

Semi-Supervised Learning to Extract Attribute-Value Pairs from Product Descriptions on the Web

Katharina Probst, Rayid Ghani, Marko Krema, Andy Fano
Accenture Technology Labs, Chicago, IL, USA

Yan Liu
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We describe an approach to extract attribute-value pairs from product descriptions on the Web. The goal is to augment product databases by representing each product as a set of such attribute-value pairs. Such a representation is useful for a variety of tasks where treating the product as a set of attribute-value pairs is more useful than as an atomic entity. Examples include product recommendations, comparison of single products or complete offerings, and demand forecasting. We formulate the extraction as a classification problem and use Naïve Bayes combined with a multi-view semi-supervised algorithm (co-EM). The extraction system requires very little initial user supervision: using unlabeled data, it automatically extracts an initial seed list that serves as training data for the classification algorithm. In addition to the automatically extracted training data, the co-EM algorithm uses the unlabeled data to extract product attributes and values. Finally, the extracted attributes and values are linked to form pairs using dependency information and co-location scores. We present promising results on Web product descriptions in two categories of sporting goods products.

1 Introduction

Retailers have been collecting a growing amount of sales data that contains quite detailed information about customers and related transactions; in contrast, the information about the actual products that were sold is often sparse and limited. After discussions with many large retailers, we found that most retailers treat their products as atomic entities with very few related attributes (typically brand, size, or color). At the same time, they offer their products to customers (on a web site) that describes each product in detail, specifying the product’s physical attributes, but typically in natural language, making it difficult to be used directly in many applications. The task we tackle in this paper requires a system that can process product descriptions and extract relevant attributes and values, and then form pairs by associating values with the attributes they describe. This can be accomplished by different means depending on the amount and type of information available. In this paper, we assume a scenario where a list of textual product descriptions can be scraped from a company’s web site. The product descriptions are assumed to be ‘unstructured’ natural language text. The system described in this paper is able to extract attribute-value pairs from product descriptions with minimal human supervision. We describe the components of our system and show experimental results on a web catalog of sporting goods products.

2 Related Work

There has been a lot of research on extracting information from text documents on the Web but we are not aware of any system that addresses the same task as we are addressing in this paper. A related task that has received attention recently is that of extracting product features and their polarity from online user reviews.

Liu et al. [7] focus on extracting relevant product attributes, such as ‘focus’ in the domain of digital cameras. These attributes are extracted by use of a rule miner, and are restricted to noun phrases. The system then extracts polarized descriptors, e.g., ‘good’, ‘too small’, etc. Popescu and Etzioni [10] describe a similar approach: they first extract noun phrases as candidate attributes, and then compute the pointwise mutual information between the noun phrases and salient context patterns (such as ‘*scanner has*’). Similarly to Liu et al. [7], the extraction phase is followed by an opinion word extraction and polarity detection phase. Our work is similar in that a product is expressed as a vector of attributes. The difference is that our work focuses not only on attributes, but also on extracting values, and on associating the extracted attributes with the extracted values. Also, the attributes that are extracted from user reviews are often different (and described differently) than the attributes of the products that retailers would mention. For example, a review might mention ‘photo quality’ as an attribute but specifications of cameras would tend to use megapixels or the lens manufacturer in the specifications.

Information extraction with the goal of filling templates, e.g., [5, 9], is related to the approach in this paper in that we extract certain parts of the text as relevant facts. It however also differs from such tasks in several ways, notably because we do not have a definitive list of ‘template slots’ available. Recent work in bootstrapping for information extraction using semi-supervised learning has focused on the task of named entity extraction [4, 2, 3], which is related to part of the work presented here (classifying the words/phrase as attributes or values or as neither).

3 Overview of the Attribute Extraction System

Our system consists of five modules: 1) Data Collection, 2) Seed Generation, 3) Attribute-Value Entity Extraction, 4) Attribute-Value Pair Relationship Extraction, and 5) User Interaction. The modular design allows us to break the problem into smaller steps, each of which can be addressed by various approaches. In this paper, we have chosen one specific approach for each phase. We only focus on tasks 1-4 in this paper, where task 5 is largely future work that we however consider very important.

