



University of Pennsylvania  
ScholarlyCommons

---

Operations, Information and Decisions Papers

Wharton Faculty Research


---

5-2007

# Constraint-Based Ontology Induction From Online Customer Reviews

Thomas Lee  
*University of Pennsylvania*

Follow this and additional works at: [http://repository.upenn.edu/oid\\_papers](http://repository.upenn.edu/oid_papers)

 Part of the [Other Biochemistry, Biophysics, and Structural Biology Commons](#), [Other Education Commons](#), [Other Life Sciences Commons](#), and the [Technology and Innovation Commons](#)

---

## Recommended Citation

Lee, T. (2007). Constraint-Based Ontology Induction From Online Customer Reviews. *Group Decision and Negotiation*, 16 (3), 255-281. <http://dx.doi.org/10.1007/s10726-006-9065-3>

This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/oid\\_papers/91](http://repository.upenn.edu/oid_papers/91)  
For more information, please contact [repository@pobox.upenn.edu](mailto:repository@pobox.upenn.edu).

---

# Constraint-Based Ontology Induction From Online Customer Reviews

## **Abstract**

We present an unsupervised, domain-independent technique for inducing a product-specific ontology of product features based upon online customer reviews. We frame ontology induction as a logical assignment problem and solve it with a bounds consistency constrained logic program. Using shallow natural language processing techniques, reviews are parsed into phrase sequences where each phrase refers to a single concept. Traditional document clustering techniques are adapted to collect phrases into initial concepts. We generate a token graph for each initial concept cluster and find a maximal clique to define the corresponding logical set of concept sub-elements. The logic program assigns tokens to clique sub-elements. We apply the technique to several thousand digital camera customer reviews and evaluate the results by comparing them to the ontologies represented by several prominent online buying guides. Because our results are drawn directly from customer comments, differences between our automatically induced product features and those in extant guides may reflect opportunities for better managing customer-producer relationships rather than errors in the process.

## **Keywords**

concept learning, knowledge acquisition, ontology, text analysis, text mining

## **Disciplines**

Other Biochemistry, Biophysics, and Structural Biology | Other Education | Other Life Sciences | Technology and Innovation

TITLE: Constraint-based Ontology Induction from Online Customer Reviews

AUTHOR: Thomas Lee, thomasyl@wharton.upenn.edu

AFFILIATION: University of Pennsylvania, The Wharton School  
Department of Operations and Information Management

KEYWORDS: Ontology; concept learning; knowledge acquisition; text analysis; text mining

ABSTRACT:

We present an unsupervised, domain independent technique for inducing a product-specific ontology of product features based upon on-line customer reviews. We frame ontology induction as a logical assignment problem and solve it with a bounds consistency constrained logic program. Using shallow natural language processing techniques, reviews are parsed into phrase sequences where each phrase refers to a single concept. Traditional document clustering techniques are adapted to collect phrases into initial concepts. We generate a token graph for each initial concept cluster and find a maximal clique to define the corresponding logical set of concept sub-elements. The logic program assigns tokens to clique sub-elements. We apply the technique to several thousand digital camera customer reviews and evaluate the results by comparing them to the ontologies represented by several prominent online buying guides. Because our results are drawn directly from customer comments, differences between our automatically induced product features and those in extant guides may reflect opportunities for better managing customer-producer relationships rather than errors in the process.

## 1. INTRODUCTION

The World Wide Web offers a vast repository of product information provided by and for producers as well as consumers. These digital resources include manufacturer information, retail locations, prices, customer reviews, and more. In the United States, the result is growth in online retail sales of over 20% each of the past three years; double-digit growth is projected to continue for the next several years [31]. Worldwide, eCommerce is projected to play a pivotal role in bridging the economic divide between developing and developed nations [19].

As markets increasingly turn to the Web, the challenge facing producers and consumers alike is the problem of understanding and interpreting the myriad sources of information that are available online. Different customers, to say nothing of vendors, will often refer to identical

products or features using inconsistent or incompatible terminology. For example, in automotive sales, ‘zoom’ might be a colloquial reference to speed whereas for digital cameras, ‘zoom’ might refer to optical magnification. Furthermore, within the context of digital cameras, ‘zoom’ might refer either to ‘digital zoom’ or to ‘optical zoom.’

