Text Mining for Product Attribute Extraction

Rayid Ghani, Katharina Probst, Yan Liu[1], Marko Krema, Andrew Fano
Accenture Technology Labs
161 N. Clark St, Chicago, IL
[1]Language Technologies Institute
Carnegie Mellon University, Pittsburgh, PA
{rayid.ghani,katharina.a.probst,marko.krema,andrew.e.fano}@accenture.com,
yanliu@cs.cmu.edu

ABSTRACT

We describe our work on extracting attribute and value pairs from textual product descriptions. The goal is to augment databases of products by representing each product as a set of attribute-value pairs. Such a representation is beneficial for tasks where treating the product as a set of attribute-value pairs is more useful than as an atomic entity. Examples of such applications include demand forecasting, assortment optimization, product recommendations, and assortment comparison across retailers and manufacturers. We deal with both implicit and explicit attributes and formulate both kinds of extractions as classification problems. Using single-view and multi-view semi-supervised learning algorithms, we are able to exploit large amounts of unlabeled data present in this domain while reducing the need for initial labeled data that is expensive to obtain. We present promising results on apparel and sporting goods products and show that our system can accurately extract attribute-value pairs from product descriptions. We describe a variety of application that are built on top of the results obtained by the attribute extraction system.

1. INTRODUCTION

Retailers have been collecting large amounts of data from various sources. Most retailers have data warehouses of transaction data containing customer information and related transactions. These data warehouses also contain product information but surprisingly that information is often very sparse and limited. For example, most retailers treat their products as atomic entities with very few related attributes (typically brand, size, or color)[1]. Treating products as atomic entities hinders the effectiveness of many applications that businesses currently use transactional data for such as demand forecasting, assortment optimization, product recommendations, assortment comparison across retailers and manufacturers, or product supplier selection. If a business could represent their products as attributes and attribute values, all of the above applications could be improved significantly.

Suppose a grocery store wanted to forecast sales of *Tropicana Low Pulp Vitamin-D Fortified Orange Juice 1-liter plastic bottle*. Typically, they would use sales of the same product from the same time last year and adjust that number based on some new information. Now suppose that this particular product is new and there is no data available from previous years. Representing the product as a set of attribute-value pairs (*Brand: Tropicana, Pulp: Low, Fortified with: Vitamin-D, Size: 1 liter, Bottle Type: Plastic*) would enable the retailer to use data from other products having similar attributes and forecast more accurately. Even if the product was not new, representing it in terms of attribute-value pairs would allow comparison with other related products and improve any sales forecasts. The same holds true in the other applications mentioned earlier.

Many retailers have realized this recently and are trying to enrich their product databases with corresponding attributes and values for each product. In our discussions with retail experts, we found that in most cases, this is being done manually by looking at (natural language) product descriptions that are available in an internal database or on the web or by looking at the actual physical product packaging in the store. Our goal is to make the process of extracting attribute-value pairs from product descriptions more efficient and cheaper by developing an interactive tool that can help human experts with this task. It is somewhat surprising that the problem we tackle in this paper actually exists. One would expect product manufacturers and retailers to have a database of products and their corresponding attributes. Unfortunately, no such data sources exists for most product categories.

In this paper, we describe two systems: one that extracts implicit (semantic) attributes and one that extracts explicit attributes from product descriptions and populates a knowledge base with these products and attributes. This work was motivated by discussions with CRM experts and retailers who currently analyze large amounts of transactional data but are unable to systematically 'understand' their products. For example, a clothing retailer would know that a particular customer bought a shirt and would also know the SKU, date, time, price, and size of a particular shirt that was purchased. While there is some value to this data, there is a lot of information not being captured: characteristics (e.g., *logo printed on the back*), as well as semantic properties (e.g., *trendiness*, *formality*). Some of the attributes, e.g., *printed logo*, are often explicit in the product descriptions that can be found on retailer web sites, whereas others, such as 'trendiness', are implicit. We describe our work on a system capable of inferring both kinds of attributes to enhance product databases.