4 Data

The data required for extracting product attributes and values can come from a variety of sources such as an internal product database or from the retailer website. We crawled the web site of a sporting goods retailer (www.dickssportinggoods.com), concentrating on the domains of tennis and football. Sporting goods is an interesting and relatively challenging domain because unlike electronics, the attributes are not easy and straightforward to detect. For example, a camera has a relatively well-defined list of attributes (*resolution, zoom, memory-type*, etc.). In contrast, a baseball bat would have some typical attributes such as brand, length, material as well as others that might be harder to identify as attributes and values (*aerodynamic construction, curved hitting surface*, etc.)

The scraping process resulted in a set of product descriptions where each product is described by a list of phrases, which we use as training data. Some examples of entries in these lists are *1 tape cutter*, *4 rolls of white athletic tape*, *Cutout midfoot*, *Extended Torsion bar*, *Synthetic leather upper*, *Audio/Video Input Jack*, *Play Dry technology offers moisture management and wicking properties*, *Vulcanized latex outsole construction is lightweight and flexible*

It can be seen from these examples that the entries are not often full sentences. This makes the extraction task more difficult, because most of the phrases contain a number of modifiers. There is often no definitive answer as to what the extracted attribute-value pair should be, even for humans inspecting the data. For instance, should the system extract *cutter* as an attribute with two separate values, *1* and *tape*, or should it rather extract *tape cutter* as an attribute and *1* as a value? To answer this question, it is important to keep in mind the goal of the system to express each product as a vector of attribute-value pairs, so as to compare between products. Therefore, it is more important that the system is consistent than which of the valid answers it gives.

5 Pre-Processing

The product descriptions collected by the web crawler are first tagged with parts of speech (POS) using the Brill tagger and stemmed with the Porter stemmer. We also replace all numbers with the unique token *#number#* and all measures (e.g., *liter*, *kg*) by the unique token *#uom#*. Additionally, we compute several correlation scores (Yule's Q statistic, pointwise mutual information, and χ^2) between all pairs of words and recognized one as a phrase if all of its correlation scores exceed certain thresholds.

6 Seed Generation

Once the data is collected and processed, the next step is to provide labeled seeds for the learning algorithms to learn from.

Generic and domain-specific lists as labeled seeds. We use a very small amount of labeled data in the form of generic and domain-specific lists. The generic value lists were easily available on the web and are fairly domain-independent. We use lists of colors, materials, countries, and units of measure. In addition, we use a list of domain-specific (in our case, sports) values and attributes consisting of sports teams (such as *Pittsburgh Steelers*)

These seeds are supplemented by automatically extracted attribute-value seed pairs, as described in the following section. In other words, aside from easily replaceable generic and domain-specific lists, the system works in an unsupervised fashion.

Unsupervised Seed Generation. Our unsupervised seed generation method extracts a small number of attribute-value pairs from the unlabeled data that serve as labeled data for classification. We use correlation scores to find candidates, and make use of POS tags by excluding certain words from being candidates for extraction.

Extracting attribute-value pairs is related to the problem of phrase recognition in that both methods aim at extracting pairs of highly correlated words. There are however differences between the two problems. Consider the following two sets of phrases: *back pockets*, *front pockets*, *zip pockets* as compared to *Pittsburgh Steelers*, *Chicago Bears*. The first list contains an example of an attribute with several possible values. The second list contains phrases that are not attribute-value pairs. The biggest difference between

the two lists is that attributes generally have more than one possible value, as in the above example. We exploit this observation to automatically extract high-quality seeds by defining a modified mutual information metric as follows.