The Semantic Web promises a set of tools to enable the navigation of eCommerce retail resources in a consistent, unambiguous manner. As a part of this effort, this paper reports on techniques to process online customer reviews in a particular product space. Our objective is to automatically learn a structured vocabulary to support the use and integration of online product information. This common vocabulary, used to discuss both concepts and relationships between concepts, is defined as an *ontology* [13, 24].

Each concept in a product ontology corresponds to a particular product feature such as *removable memory storage*. In an abstract sense, an online customer review is a list of those features or concepts that a customer likes or dislikes. Because different customers might refer to a particular feature in different ways (e.g. ‘compact flash,’ ‘compactflash,’ and ‘memory stick’ are different string literals for describing *removable memory*), an ontology normalizes the language for distinguishing between different features. Normalizing the terminology enables users to mine heterogeneous sources of product information in a uniform manner. More than terminology, however, ontologies also convey information about the relationships between concepts.

There are two types of relationships between concepts that we seek to capture in an ontology. To illustrate these two relationships, consider the product space of digital cameras. Where each review is a list of camera feature(s), the first relationship is the *hyponym* relationship that subdivides a concept like *camera feature* into its sub-classes. *Removable memory* is a feature of

digital cameras as are *resolution*, and *compression*. We can then distinguish review phrases like '2 AA NiMH,' which are instances of *battery type*, from other literals like 'MPEG-1'. Moreover, a given subclass might be hierarchically decomposed into finer granules of specificity. For example, *MPEG*, *JPEG*, and *TIFF* are hyponyms of *compression*. The string 'MPEG-1' is an instance of the concept *MPEG* while the string 'JPEG' is an instance of the concept *JPEG* which is-a *compression* scheme which is-a *camera feature*. By representing concept features at different levels of precision, we can aggregate reviews at different levels of granularity and isolate whether the concept of *compression* is important or whether customers care about a specific compression scheme because it is easier to import into their software for making holiday greeting cards.

The second type of concept relationship that we would like to learn is the 'part-of' or 'has-a' relationship(s) (*meronym/holonym*) that describe the 'attributes' of a particular product feature. For example, a battery type might be described by the *size* (instances of size include 'kcrv3' and 'AA'), by the *chemistry* (instances of battery chemistry are 'LiMH' and 'Alkaline') or by the *voltage*. Each attribute constitutes its own distinct concept. Distinguishing between concepts is potentially significant to both producers and consumers. For example, digital camera users who travel worldwide are concerned about battery *size*. AA and AAA sized batteries are easily available world-wide while kcrv3 cells may be more difficult to purchase overseas. New product designs targeting the travel customer segment might target the holonym *size* rather than the more general concept *battery type*.

The most common approach to ontology development involves expert intervention. Unfortunately, product categories evolve over time. When evolution is compounded by the breadth of product variety, relying upon experts in multiple domains becomes impractical. Thus,

we seek to develop a fully automated, unsupervised process for learning an ontology of product-specific features from a set of online customer reviews about that product. In this paper, we describe a four-step process for ontology induction from online customer reviews:

- Preprocess customer reviews (stop words, stemming, etc) into lists of phrases.
- Cluster phrases (hyponyms) using a variation of standard k-means.
- Generate subclusters (hyponyms) using a novel application of constrained logic programming (CLP).
- Derive ‘parts’ meronyms and/or holonyms from the output of the CLP.

We test the process by inducing a feature ontology for digital cameras based upon several thousand online customer reviews. The induced features are compared to pre-existing manufacturer product information and retail buying guides.

The evaluation highlights three distinct characteristics of our approach. First, our approach is unsupervised, utilizing no human intervention. By contrast, much of the prior work on ontology induction leverages supervised machine learning methods, requiring hand-coded training data. We do discuss supervised extensions which might increase accuracy. However, unsupervised methods reduce the amount of expert intervention required to automatically process product information.