We also describe several applications of an enriched product database including recommender systems and competitive intelligence tools and provide evidence that our approach can successfully build a product database with accurate facts which can then be used to create profiles of individual products, groups of products, or entire retail stores. Similarly, we can create profiles of individual *customers* or groups of customers. Another possible application is a retailer comparison system that allows a retailer to compare its assortment with that of a competitor's, e.g., to determine how many high-end products each retailer offers.

[1]We were very surprised to discover this after talking to many large retailers currently trying to use transactional data for data mining.

2. RELATED WORK

There has been significant research on extracting information from text documents but we are not aware of any 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.

[1] describe a system that consists of two parts: the first part focuses 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 second phase deals with extraction of polarized descriptors, e.g., 'good', 'too small', etc. [16] describe a similar approach: they extract attributes by first extracting noun phrases, and then computing the pointwise mutual information between the noun phrases and salient context patterns (such as *'scanner has'*). Similarly to [1], the extraction phase is followed by an opinion word extraction and polarity detection phase. This work is related to our work on extracting explicit attributes: in both cases, 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 to form pairs. 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 probably use *megapixels* or *lens manufacturer* in the specifications.

Information extraction with the goal of filling templates, e.g., [11; 15], 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 for explicit attributes. For the extraction of implicit attributes, we fill a pre-defined template list, but nothing is explicitly extracted from the descriptions themselves.

Recent work in bootstrapping for information extraction using semi-supervised learning has focused on the task of named entity extraction [7; 10; 4] which only deals with the first part of our task (classifying the words/phrase as attributes or values independently of each other) and not with associating the extracted attributes with the corresponding extracted values.

3. EXTRACTING IMPLICIT SEMANTIC ATTRIBUTES

At a high level, our system deals with text associated with products to infer a predefined set of semantic attributes for each product. These attributes can generally be extracted from any information related to the product but in this paper, we only use the descriptions associated with each item. The attributes extracted are then used to populate a product database. The process is described below.

3.1 Data Collection

We constructed a web crawler to visit web sites of several large apparel retail stores and extract names, URLs, descriptions, prices and categories of all products available. This was done very cheaply by exploiting regularities in the html structure of the websites and by manually writing wrappers[2]. We realize that this restricts the collection of data from websites where we can construct wrappers; although automatically extracting names and descriptions of products from arbitrary websites would be an interesting application area for information extraction or segmentation algorithms [11], we decided to take the manual approach. The extracted items and attributes were placed in a database and a random subset was chosen to be labeled.

[2]In our case, the wrappers were simple regular expressions that took the html content of web pages into account and extracted specific pieces of information.

3.2 Defining the set of attributes to extract

After discussions with domain experts, we defined a set of semantic attributes that would be useful to extract for each product. We believe that the choice of attributes should be made with particular applications in mind and that extensive domain knowledge should be used. We currently infer values for 8 kinds of attributes for each item; more attributes that are potentially interesting could be added. The attributes we use are *Age Group*, *Functionality*, *Price point*, *Formality*, *Degree of Conservativeness*, *Degree of Sportiness*, *Degree of Trendiness*, and *Degree of Brand Appeal*.

The last four attributes (conservative, sportiness, trendiness, and brand appeal) have five possible values 1 to 5 where 1 corresponds to low and 5 is the highest (e.g., for *trendiness*, 1 would be not trendy at all and 5 would be extremely trendy).

3.3 Labeling Training Data

The data (product name, descriptions, categories, price) collected by crawling websites of apparel retailers was placed into a database and a small subset ($\sim$600 products) was given to a group of fashion-aware people to label with respect to each of the attributes described in the previous section. They were presented with the description of the predefined set of attributes and the possible values that each feature could take (see above).