We consider all bigrams $w_i w_{i+1}$ as candidates for pairs, where w_i is a candidate value, and w_{i+1} is a candidate attribute. Although the modifying value does not always occur (directly) before its attribute, this heuristic allows us to extract seeds with high precision. Suppose word w (in position $i + 1$) occurs with n unique words $w_{1...n}$ in position i . We rank the words $w_{1...n}$ by their conditional probability $p(w_j|w), w_j \in w_{1...n}$, where the word w_j with the highest conditional probability is ranked highest.

The words w_j that have the highest conditional probability are candidates for values for the candidate attribute w . Clearly, however, not all words are good candidate attributes. We observed that attributes generally have more than one value and typically do not occur with a wide range of words. For example, frequent words such as *the* occur with many different words. This is indicated by their conditional probability mass being distributed over a large number of words. We are interested in cases where few words account for a high proportion of the probability mass. For example, both *Steelers* and *on* will not be good candidates for being attributes. *Steelers* only occurs after *Pittsburgh* so all of the conditional probability mass will be distributed on one value whereas *on* occurs with many words with the mass distributed over too many values. This goal can be accomplished in two phases: in the first phase, we retain enough words w_j to account for a part $z, 0 < z < 1$, of the conditional probability mass $\sum_{j=1}^k p(w_j|w)$. In the experiments reported here, z was set to 0.5.

In the second phase, we compute the *cumulative* modified mutual information for all candidate attribute-value pairs. We again consider the perspective of the candidate attribute. If there are a few words that together have a high mutual information with the candidate attribute, then we are likely to have found an attribute and (some of) its values. We define the cumulative modified mutual information as follows:

Let $p(w, w_{1...k}) = \sum_{j=1}^k p(w, w_j)$. Then

$$cmi(w_{1...k}; w) = \log \frac{p(w, w_{1...k})}{(\lambda * \sum_{j=1}^k p(w_j)) * ((\lambda - 1) * p(w))}$$

λ is a user-specified parameter, where $0 < \lambda < 1$. We have experimented with several values, and have found that setting λ to 1 yields robust results. Setting λ to 0 implies that a candidate pair is not penalized for the word w being frequent, as long as few words cover most of its conditional probability mass. Table 1 lists several examples of extracted attribute-value pairs.

value	attribute
carrying, storage	case
main, racquet	compartment
ball, welt, side-seam, key	pocket
coat, durable	steel

Table 1. Automatically extracted seed attribute-value pairs

As we can observe from the table, our unsupervised seed generation algorithm captures the intuition we described earlier and extracts high-quality seeds for training the system. We expect to refine this method in the future. Currently, not all extracted pairs are actual attribute-value pairs. One typical example of an extracted incorrect pair are first name - last name pairs, e.g., *Smith* is extracted as an attribute as it occurs as part of many phrases and fulfills our criteria (*Joe Smith, Mike Smith, etc.*) after many first names. Other examples of incorrectly extracted attribute-value pairs include ‘*more* (attribute) – *much* (value)’ and ‘*more* (attribute) – *achieve* (value)’. However, some of the incorrectly extracted examples are rare enough that they do not have much impact on subsequent steps. The current metric accomplishes about 65% accuracy in the tennis category and about 68% accuracy in the football category. We have experimented with manually correcting the seeds by eliminating all those that were incorrect. This did not result in any improvement of the final performance of the overall system, leading us to conclude that our algorithm is robust to noise and is able to deal with noisy seeds.

7 Attribute and Value Extraction

After generating initial seeds, the next step is to use the seeds as labeled training data to extract attributes and values from the unlabeled data. We formulate the extraction as a classification problem where each word or phrase can be classified as attributes or values (or as neither). The classification algorithm is described in the sections below.

7.1 Initial labeling

The initial labeling of data items (words or phrases) is based on whether they match the labeled data. We define four classes to classify words into: *unassigned, attribute, value, or neither*. The initial label for each word defaults to *unassigned* and is changed to the label of any labeled data that it matches or to *neither* if it is a stopword.