Second, the process is domain independent, relying upon shallow natural language processing (NLP) techniques rather than domain specific assumptions about particular products. Our objective is to develop a general process, applicable across heterogeneous product categories and capable of distinguishing diverse features in a domain independent manner. The evaluation in this paper is limited to a single product category, that of digital cameras. A natural step for future work is to extend the evaluation to multiple product categories; we address this in the discussion below.

Third, the system is highly parameterized. Although we use k-means for the initial clustering, the steps are separable. Our CLP approach can accept input from any initial clustering, including non-disjoint clusters.

In the remainder of this paper, after surveying the related literature in Section 2, we address each step of the process in Sections 3-6. We describe our data set of customer reviews and evaluation metrics in Section 6. A discussion of results and future work conclude in Section 7.

## **2. RELATED WORK**

There is a large body of work on the topic of ontology induction that appears in different disciplines under different names. Linguists learn specific taxonomic relationships such as hyponyms and meronyms. Others use the term 'concept hierarchy.' 'Clustering categorical data' defines a third body of similar research. Although the terminology may differ, the underlying objective is the same: to group terms into concepts and to identify different types of relationships between concepts. We can loosely separate the literature based upon whether the source data being categorized is relational or not. The different approaches will also vary in the degree to which they are supervised or unsupervised.

### **2.1 Non-relational data.**

Most of the approaches to ontology induction that begin with non-relational data identify two distinct processes: extract data and then learn relationships. Supervision may be introduced in either of the two processes.

One approach is to begin with a single seed ontology and then to grow that ontology by using supervised learning techniques to extract data from relevant Web pages. Missikoff and Navigli [27] begin with a generic reference ontology and then extract from domain-specific web pages to

tailor the resulting ontology. By contrast, Modica, Gal and Jamil [28] seed their ontology with a single, representative Web source. They use the source HTML to define concept names and the underlying Document Object Model (DOM) to define the initial relationship between concepts. Incremental refinement is managed using techniques borrowed from the database schema matching research literature. In particular, they use unsupervised techniques such as term similarity, concept (instance) matching, and external sources such as thesauruses.

Rather than explicitly separating extraction from concept learning, it is possible to combine the two steps to define extraction patterns that are explicitly indicative of a particular conceptual relation. Hearst did early work applying this approach to learning hyponym relationships [17]. Maedche and Staab [23, 25] use lexical analysis and syntactic patterns to extract sets of candidate hyponym concept pairs. They then apply frequent item-set analysis [3]. The confidence and support measures from association rule mining are used to estimate the validity of a particular concept pair. Instead of association rule mining, KnowItAll [9, 32] draws upon external sources such as WordNet and labeled training data to assess the validity of relationships. Specifically, the frequency of a term pair is used to calculate the probability that the pair is an instance of a particular sub-class relationship.

While Maedche and Staab use generic extraction rules that are indicative of class relationships in many different domains (e.g. *X such as Y*), Finkelstein-Landau and Morin [10] explicitly draw on supervised techniques to learn domain-specific patterns of hyponyms. They identify a training set of specific sentences containing relationship instances from which to build their extraction patterns. Instead of an automated concept assessment technique, Finkelstein-Landau and Morin rely upon iteration with experts to evaluate the resulting concept pairs.



Finkelstein-Landau and Morin contrast their supervised approach with an unsupervised technique that returns to the earlier separation of extraction from relationship learning. Extraction is performed using shallow NLP techniques such as tokenizers, Parts-of-Speech (POS) labeling, and chunkers. Relationships are defined based upon statistical measures of co-occurrence both within a single document and between different documents.

Like Finkelstein-Landau and Morin's unsupervised approach, we separate "information extraction" from relationship learning. While we do rely upon lexical analysis to clean the data, by focusing exclusively on customer review data and using a very narrow representation, we greatly simplify the extraction step. In this way, we do not need a seed ontology like Missikoff and Navigli and we do not explicitly need to leverage the source HTML or underlying DOM as Modica, Gal, and Jamil do. At the same time, we believe that the heterogeneity of products reviewed in online forums present a reasonable test for the robustness of our approach.