3.4 Training from the Labeled Data

We treat the learning problem as a traditional text classification problem and create one text classifier for each semantic attribute. For example, in the case of the *Age Group* attribute, we classify the product into one of five classes (*Juniors*, *Teens*, *GenX*, *Mature*, *All Ages*). We use Naïve Bayes as commonly used for text classification tasks as the initial approach for this supervised learning problem.

3.5 Incorporating Unlabeled Data using EM

In our initial data collection phase, we collected names and descriptions of thousands of women's apparel items from websites. Since the labeling process was expensive, we only labeled about 600 of those, leaving the rest as unlabeled. Recently, there has been much interest in learning algorithms that combine information from labeled and unlabeled data. Such approaches include using Expectation-Maximization to estimate maximum a posteriori parameters of a generative model [14], using a generative model built from unlabeled data to perform discriminative classification [8], and using transductive inference for support vector machines to optimize performance on a specific test set [9]. These results have shown that using unlabeled data can significantly decrease classification error, especially when labeled training data are sparse.

For the case of textual data in general, and product descriptions in particular, obtaining the data is very cheap. A simple crawler can be built and large amounts of unlabeled data can be collected for very little cost. Since we had a large number of product descriptions that were collected but unlabeled, we decided to use the Expectation-Maximization algorithm to combine labeled and unlabeled data for our task.

3.5.1 Expectation-Maximization

If we extend the supervised learning setting to include unlabeled data, Naïve Bayes is no longer adequate to find maximum a posteriori parameter estimates. The Expectation-Maximization (EM)

technique can be used to find locally maximum parameter estimates.

EM is an iterative statistical technique for maximum likelihood estimation in problems with incomplete data [5]. Given a model of data generation, and data with some missing values, EM will locally maximize the likelihood of the parameters and give estimates for the missing values. The Naïve Bayes generative model allows for the application of EM for parameter estimation. In our scenario, the class labels of the unlabeled data are treated as the missing values.

3.6 Experimental Results

In order to evaluate the effectiveness of the algorithms described above for building an accurate knowledge base, we calculated classification accuracies using the labeled product descriptions and 5 fold cross-validation. The evaluation was performed for each attribute and the table below (Table 1) reports the accuracies. The first row in the table (baseline) gives the accuracies if the most frequent attribute value was predicted as the correct class. The experiments with Expectation-Maximization were run with the same amount of labeled data as Naïve Bayes but with an additional 3500 unlabeled product descriptions.

We can see that Naïve Bayes outperforms our baseline for all the attributes. Using unlabeled data and combining it from the initially labeled product descriptions with EM helps improve the accuracy even further.

3.7 Results on a new test set

The results reported earlier in Table 1 are extremely encouraging but are indicative of the performance of the algorithms on a test set that follows a similar distribution as the training set. Since we first extracted and labeled product descriptions from a retail website and then used subsets of that data for training and testing (using 5 fold cross-validation), the results may not hold for test data that is drawn from a different distribution or a different retailer.

The results we report in Table 2 are obtained by training the algorithm on the same labeled data set as before but testing it on a small (125 items) new labeled data set collected from a variety of retailers that were different from initial training (both labeled and unlabeled) set. As we can observe, the results are consistently better than baseline and in some cases, even better than in Table 1. This results enables us to hypothesize that our system can be applied to a wide variety of data and can adapt to different distributions of test sets using the unlabeled data.

4. EXTRACTING EXPLICIT ATTRIBUTES

The first part of this paper dealt with extracting soft, semantic attributes that are implicitly mentioned in descriptions. Another class of attributes associated with products are explicit physical attributes such as size and color. The second part of this paper discusses the task of extracting these explicit attributes, i.e., attribute-value pairs that are explicitly mentioned in the data.

As mentioned above, our discussions with retail experts led us to conclude that in most cases, this is being done today manually by looking at (natural language) product descriptions that are available in an internal database or on the web or by looking at the actual physical product packaging in the store. The work presented in this paper is motivated by the need to make the process of extracting attribute-value pairs from product descriptions more efficient and cheaper by developing an interactive tool that can help human experts with this task. We begin with an overview of our system:

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 and is able to automatically extract automatically initial seed list for training using the unlabeled data. The output of the unsupervised seed extraction algorithm is combined with the unlabeled data and used by co-EM to extract product attributes and values which are then linked together using dependency information and correlation scores. We present promising results on multiple categories of sporting goods products and show that our system can accurately extract attribute-value pairs from product descriptions.