7.2 Naïve Bayes Classification

The labeled words are then used as training data for Naïve Bayes that classifies each word or phrase in the unlabeled data as an attribute, a value, or neither. The features used for classification are the words of each unlabeled data item, plus the surrounding 8 words and their corresponding parts of speech. With this feature set, we capture not only each word, but also its context as well as the parts of speech in its context. This is similar to earlier work in extracting named entities using labeled and unlabeled data [3].

7.3 co-EM for Attribute Extraction

Since labeling attributes and values is an expensive process, we use the semi-supervised learning setting by combining small amounts of labeled data with large amounts of unlabeled data. We use the multi-view or co-training [1] setting, where each example can be described by multiple views (e.g., the word itself and the context in which it occurs). The specific algorithm we use is co-EM [8]. Co-EM with Naïve Bayes has been applied to classification, e.g., by [8], but so far as we are aware, not in the context of information extraction. The separation into feature sets we use is that of the word to be classified and the context in which it occurs. Each word is expressed in *view1* by the stemmed word itself, plus the part of speech as assigned by the Brill tagger. The *view2* for this data item is a context of window size 8, i.e. up to 4 words (plus parts of speech) before and up to 4 words (plus parts of speech) after the word or phrase in *view1*. If the context around a *view1* data item is less than 8 words long, we simply limit to the context to what is available.

co-EM Algorithm: co-EM proceeds by initializing the *view1* classifier using the labeled data only. Then this classifier is used to probabilistically label all the unlabeled data. The context (*view2*) classifier is then trained using the original labeled data plus the unlabeled data with the labels provided by the *view1* classifier. Similarly, the *view2* classifier then relabels the data for use by the *view1* classifier, and this process iterates for a number of iterations or until the classifiers converge.

Each iteration consists of collecting evidence for each data item from all the data items in the other view that it occurs with. For example, if a *view2* data item $view2_k$ occurs with (i.e., in the context of) *view1* data items $view1_{i1}$ and $view1_{i2}$, then the probability distribution for $view2_k$ is the averaged distribution of the probabilities currently assigned to $view1_{i1}$ and $view1_{i2}$, weighted by the number of times $view2_k$ appears together with $view1_{i1}$ and $view1_{i2}$, respectively, as well as by the class priors. Our goal is to label unlabeled training examples that are attributes or values, and leave the others unlabeled. Co-EM can be summarized by the following steps: 1) Initialize based on labeled data (see above). 2) Use *view1* to label *view2*. 3) Use *view2* to label *view1*. 4) Repeat for steps 2 and 3 n iterations. 5) Assign final labels to words using the predictions from both views.

Estimating class priors: When estimating class priors for labeling a view, the class priors are estimated from the respective other view's probability distributions. As each data item is associated with a set of data items from the other view with which it co-occurs, together with a count of how many times the two data items co-occurred, we could gather the class prior information by traversing through each data item and weighing the probability distributions from the aligned data elements by the co-occurrence counts.

Conceptually, however, it is easier to think of the class priors as simply obtained from the training data's current distribution in the other view. In other words, when labeling *view2* from *view1*, the class priors for the Naïve Bayes classifier are computed only on *view1*, without reference to the *view2* data items. The resulting probability distributions from these two approaches are the same.

The class probabilities are thus estimated as follows:

$$P(c_k) = \frac{1 + \sum_i^{n_1} cnt(view1_i) * P(c_k|view1_i)}{numclasses + \sum_i^{n_1} cnt(view1_i)}$$

Estimating word probabilities: As with class priors, word probabilities from *view1* are used as training data for *view2*. For example, if a *view1* element has a probability distribution of $p(value) = 0.5$ and $p(attribute) = 0.5$, then the data element is counted as a value example with weight 0.5, but also as an attribute example with weight 0.5.