More significantly, many of the lexico-syntactic rules used for extracting hyponym concept pairs are less applicable in the domain of customer reviews where grammatical conventions might break down and text, as in the case of lists of Pros or Cons, are simply lists of phrases with no associated syntactic or semantic context. A customer might simply write that "NiMH rechargeables are great" without reference to the concept's hypernym. As a consequence, techniques to learn syntactic patterns that are representative of hyponym relationships like Hearst or Maedche and Staab might prove less applicable.

By contrast, phrases in customer reviews often encode 'part-of' meronym relationships. We leverage co-occurrences within a single review as a heuristic for discarding certain tokens and discuss the use of co-occurrences between reviews as a weighting factor to relax some of our stronger assumptions about cliques.

## 2.2 Relational data

Processing relational data greatly facilitates the use of unsupervised techniques by leveraging knowledge of the structure to simplify the task.

Han and Fu [15] and Lu [22] develop routines for structuring both numerical and nominal data. For numerical data, they use binning strategies to induce categories within a single attribute domain. For example, how to categorize different levels of digital camera resolution ranging from less than 1 megapixel to more than 8 megapixels. In binning, one categorizing criteria might be to evenly distribute the domain into a pre-defined number of categories. For nominal data, they do not categorize the values within a particular domain. Instead, they attempt to learn the hyponym relationship between attribute domains. Based upon the cardinality and the frequency histogram of each attribute domain, they define a partial order on the attribute domains that define the hierarchical relationship between domains.

Suryanto and Compton [32] begin with a database in the form of a rule-based classifier. Their objective is to define a taxonomy on the classes that are identified by the classifier. They distinguish between subsumption (the hyponym relationship), mutual exclusivity (disjoint subclasses), and similarity. The intuition is that concepts with similar classification rules are similar and the derivation to subclasses follows from transitivity of the rules.

Ganti, Gehrke, and Ramakrishnan [11] as well as Gibson, Kleinberg, and Raghavan [12] assume a relational table and attempt to find subcategories within a particular attribute domain  $D_i$ . This differs from the work by Han and Fu and Lu, which define orderings between attribute domains. Ganti et al. define the 'inter-attribute' summary as a weighted graph where all values of all domains in the relation are vertices in the graph. The weight of an edge between  $a_i$  in  $D_i$  and  $a_j$  in  $D_j$  where  $i \neq j$ , is the number of tuples in which  $a_i$  and  $a_j$  appear. In addition, they assume

a priori knowledge of the relation and use that knowledge to calculate the 'intra-attribute' summary between two attribute values in the same domain. Finally, their relations have no nulls and the mapping of every value to its corresponding attribute domain is known. Gibson et al. use the same relational assumptions but instantiate multiple instances of a vertex-weighted graph which they call basins. They iterate over the set of basins until achieving a fixed-point where vertex weights are separated into distinct positive and negative subsets defining the domain subclasses.

Contrasting the work that assumes *a priori* knowledge of the relations, our CLP approach bears similarities to schema induction. Using the metaphor of learning relational schemas, each concept is a relation; the corresponding phrases, extracted from the text of customer reviews or product descriptions, are tuples of the relation. Finding the largest maximal clique defines the relational schema. However, our tuples have many nulls. Thus, though we begin with a graph representation that is similar to Ganti's, our graph is incomplete. Furthermore, we exploit the graph in a different way. First, we constrain the principle of connectedness with that of logical assignment to cluster the tuples of a relation rather than categorizing the values of a single attribute domain. Second, we cannot leverage intra-attribute knowledge because the schema is unknown a priori. To the degree that we can induce a relational structure, Gibson et al. and Ganti et al. offer a complementary strategy to our approach for learning hierarchical sub-class relations (hyponyms) within a domain. Where Han and Fu and Lu define a partial order over the attribute domains in the schema, we treat the attribute domains as part-of meronyms. In general, neither case is always true.