1. **Data Collection** from an internal database or from the web using web crawlers and wrappers, as done in the previous section.
2. **Seed Generation** either by generating them automatically or by obtaining human-labeled training data.
3. **Attribute-Value Entity Extraction** using a semi-supervised co-EM algorithm, because it can exploit the vast amounts of unlabeled data that can be collected cheaply.
4. **Attribute-Value Pair Relationship Extraction** by associating extracted attributes with corresponding extracted values. We use a dependency parser (Minipar, [12]) to establish links between attributes and values as well as correlation scores between words.
5. **User Interaction** to correct the results as well as to provide training data for the system to learn from using active learning techniques.

The modular design allows us to break the problem into smaller steps, each of which can be addressed by various approaches. We only focus on tasks 1-4 in this paper. In the following sections, we describe our approach to each of the four tasks in greater detail.

4.1 Data

The data required for extracting product attributes and values can come from a variety of sources. The product descriptions can reside in an internal product database or they may be found on the retailer website. For the experiments reported in this paper, we developed a web crawler that crawls retailer websites and extracts product descriptions.

For the work presented in this paper, we crawled the web site of a sporting goods retailer[3]. We believe that sporting goods is an interesting and relatively challenging domain because unlike categories such as 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 input to our system is a set of product descriptions. Some examples of entries in these descriptions are:

1 tape cutter
4 rolls white athletic tape
Audio/Video Input Jack
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, e.g., *cutter* being modified both by *1* and by *tape*. For this reason, there is often

[3]www.dickssportinggoods.com

Table 1: Classification accuracies for each attribute using 5 fold cross-validation. Naïve Bayes uses only labeled data and EM uses both labeled and unlabeled data.

Algorithm	Age Group	Functionality	Formality	Conservative	Sportiness	Trendiness	Brand Appeal
Baseline	29%	24%	68%	39%	49%	29%	36%
Naïve Bayes	66%	57%	76%	80%	70%	69%	82%
EM	78%	70%	82%	84%	78%	80%	91%

Table 2: Classification accuracies when trained on the same labeled data as before but tested on a new set of test data that is collected from a new set of retailers

Algorithm	Age Group	Functionality	Formality	Conservative	Sportiness	Trendiness	Brand Appeal
Naïve Bayes	83%	45%	61%	70%	81%	80%	87%

no definitive answer as to what the extracted attribute-value pair should be, even for humans inspecting the data.

4.2 Pre-Processsing

The product descriptions collected by the web crawler are pre-processed in several steps. First, the data is tagged with parts of speech using the Brill tagger [3]. Second, the data is stemmed, using the Porter stemmer [17], in order to normalize the data by mapping morphological variations of words to the same token.

In order to generalize the observed data to the appropriate level of generalization, and in order to increase the amount of training data available for a given pattern or context, we replace all numbers (in any notation, e.g., scientific, floating point, etc.) with a unique token (*#number#*). For the same reason, all measures (e.g., *liter, kg*) are replaced by a unique token (*#uom#*).

Additionally, we compute several correlation scores between all pairs of words: we compute Yule's Q statistic, mutual information, as well as the χ^2 scores in order to recognize phrases with high precision.

5. SEED GENERATION

Once the data is collected and processed, the next step is to provide labeled seeds for the learning algorithms to learn from. The extraction algorithm is seeded in two ways: with a list of known values and attributes, as well as by an unsupervised, automated algorithm that extracts a set of seed attribute-value pairs from the unlabeled data. Both of these seeding mechanisms are designed to facilitate scaling to other domains.