For all words $view2_j$, estimate the new probability for each class $c_k, 1 \leq k \leq 4$, from all words $view1_i, 1 \leq i \leq n_1$. In practice, the algorithm considers only those $view2_j$ items whose cooccurrence count with $view1_i$ is greater than zero.

$$P(view2_j|c_k) = \frac{1 + \sum_{i=1}^{n_1} cooc(view1_i, view2_j) * P(c_k|view1_i)}{n_2 + \sum_{i=1}^{n_1} cooc(view1_i, view2_j)}$$

$$P(view1_i|c_k) = \frac{1 + \sum_{j=1}^{n_2} cooc(view1_i, view2_j) * P(c_k|view2_j)}{n_1 + \sum_{j=1}^{n_2} cooc(view1_i, view2_j)}$$

Labeling unlabeled examples: In each iteration, we want to use the computed class and word probabilities to label unlabeled data items in the respective other view. This can be done as follows:

$$P(c_k|view2_i) \propto P(c_k) * P(view2_i|c_k)$$

if $view2_i$ does *not* match the labeled training data.

After computing the probabilities for all classes, we must renormalize:

$$P(c_k|view2_j) = \frac{P(c_k|view2_j)}{\sum_{k=1}^{numclasses} P(c_k|view2_j)}$$

However, if $view2_i$ matches the labeled training data,

$$P(c_k|view2_i) = InitialLabeling.$$

$$P(c_k|view1_i) \propto P(c_k) * P(view1_i|c_k)$$

if $view1_i$ does *not* match the labeled training data. As in the case of $view2$, we will need to renormalize after computing the probabilities for each class. Also as above, if $view1_i$ matches the labeled training data,

$$P(c_k|view1_i) = InitialLabeling.$$

Assigning co-EM probabilities to $\langle view1_i, view2_j \rangle$ pairs: After co-EM is run for a pre-specified number of iterations, we assign final co-EM probability distributions to all $\langle view1_i, view2_j \rangle$ pairs as follows:

$$P(c_k|\langle view1_i, view2_j \rangle) = \frac{P(c_k|view1_i) + P(c_k|view2_j)}{2}$$

Final labels are assigned to words and phrases by averaging the predictions of each view's classifier. It should be noted that words that are tagged as attributes or values are not necessarily extracted as part of an attribute-value pair in the next phase. They will only be extracted if they form part of a pair, or if they occur frequently enough by themselves or as part of a longer phrase. The next section will describe this in greater detail.

8 Finding Attribute-Value Pairs

After the classification algorithm has assigned a (probabilistic) label to all unlabeled words, a final important step remains: using these labels to tag attributes and values in the actual product descriptions, i.e., in the original data, and finding correspondences between words or phrases tagged as attributes and values. The classification phase assigns a probability distribution over all the labels to each word (or phrase). This is not enough, because aside from n-grams that are obviously phrases, some subsequent words that are tagged with the same label should be *merged* to form an attribute or value phrase. Additionally, the system must establish *links* between attributes (or attribute phrases) and their corresponding values (or value phrases), so as to form attribute-value pairs. Some unlabeled data items contain more than one attribute and more than one value, so that it is important to find the correct associations between them. We accomplish merging and linking in an interleaved fashion, using the following steps:

- **1:** Link attributes and values if they match a seed pair.

- **2:** Merge words of the same label into phrases if their correlation scores exceed a threshold.
- **3:** Link attribute and value phrases based on directed dependencies as given by a dependency parser [6]: attribute phrases and value phrases can form a pair if there is a governor-dependent relationship between them.
- **4:** Link attribute and value phrases if they exceed a correlation score threshold: unassigned attribute phrases are linked with value phrases if their words exceed a correlation threshold.
- **5:** Link attribute and value phrases based on proximity: unassigned attribute phrases are linked with value phrases if they are adjacent.
- **6:** Adding known, but not overt, attributes: material, country, and/or color.
- **7:** Extract binary attributes, i.e., attributes without values, if they appear frequently or if the unlabeled data item consists of only one word.