### **3 PRELIMINARIES**

In this section, we describe two different data representations used to model an individual review and a collection of reviews respectively. To define the representations, we first outline the shallow NLP steps used to transform a set of reviews into the atomic units (tokens) used to construct the models.

### 3.1 Reviews into tokens

Abstractly, each customer review is a list of phrases where each phrase represents a distinct feature of the product being reviewed. Phrases themselves are composed of atomic tokens. To minimize the complexities of NLP and information extraction, we construct the token set as the union of all Pros and Cons listed directly by consumers as part of each review (see Figure 1). The Pros and Cons in each review are intermixed because, for ontology construction, we only seek to identify concepts of interest. Whether the feature is a strength or weakness of a particular product is not important. This contrasts with earlier work on reviews that explicitly seeks to learn sentiment terms [18; 29]. An excerpt from the list of all Pros and Cons, including those from the review in Figure 1, is provided in Figure 2.

Each phrase is tokenized into its constituent words. Because phrases are composed of tokens taken from product reviews that are directly input by customers, the raw text includes



Figure 1. Epinions Customer Review Page

misspellings and other inconsistencies that can lead to spurious elements in our data models. Therefore, we actively prune tokens using a number of shallow NLP heuristics.

Beginning with a text string, we parse on list separators including commas, slashes, or

- Does not work with Windows XP
- LCD Display
- T.V. outputs
- Only 2x zoom
- Not Support on Windows 2000
- Not compatible with Windows 2000
- Works with Windows 98
- No Windows XP support
- Windows XP compatible
- Weak Mac Compatibility
- Mac Compatible Battery
- Not fully Mac compatible
- Lack of Mac support
- Marginal technical support

**Figure 2. Excerpted list of Pros and Cons**

semicolons. We apply a simple heuristic to count list separators and assume that multiple instances of a separator correspond to a list (e.g. two or more commas) while single instances are ignored.

For each resulting phrase in the text string, we apply the following filters:

- Discard stop words to reduce noise
- Convert all tokens to lower case to address

capitalization and proper nouns (e.g. 'Microsoft Windows' is a required operating system for some digital camera software)

- Apply Porter stemming to address plural forms as well as simple misspellings. Note that introducing a dictionary, thesaurus, or Wordnet-type variants might also be of use. However, in this initial work, we wanted to limit the use of external sources that might introduce domain dependence. In this way, we focus on the robustness of domain-independent lexical and graph-based processing.
- Discard any phrase that contains a non-alphanumeric or punctuation feature not eliminated by parsing phrases. Special exceptions were generated by hand for decimal numbers and hyphenated terms. Examples of discard phrases taken from the digital camera product feature *Computer Requirements* include: 'ibook(tm),' 'windows®98 second edition (se),' and 'windows 98\*.' The intuition is that our data set ranges from several hundred to several thousand phrases depending upon the starting feature concept. Therefore, we can safely

discard outlier phrases. Discarding does raise the question of an optimal sample size of phrases; we revisit this issue in the Evaluation and Discussion sections below.

For example, the review in Figure 1 yields Cons such as "Only 2x zoom" and "does not work with Windows XP." Using our shallow NLP techniques, these Cons are reduced to the token phrases "2x zoom" and "window xp" respectively. The complete list of token phrases corresponding to the Pros and Cons from Figure 2 is depicted in Figure 3.

### 3.2 Modeling a set of reviews in a phrase by token matrix

Borrowing from the information retrieval community, the phrases from all reviews are aggregated into a single phrase by token matrix. Each phrase is represented as a vector using the well-known vector-space model.