5.1 Generic and domain-specific lists as labeled seeds

We use a very small amount of labeled training data in the form of generic and domain-specific value lists for colors, materials, countries, and units of measures (*kg, oz.*, etc.). In addition to the generic value list, we use a list of domain-specific (in our case, sports) values and attributes. The values consist of sports teams (such as *Pittsburgh Steelers*), and contains 82 entries. Aside from these easily replaceable generic and domain-specific lists, the first four phases of the system (as specified in the overview above) work in an unsupervised fashion.

5.2 Unsupervised Seed Generation

Our unsupervised seed generation method extracts few, but relatively accurate attribute-value pairs from the training data. The approach uses correlation scores to find candidates, and makes use of POS tags by excluding certain words from being candidates for extraction. Unsupervised seed extraction is performed after the pre-processing steps described above.

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, the biggest being that attributes generally have more than one possible value, e.g., 'front pockets', 'side pockets', 'zipper pockets', etc. 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, a reasonable heuristic. Suppose word w (in position $i+1$) occurs with n unique words $w_{1\ldots n}$ in position i. We rank the words $w_{1\ldots n}$ by their conditional probability of occuring right before word w $p(w_j|w), w_j \in w_{1\ldots n}$, 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. 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 intuition is captured 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.:

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

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

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.

Table 3 lists several examples of extracted attribute-value pairs.

Not all extracted pairs are actual attribute-value pairs. One typical example of an extracted incorrect pair are first name - last name pairs. We could use a list of common names to filter out these seeds but during our experiments, we found that 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

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

Table 3: Automatically extracted seed attribute-value pairs

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 extraction performance, leading us to conclude that our algorithm is robust to noise and able to deal with noisy seeds.

5.3 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. In this phase, we treat each word separately with two exceptions: one, if a phrase is listed in the generic or domain-specific seed lists, we treat the entire phrase as an atom. Second, if an n-gram is recognized with high certainty as a phrase, as measured by Yule's Q, mutual information, and χ^2 scores, it is again treated as an atomic entity.

We formulate the extraction as a classification problem where each word (or phrase) can be classified in one of three classes: attribute, value, or neither. We treat it as a supervised learning problem and use Naïve Bayes as our first approach. The initial seeds generated (as described in the previous section) are used to label training data which Naïve Bayes uses to train a classifier. Since our goal is to create a system that minimizes human effort required to train the system, we use semi-supervised learning to improve the performance of Naïve Bayes by exploiting large amounts of unlabeled data available for free on the web. Gathering product descriptions (from retail websites) is a relatively cheap process using simple web crawlers. The expensive part is labeling the words in the descriptions as attributes or values. We augment the initial seeds (labeled data) with the all the unlabeled product descriptions collected in the data collection phase and use semi-supervised learning (co-EM [13] with Naïve Bayes) to improve attribute-value extraction performance. The classification algorithm is described in the sections below.

5.3.1 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 probability distribution for each word defaults to 'unassigned'. If the unlabeled example does match the labeled data, then we simply assign it this label. If the word appears on a stoplist, it is tagged as *neither*, if it appears on the list of known attributes or values, it is tagged accordingly.

5.3.2 Naïve Bayes Classification

We apply the extracted and generic lists to the unlabeled data in order to assign labels to as many words as possible, as described in the previous section. These labeled words are then used as training data for Naïve Bayes that classifies each word or phrase in the unlabeled data as an attribute, 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 each word, its context, as well as the parts of speech in its context.

5.3.3 co-EM for Attribute Extraction

The availability of a small amount of labeled training data and a large amount of unlabeled data allows us to use the semi-supervised learning setting. We use the multi-view or co-training [2] 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: a multi-view semi-supervised learning algorithm, proposed by Nigam & Ghani [13], that combines features from both co-training [2] and EM. co-EM is iterative, like EM, but uses the feature split present in the data, like co-training. The separation into feature sets we used is that of the word to be classified and the context in which it occurs. co-EM with Naïve Bayes has been applied to classification, e.g., by [13], but so far as we are aware, not in the context of information extraction.

co-EM is a multi-view algorithm, and requires two views for each learning example. Each word or phrase is expressed in *view1* by the stemmed word or phrase itself, and the parts 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*. 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.