Even after all the above pair identification steps, some attribute or value phrases can remain unaffiliated. Some of them are extracted noise, and should not be output. Others are valid attributes with binary values. For instance, the data item *Imported* is a valid attribute with two possible values: *true* or *false*, where the value is simply assigned by the absence or presence of the attribute. We extract only those attributes that are single word data items and those attributes that occur frequently in the data as a phrase.

9 Evaluation

We present evaluation results for experiments performed on tennis and football categories. The tennis category contains 3194 unlabeled data items (i.e., individual phrases from the bulleted list of product descriptions), the football category 72825 items. Automated seed extraction resulted in 169 attribute-value pairs for the tennis category and 180 pairs for football. Table 2 shows a sample list of extracted attribute-value pairs (i.e., the output of the full system), and the phrases that they were extracted from. We ran

Full Example	Attribute	Value
1 1/2-inch polycotton blend tape	polycotton blend tape	1 1/2-inch
1 roll underwrap	underwrap	1 roll
1 tape cutter	tape cutter	1
Extended Torsion bar	bar	Torsion
Synthetic leather upper	#material# upper	leather
Metal ghillies	#material# ghillies	Metal
adiWear tough rubber outsole	rubber outsole	adiWear tough
Imported	Imported	#true#
Dual-density padding with Kinetof foam	padding	Dual-density
Contains 2 BIOflex concentric circle magnet	BIOflex concentric circle magnet	2
93% nylon, 7% spandex	#material#	93% nylon 7% spandex
10-second start-up time delay	start-up time delay	10-second

Table 2. Examples of extracted pairs for system run with co-EM

our system in the following three settings to gauge the effectiveness of each component: 1) only using the automatically generated seeds and the generic lists ('Seeds' in

the tables), 2) with the baseline Naïve Bayes classifier (‘NB’), and 3) co-EM with Naïve Bayes (‘coEM’). To make the experiments comparable, we do not vary pre-processing or seed generation, and keep the pair identification steps constant as well.

The evaluation of this task is not straightforward. The main problem is that people often do not agree on what the ‘correct’ attribute-value pair should be. Consider the example *Audio/JPEG navigation menu*. This phrase can be expressed as an attribute-value pair in multiple ways:

Possible Attribute	Possible Value
<i>navigation menu</i>	<i>Audio/JPEG</i>
<i>menu</i>	<i>Audio/JPEG navigation</i>
<i>Audio/JPEG navigation menu</i>	<i>#true#</i>

In the last case, the entire phrase is considered a binary attribute. All three pairs are both possibly useful attribute-value pairs. The implication is that a human annotator will make one decision, while the system may make a different decision (with both of them being consistent). For this reason, we give partial credit to an automatically extracted attribute-value pair, even if it does not completely match the human annotation. In some cases, an extracted pair deserves only partial credit, while in other cases, the automatically extracted pair is an equally valid attribute-valid pair.

For each of the metrics, we report *type* and *token* performance. Type performance (at the data item level, i.e., at the level of individual product description phrases) refers to performance for unique examples (each example contributes the same regardless of frequency). The data sets contain a number of duplicates, as many attributes apply to more than one product. Token performance refers to performance including duplicates, therefore emphasizing those examples that occur more frequently than others.

9.1 Precision

To measure precision, we evaluate how many automatically extracted pairs match manual pairs completely, partially, or not at all. The percentage of pairs that are fully or partially correct is useful as a metric especially in the context of human post-processing: partially correct pairs are corrected faster than completely incorrect pairs. Tables 3 and 4 list results for this metric for both categories, and for both *type* and *token* evaluations.

	Seeds	NB	coEM		Seeds	NB	coEM
# corr pairs	14	20	50	# corr pairs	252	264	316
# part corr pairs	54	73	132	# part corr pairs	202	247	378
% fully correct	20.29	21.28	26.60	% fully correct	54.90	51.16	44.44
% full or part correct	98.56	98.94	96.81	% full or part correct	98.91	99.03	97.60
% incorrect	1.44	1.06	3.19	% incorrect	1.08	0.97	2.39

Table 3. *Type* (left) and *Token* (right) Precision for *Tennis* Category

The results show that all three systems achieve very high performance for partially correct pairs. As expected, seed generation alone achieves higher accuracy than the system achieves when using unlabeled, and thus noisy, data. As we will see in the following section, however, the decrease in precision when co-EM is used is more than offset by a large increase in recall.