More formally,  $i \in I$  is a token in the set of all tokens.  $j \in J$  is a phrase. A phrase is simply a finite sequence of tokens and  $J$  is a subset of the set of finite token sequences  $\{ \langle i \rangle \mid i \in I \}$ . We may then construct a phrase by token matrix as:  $\text{Matrix}(j,i) = (TF_{ij} \times IPF_i)$ . The matrix modifies

- |   |   |
|---|---|
| <ul style="list-style-type: none"> <li>• work window xp</li> <li>• lcd display</li> <li>• t.v. out</li> <li>• 2x zoom</li> <li>• support window 2000</li> <li>• compat window 2000</li> <li>• work window 98</li> <li>• window xp support</li> <li>• window xp compat</li> <li>• mac compat</li> <li>• mac compat batt</li> <li>• fulli mac compat</li> <li>• lack mac support</li> <li>• margin technic support</li> </ul> | <p>the traditional term-frequency inverse-document-frequency (TFIDF) vector space model for clustering documents [33].</p> <p><math>TF_{ij} = T_{ij} \times P_j</math> counts the total number of occurrences of token <math>i</math> in the entire collection. <math>T_{ij}</math> counts the frequency of token <math>i</math> in phrase <math>j</math> and <math>P_j</math> counts the occurrences of phrase <math>j</math>. Following the</p> |
|---|---|

**Figure 3. Token phrases corresponding to the Pros and Cons in Figure 2**

standard TFIDF model, it seems intuitive that a token might appear in so many different phrases

that the token proves less useful for discriminating between phrases. Hence, we introduce  $IPF =$

$\log(N/n_i)$  as a weighting factor to correct for tokens that are more widely distributed.  $N = \sum P_j$  ranges over all  $j$  and counts the total number of phrases. Likewise,  $n_i = \sum \min(1, T_{ij}) \times P_j$  counts the total number of phrases containing token  $i$ . Recall that a token may repeat within a phrase hence we take  $\min(1, T_{ij})$ .

### 3.3 Modeling a set of reviews as a graph

We use the phrase by token matrix to cluster the token phrases into an initial set of hyponyms using a clustering process described in Section 4. However, we then transform each individual cluster of token phrases into a second representation to derive meronyms and holonyms.

The token phrases in each cluster are then transformed into a separate graph. Each token represents a vertex in the graph and edges are defined by the token pairs in the phrase  $(v_i, v_j) \ i \neq j$ . For example, considering only the token phrases in Figure 3, our initial matrix-based clustering might yield the token phrase clusters in Figure 4. The corresponding graphs are depicted in Figure 5.

More formally, a graph  $G = (V, E)$  is a pair of the set of vertices  $V$  and the set of edges  $E$  [2].

An *edge* in  $E$  is a connection between two vertices and may be represented as a pair  $(v_i, v_j) \in V$ .

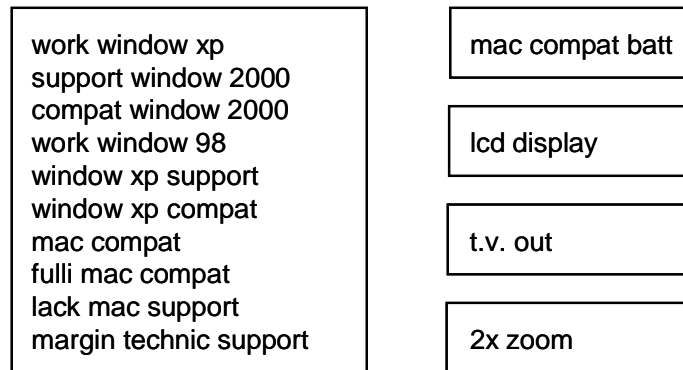


Figure 4. Clustering the token phrases from Figure 3 into hyponyms

A path is a list of vertices in  $V$  where there is an edge in  $E$  from each vertex to the next vertex in the list. A graph is *connected* if there exists a path between any two vertices in the graph.  $S = (V', E')$  is a subgraph of  $G$  if  $(v \in V' \rightarrow v \in V)$  and  $(v_i, v_j \in E') \rightarrow (v_i, v_j \in E \text{ and } v_i, v_j \in V')$ .  $G_i$  is a *maximally connected subgraph* of  $G$  if  $G_i$  is a connected subgraph  $S = (V', E')$  and for all  $v_j \in V$  and  $v_j \notin V'$ , there is no  $v_k \in V'$  where  $(v_j, v_k) \in E$ . The connected components of  $G$  are the set  $G_1 \dots G_n$  of maximally connected subgraphs of  $G$ . By definition, for any two maximally connected subgraphs  $G_i$  and  $G_j$   $V_i \cap V_j$  is empty and  $G_1 \cup \dots \cup G_n = G$ .