5.4 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, 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 consecutive words that are tagged with the same label should be merged to form an attribute or a value. 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-value pair, so that it is important to find the correct associations between them. We do this by first establishing attribute-value pairs using the seed pairs that are extracted at the beginning of the learning process. We then use the labels that were assigned during the classification stage together with correlation scores to merge words into phrases, and to establish attribute-value links using a set of selection criteria. Attributes and values are then linked into pairs using the dependencies given by Minipar. We add additional attributes that are not present in the data, but were contained in the initial list of seeds (colors, countries, and materials). Finally, some unlinked attributes are retained as binary attributes. In the process of establishing attribute-value pairs, we exclude words of certain parts of speech, namely most closed-class items. For example, prepositions, conjunctions, etc., are not good candidates for attributes or values, and thus are not extracted.

The pair finding algorithm proceeds in seven steps:

- **Step 1:** Link based on seed pairs
- **Step 2:** Merge words of the same label into phrases if their correlation scores exceed a threshold
- **Step 3:** Link attribute and value phrases based on directed dependencies as given by Minipar
- **Step 4:** Link attribute and value phrases if they exceed a correlation score threshold
- **Step 5:** Link attribute and value phrases based on proximity
- **Step 6:** Adding known, but not overt, attributes: material, country, and/or color
- **Step 7:** Extract binary attributes, i.e., attributes without values, if they appear frequently or if the unlabeled data item consists of only one word

5.5 Evaluation

In this section, 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 list of product descriptions), the football category 72825 items. Table 4 shows a sample list of extracted attribute-value pairs, together with the phrases that they were extracted from. This is to give an idea of what kinds of attributes are extracted, and is supplemented with a more quantitative evaluation in the following section.

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 Kinetofoam	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 4: Examples of extracted pairs for system run with co-EM

We ran 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 ('co-EM'). In order 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 following example:

Audio/JPEG navigation menu

This phrase can be expressed as an attribute-value pair in multiple ways, e.g., *navigation menu* (attribute) and *Audio/JPEG* (value) or *menu* (attribute) and *Audio/JPEG navigation* (value) or as the whole phrase forming a binary attribute.

All three pairs are 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 have to give partial credit to an automatically extracted attribute-value pair that is correct, even if it does not completely match the human annotation.

5.5.1 Precision

To measure precision, we evaluate how many automatically extracted pairs match manual pairs completely, partially, or not at all. If the system extracts a pair that has no overlap with any human extracted pair for this data item, then the pair would be counted as fully incorrect.

We report percentages of fully correct, partially correct, and incorrect pairs as well as the percentage of pairs that are fully *or* partially correct. The last metric is useful especially in the context of human post-processing: partially correct pairs are corrected faster than completely incorrect pairs. Tables 5 and 6 list the results.

	Seeds	**NB**	**coEM**
# corr pairs	252	264	316
# part corr pairs	202	247	378
% fully correct	**54.90**	**51.16**	**44.44**
% full or part correct	**98.91**	**99.03**	**97.60**
% incorrect	**1.08**	**0.97**	**2.39**

Table 5: Precision for *Tennis* Category

	Seeds	**NB**	**coEM**
# corr pairs	4704	5055	6639
# part corr pairs	8398	10256	13435
% fully correct	**35.39**	**31.85**	**32.04**
% part or full correct	**98.56**	**96.48**	**96.88**
% incorrect	**1.44**	**3.52**	**3.12**

Table 6: Precision for *Football* Category

5.5.2 Recall

When the system extracts a partially correct pair that is also extracted by the human annotator, this pair is considered recalled. The results for this metric can be found in tables 7 and 8.