	Seeds	NB	coEM		Seeds	NB	coEM
# corr pairs	12	18	39	# corr pairs	4704	5055	6639
# part corr pairs	63	95	159	# part corr pairs	8398	10256	13435
% fully correct	15.38	14.44	17.65	% fully correct	35.39	31.85	32.04
% full or part correct	96.15	90.40	89.59	% part or full correct	98.56	96.48	96.88
% incorrect	3.85	9.60	10.41	% incorrect	1.44	3.52	3.12

Table 4. *Type* (left) and *Token* (right) Precision for *Football* Category

9.2 Recall

Whenever the system extracts a partially correct pair for an example that is also given by the human annotator, the pair is considered recalled. The results for this metric can be found in tables 5 and 6. Unlike for precision, the recall differs greatly between system settings. More specifically, co-EM aids in recalling a much larger number of pairs, whereas seed generation and Naïve Bayes result in relatively poor recall performance.

	Seeds	NB	coEM		Seeds	NB	coEM
# recalled	66	87	167	# recalled	451	502	668
% recalled	27.62	36.40	69.87	% recalled	51.25	57.05	75.91

Table 5. *Type* (left) and *Token* (right) Recall for *Tennis* Category

	Seeds	NB	coEM		Seeds	NB	coEM
# recalled	68	98	164	# recalled	12629	14617	17868
% recalled	32.69	47.12	78.85	% recalled	39.21	45.38	55.48

Table 6. *Type* (left) and *Token* (right) Recall for *Football* Category

9.3 Word-based Label-independent Precision and Recall

Often there is partial overlap between an automatically extracted pair and a pair given by a human annotator. Sometimes both pairs are equally valid, and sometimes the automatically pair is useful even if it is not completely correct, because it can easily be corrected by a human annotator. For this reason, we also measure the word overlap between manual and automatic pairs. This gives us an idea of how well the system can predict that a word should be part of a pair, even though it may confuse whether the word should be tagged as an attribute or a value. We define the word overlap, word precision, word recall, and word F1 in the standard way. We also measure the amount of ‘confusion’, i.e., how often a (human-tagged) value word was automatically labeled as an attribute or vice versa. Tables 7 and 8 contain the detailed results for this metric.

As was discussed in the seed extraction section, we experimented also with correcting the automatically extracted seeds and running our system with the corrected seeds. This experiment was run only for tennis with co-EM. The result was no significant change in performance. This leads us to conclude that our algorithm is quite robust to noise. It also leads us to the conclusion that the time of a human annotator is likely better spent correcting the final output of the system rather than the input seeds. Correcting the input seeds does not necessarily lead to improved performance, whereas correcting complete output pairs is likely to do so. We will explore this issue further in the context of the active learning phase in our system.

	Seeds	NB	coEM		Seeds	NB	coEM
precision	93.84	93.35	89.53	precision	96.19	95.88	93.11
recall	64.88	76.74	77.46	recall	80.11	84.19	82.16
F1	73.73	82.24	81.47	F1	85.29	88.13	85.84
confusion	10.27	22.68	26.15	confusion	4.76	11.14	16.64

Table 7. *Type* (left) and *Token* (right) Label-independent Word-based Results for *Tennis*

	Seeds	NB	coEM		Seeds	NB	coEM
precision	91.03	83.53	80.07	precision	97.76	94.22	92.94
recall	61.71	69.50	69.69	recall	76.10	81.35	79.70
F1	71.17	73.75	72.47	F1	83.05	85.23	83.47
confusion	18.59	26.04	27.19	confusion	17.80	25.37	25.65

Table 8. *Type* (left) and *Token* (right) Label-independent Word-based Results for *Football*