A *complete graph*  $G = (V, E)$  is a graph for which there is an edge between every pair of

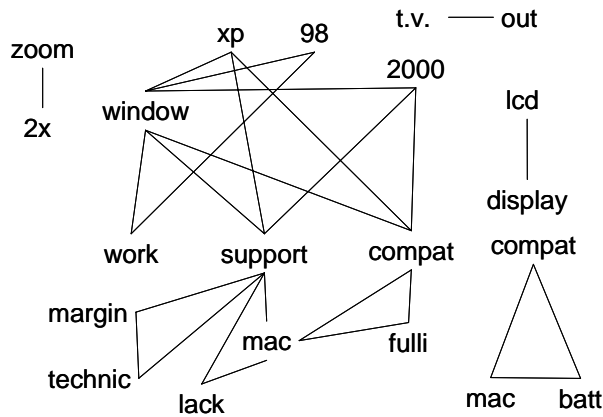


Figure 5. Token graphs for the clusters in Figure 4

vertices.  $\forall v_i, v_j \in V, v_i, v_j \in E$ . A *clique*  $S$  is a complete subgraph of  $G$ .  $S' = (V', E')$  is a *maximal clique* of  $G$  if  $S'$  is a clique and there is no  $v_j \in V$  and  $v_j \notin V'$  such that  $\forall v_k \in V', (v_j, v_k) \in E$ .

#### 4 CLUSTERING PHRASES

In this section, we describe the use of unsupervised text clustering techniques to induce hyponyms from the phrase-token matrix. The general approach is to begin with the assumption



that every phrase represents a unique concept (product feature). We then separate phrases into disjoint clusters so that all phrases representing a particular feature are grouped together. Each cluster then represents a distinct hyponym (or subclass) of the generic root concept *product feature*. For example, we assume that each cluster in Figure 4 represents a separate feature.

More formally, given the matrix  $(j,i)$  over the set of phrases  $J$  and the set of tokens  $I$ , we seek to separate phrases into a set  $C$  of  $k$  concept clusters  $\{c_i | c_1 \cup \dots \cup c_k = J \text{ and } \forall i,j \ i \neq j \ c_i \cap c_j = \emptyset\}$

The clustering is necessarily dependent upon and susceptible to the quality of phrase parsing. Poor parsing (e.g. a single phrase that combines multiple features) can introduce noise into the resulting clusters. However, our objective is to capture all phrases corresponding to a particular feature in one cluster. It is worth noting that a feature/concept can be distinguished at somewhat arbitrary levels of granularity. Thus, it is possible that ‘digital zoom’ and ‘optical zoom’ could be clustered as distinct features or aggregated as the single concept ‘zoom.’

Clustering algorithms are typically distinguished by their means for measuring similarity and their metric for separating clusters. Similarity may be measured by edit distance, Manhattan distance, Euclidean distance, etc. depending upon the data representation and desired characteristics. Likewise, clusters may be formed by minimizing distance to a cluster median, maximizing cluster density, etc. In this work, complementing our matrix representation of phrases and tokens, we use the *cosine* measure of angular distance between vectors to calculate similarity. This cosine measure is then applied to the well-understood k-means clustering algorithm that is both fast and easy to calculate.

#### **4.1 Distance measures**

The *cosine* distance measure of two vectors calculates the dot product of the inputs and returns a value between 0 and 1 where a 0 indicates that the two vectors are orthogonal and a 1 indicates that the vectors are identical.

Given the set of phrases  $J$  and the set of tokens  $I$ , let  $w_{ji}$  be a *TFIDF* weight in  $Matrix(j,i)$ .