	Seeds	**NB**	**coEM**
# recalled	451	502	668
% recalled	**51.25**	**57.05**	**75.91**

Table 7: Recall for *Tennis* Category

5.5.3 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 frequent 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.

	Seeds	NB	coEM
# recalled	12629	14617	17868
% recalled	**39.21**	**45.38**	**55.48**

Table 8: Recall for *Football* Category

	T nW	T W	F nW	F W
# correct	123	702	178	21362
% fully correct	**51.25**	**55.89**	**51.90**	**60.01**
# flip to correct	29	253	33	3649
% flip to correct	**12.08**	**20.14**	**9.62**	**10.25**
# flip to partially correct	7	22	3	761
% flip to partially correct	**2.92**	**1.75**	**0.87**	**2.14**
# partially correct	79	273	121	9245
% partially correct	**32.92**	**21.74**	**35.27**	**25.98**
# incorrect	2	6	8	579
% incorrect	**0.83**	**0.48**	**2.33**	**1.63**

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*

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.

5.5.4 Discussion

The results presented in the previous section show that we can learn product attribute-value pairs in a largely unsupervised fashion with encouraging results. It is not straightforward to evaluate the performance of our system, but by using a variety of metrics, we can detect several trends in the results. First, the automatically extracted seeds plus the generic lists (without classification) result in a high-precision system, but with very low recall. Learning from these seeds by adding supervised learning (Naïve Bayes) into the process results in somewhat higher recall of pairs with only small drops in precision if any. Exploiting unlabeled data by using co-EM improves the recall even more while keeping precision comparable. Especially in the tennis category, recall improves dramatically as a result of co-EM learning.

6. APPLICATIONS

The system we presented in this paper is used to augment product databases with attributes and corresponding values for each product. Such an augmented product database can be used for a variety of applications. Demand Forecasting, Assortment Optimization, Product Recommendations, Assortment comparison across retailers and manufacturers, and Product Supplier Selection are just some of the applications that can be improved using the augmented product database. In this section we describe some specific applications that we have developed on top of our system.

6.1 Recommender Systems

Being able to analyze the text associated with products and map it to a set of attributes and values in real-time gives us the ability to create instant profiles of customers shopping in an online store. As the shopper browses products in a store, the system running in the background can extract the name and description of the items and using the trained system, can infer implicit (semantic) and explicit features of that product. This process can be used to create instant profiles based on viewed items without knowing the identity of the shopper or the need to retrieve previous transaction data. The system can then be used to suggest subsequent products to new and infrequent customers for whom past transactional data may not be available. Of course, if historical data is available, our system can use that to build a better profile and recommend potentially more targeted products. We believe that this ability to engage and target new customers tackles one of the challenges currently faced by commercial recommender systems [18] and can help retain new customers.

We have built a prototype of a recommender system for women's apparel items by using our knowledge base of product attributes. More details about the recommender system can be found in [6]. The user profile is stored in terms of probabilities for each attribute value which allows us flexibility to include mixture models in future work in addition to being more robust to changes over time. Our recommender system improves on collaborative filtering as it would work for new products which users haven't browsed yet and can also present the user with explanations as to why they were recommended certain products (in terms of the attributes). We believe that our system also performs better than standard content-based systems. Although content-based systems also use the words in the descriptions of the items, they traditionally use those words to learn one scoring function. In contrast, our system changes the feature space from words (thousands of features) to only the implicit and/or explicit attributes that were extracted.

6.2 CopyWriters Marketing Assistant

The ability to extract attributes for products is not only useful for customer profiling but also for product marketing. In our discussions with retailers, we realized that an important component of product marketing is the product description that is used in a catalog or website. We built the CopyWriters Marketing Assistant to help marketing professionals position the product correctly by helping them write the product descriptions. The writers select a set of target attributes that they intend to convey to the customer about that product and then write a description that is intended to convey those attributes. The description is then passed through the attribute extraction system which gives the attributes that the description "actually" contains or conveys. The system compares the actual attributes with the intended ones given by the writers and gives suggestions about what kinds of concepts and words to add in order to move the descriptions towards the intended attributes. For example, if the writers intended the marketing to convey that the product is extremely classic but the current description is rated by the extraction system as trendy, it would suggest using words such as timeless or seasonless. Multiple systems can be trained by obtaining labeled examples from different groups of people (different customer segments for example) which would allow the tool to give feedback about what a particular group of people would think of a particular product description. We have shown this tool to several retailers and have received encouraging feedback about its utility and effectiveness.

