations that can be helpful when the number of key attributes is large. Finally, it will be interesting to apply the proposed techniques to other application domains (eg. Travel, Health) where there is preponderance need for matching unstructured text to structured data records.

6. REFERENCES

[1] G. Fulgoni, "State of the U.S. Retail Economy in Q2 2009," Comscore, Tech. Rep., August 20 2009.

[2] W. E. Winkler, "Overview of record linkage and current research directions," Bureau of the Census, Tech. Rep., 2006.

[3] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[4] H. B. Newcombe, M. J. Kennedy, S. J. Axford, and A. P. James, "Automatic linkage of vital records." *Science*, vol. 130, pp. 954–959, October 1959.

[5] P. Ravikumar and W. W. Cohen, "A hierarchical graphical model for record linkage," in *UAI*, 2004.

[6] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 1, pp. 1–16, 2007.

[7] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in *KDD*, 2002, pp. 269–278.

[8] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant, "Auditing compliance with a hippocratic database," in *VLDB*, 2004, pp. 516–527.

[9] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: a generic approach to entity resolution," *The VLDB Journal*, vol. 18, no. 1, pp. 255–276, 2009.

[10] S. Singh, K. Schultz, and A. McCallum, "Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs," in *ECML-PKDD*, 2009, pp. 414–429.

[11] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *SIGMOD*, 1995, pp. 127–138.

[12] R. Mitkov, *Anaphora Resolution*. Longman, 2002.

[13] A. Monge and C. Elkan, "The field-matching problem: algorithm and application," in *KDD*, 1996.

[14] W. Cohen, "Integration of heterogeneous databases without common domains using queries based on textual similarity," in *SIGMOD*, 1998, pp. 202–212.

[15] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha, "Efficient data reconciliation," *Information Sciences*, vol. 137, no. 1-4, pp. 1–15, 2001.

[16] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 16–23, 2003.

[17] N. Sarkas, S. Paparizos, and P. Tsaparas, "Structured annotations of web queries," in *SIGMOD*, 2010, pp. 771–782.

[18] K. Q. Pu and X. Yu, "Keyword query cleaning," in *PVLDB*, 2008, pp. 909–920.

[19] J. N. S. D'Andrea Du Bois, "A solution to the problem of linking multivariate documents," *Journal of the American Statistical Association*, vol. 64, no. 325, pp. 163–174, March 1969.

[20] M. Bilenko, S. Basu, and M. Sahami, "Adaptive product normalization: Using online learning for record linkage in comparison shopping," in *ICDM*, 2005, pp. 58–65.

[21] A. Mccallum, K. Nigam, and H. L. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *In Knowledge Discovery and Data Mining*, 2000, pp. 169–178.