9.4 Precision Results for Most Frequent Data Items

As the training data contains many duplicates, it is more important to extract correct pairs for the most frequent pairs than for the less frequent ones. In this section, we report precision results for the most frequently data items. This is done by sorting the training data by frequency, and then manually inspecting the pairs that the system extracted for the most frequent 300 data items. This was done only for the system run that includes co-EM classification. We report precision results for the two categories (*tennis* and *football*) in two ways: first, we do a simple evaluation of each unique data item. Then we weight the precision results by the frequency of each sentence. In order to be consistent with the results from the previous section, we define five categories that capture very similar information to the information provided above. The five categories contain *fully correct* and *incorrect*. Another category is *Flip to correct*, meaning that the extracted pair would be *fully correct* if attribute and value were flipped. *Flip to partially correct* refers to pairs that would be *partially correct* if attribute and value were flipped. Finally, we define *partially correct* as before. Table 9 shows the results.

	T nW	T W	F nW	F W
% fully correct	51.25	55.89	51.90	60.01
% flip to correct	12.08	20.14	9.62	10.25
% flip to partially correct	2.92	1.75	0.87	2.14
% partially correct	32.92	21.74	35.27	25.98

Table 9. *Non-weighted* and *Weighted* Precision Results for *Tennis* and *Football* Categories. ‘T’ stands for *tennis*, ‘F’ is *football*, ‘nW’ *non-weighted*, and ‘W’ is *weighted*

10 Discussion

The results show that we can learn product attribute-value pairs in a largely unsupervised fashion with encouraging results. One conclusion is that there is some confusion over which label an extracted word or phrase should have. This is consistent with human disagreement over the labels. Confusion levels increase when co-EM is added to the system, indicating that there were not enough seeds to train a strong classifier to differentiate between attributes and values. Future work will include user-specified lists that can serve as attribute seeds. Such labeled examples can be provided as part of an interactive step or before learning takes place, as is done currently.

The baseline Naïve Bayes algorithm also outperforms the seed only algorithm in recall. This is not surprising, as the seeds are used as labeled training data for Naïve Bayes, which in turn labels additional examples that cannot be labeled by the seeds only. It does, however, not match the recall performance of co-EM, and only outperforms co-EM slightly in terms of precision for *tennis*, but not so for *football*.

Evaluating precision on the most frequent data items yields similar results. We show that there are few incorrect pairs, and we show that especially if we weight by the frequency, the number of completely correct examples is encouragingly high. Furthermore, a fair number of examples can become completely correct if flipped. In the future, we will investigate techniques to detect pairs that should be flipped, which could lead to improved precision. Finally, we can conclude that the results are consistent for both categories, making a strong case for the scalability of the system to other domains.

11 Conclusions and Future Work

We plan to focus on adding an interactive step to the extraction algorithm that will allow users to correct extracted pairs as quickly and efficiently as possible. We are experimenting with different active learning algorithms to minimize the number of corrections required to improve the system. We also plan on experimenting with other categories such as office supplies. We believe that a powerful attribute extraction system can be useful in a wide variety of contexts, as it allows for the normalization of products as attribute-value vectors, which in turn enables fine-grained comparison between products and assortments and improve a variety of applications such as product recommender systems, price comparison engines, demand forecasting, assortment optimization and comparison systems.

References

1. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT-98*, 1998.
2. M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *EMNLP/VLC*, 1999.
3. R. Ghani and R. Jones. A comparison of efficacy of bootstrapping algorithms for information extraction. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition*, 2002.
4. R. Jones. Learning to extract entities from labeled and unlabeled text. Ph.D. Dissertation, 2005.
5. A. M. Kristie Seymore and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.
6. D. Lin. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, 1998.
7. B. Liu, M. Hu, and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW 2005*, 2005.
8. K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM-2000)*, 2000.
9. F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT 2004*, 2004.
10. A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of EMNLP 2005*, 2005.