6.3 Store Profiling & Assortment Comparison Tool

We also have a prototype that profiles retailers to build competi-

tive intelligence applications. For example, by closely tracking the product offerings we can notice changes in the positioning of a retailer. We can track changes in the industry as a whole or specific competitors and compare it to the performance of retailers. By profiling their aggregate offerings, our system can enable retailers to notice changes in the positioning of product lines by competitor retailers and manufacturers. This ability to profile retailers enables strategic applications such as competitive comparisons, monitoring brand positioning, tracking trends over time, etc.

Our assortment comparison tool is used to compare assortment between different retailers. It allows the user to explore the assortment, as expressed by attribute-value pairs, in a variety of ways: for example, the user can visualize how many products a retailer offers with a certain value for an attribute. The user can also compare what proportion of one retailer's products fall into a specific category as expressed by an attribute-value pair, e.g., what proportion of the clothing offered by the retailer are children's clothing and compare it with that of a competing retailer.

Another application of our system is assortment optimization. A retailer can express each product as a vector of attribute-value pairs, and can then run regression algorithms using sales data. This can provide quantitative information about the monetary value of each attribute and what makes certain customer buy certain products.

7. CONCLUSIONS AND FUTURE WORK

We described our work on a system capable of inferring implicit and explicit attributes of products enabling us to enhance product databases for retailers. Treating products as sets of attribute-value pairs rather than as atomic entities can boost the effectiveness of many business applications such as demand forecasting, assortment optimization, product recommendations, assortment comparison across retailers and manufacturers, or product supplier selection. Our system allows a business to represent their products in terms of attributes and attribute values without much manual effort.The system learns these attributes by applying supervised and semi-supervised learning techniques to the product descriptions found on retailer web sites. The system can be bootstrapped from a small number of labeled training examples utilizes the large number of cheaply obtainable unlabeled examples (product descriptions) available from retail websites.

The completed work leaves many avenues for future work. Most immediate future work will 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 working on active learning algorithms that are able to utilize the unlabeled data in order to most effectively learn from user feedback.While future work remains, we have shown the usefulness of the approaches in many prototype applications that we have built at Accenture Technology Labs. We believe that the work described in this paper not only improves the state of data mining in the retail industry by augmenting product databases with attributes and values, but also provides interesting challenges to the research community working on information extraction, semi-supervised and active learning techniques.

8. REFERENCES

[1] M. H. Bing Liu and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of WWW 2005*, 2005.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT-98*, 1998.

[3] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 1995.

[4] M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *EMNLP/VLC*, 1999.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[6] R. Ghani and A. E. Fano. Building recommender systems using a knowledge base of product semantics. In *Proceedings of the Workshop on Recommendation and Personalization in ECommerce at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web based Systems*, 2002.

[7] R. Ghani and R. Jones. A comparison of efficacy of bootstrapping algorithms for information extraction. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition*, 2002.

[8] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*, 1999.

[9] T. Joachims. Transductive inference for text classification using support vector machines. In *Machine Learning: Proceedings of the Sixteenth International Conference*, 1999.

[10] R. Jones. *Learning to Extract Entities from Labeled and Unlabeled Text*. Ph.D. Dissertation, 2005.

[11] 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.

[12] D. Lin. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, 1998.

[13] 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.

[14] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[15] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT 2004*, 2004.

[16] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of EMNLP 2005*, 2005.

[17] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[18] J. Schafer, J. Konstan, and J. Riedl. Electronic commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 5:115–152, 2000.