Then, we can say that cosine distance between any two phrases  $m,n \in J$  is:

$$\cos(p_m, p_n) = \frac{\sum_{\forall i \in I} w_{mi} \bullet w_{ni}}{\sqrt{\sum_{\forall i \in I} w_{mi}^2} \sqrt{\sum_{\forall i \in I} w_{ni}^2}}$$

Note that the cosine distance is simply the dot product of the vectors for each phrase where each vector is normalized to unit length by the denominator.

## 4.2 K-means

K-means is a well-studied unsupervised technique for clustering data. K-means begins by randomly identify  $k$  phrases as cluster centers. Every phrase is then assigned to the ‘most similar’ center. We use the cosine distance measure for measuring similarity. The algorithm then calculates a centroid (the mean) for each cluster producing a new set of  $k$  cluster centers. The process of (re)assigning phrases to the closest center and then (re)calculating centers is repeated until the centroids stabilize. Note that this process is identical to maximizing the pairwise similarity between phrases within a single cluster. For a cluster  $c_i \in C$ , k-means seeks

to maximize:  $\frac{1}{|c_i|^2} \sum_{v,v' \in c_i} \cos(v, v')$ . Although k-means is guaranteed to converge, it is known to be extremely sensitive to its initial conditions. Consequently, we repeat the algorithm multiple times, beginning with a new, random set of  $k$  centers on each iteration.

The quality of a cluster can be calculated by the sum of the similarities between each vector in the cluster and its centroid. This is more simply calculated as the length of the centroid.

Formally, the quality of a cluster  $c_i \in C = \sum_{\forall v \in c_i} \cos(v, \text{centroid}(c_i)) = |\text{centroid}(c_i)|$  [39]. The quality of a clustering  $QC$  is then simply the sum  $QC = \sum_{\forall c_i \in C} |\text{centroid}(c_i)|$ . To pick the best clustering from among the repeated k-means trials, one picks the iteration that maximizes  $QC$ .

## 5 CONSTRAINED LOGIC PROGRAMMING

Where we use k-means, we might have used any of a number of unsupervised techniques to produce phrase clusters that approximate hyponyms. All the phrases referring to a particular product feature are thus gathered in a single set.

However, a product feature may itself be described by distinct components. The meronyms and holonyms that describe the memory storage in a consumer electronic, for example, might include such features as capacity (4 mb, 8 mb, 16 mb) memory type (memory stick, compact flash, xD), and whether the memory is included with the device or requires separate purchase. To partition the holonyms and meronyms underlying a product feature, we introduce a novel, constrained logic programming (CLP) approach.

### 5.1 Clustering as assignment

To cluster the tokens within a hyponym, we treat each meronym and holonym as an attribute or column of a database relation. Each phrase is a set of tokens, one row or tuple of the relation. Clustering tokens involves an assignment process. For each phrase in the relation, assign the constituent tokens to the corresponding attribute domain or relational column. The resulting columns define each meronym or holonym.

Formally, we may say that a feature is constructed from  $k$  meronyms and holonyms. Each meronym or holonym names a Domain ( $D$ ). Each domain  $D$  is defined by a set of tokens that

includes the value NULL. The Cartesian product of domains  $D_1 \dots D_k$  is written  $D_1 \times \dots \times D_i \times \dots \times D_k$ , and is the set of all  $k$ -tuples  $\{t_1 \dots t_k \mid t_i \in D_i\}$ . Each phrase is simply one such  $k$ -tuple and the set of all phrases in the cluster simply defines a relation in extension, a finite subset of the Cartesian product. A relational schema is simply a mapping of meronym and holonym names  $A_1 \dots A_k$  to domains  $D_1 \dots D_k$  [38].

Taken in this light, when beginning with an initial set of phrases, the task of clustering tokens into meronyms and holonyms is reduced to two steps: one, determine how many meronyms and holonyms there are and two, assign tokens to their corresponding meronym or holonym domains. More generally, we may treat this as a relational learning problem. Given a relational instance, we must first induce the relational schema and second, assign the values in each relational tuple to their corresponding attribute domains.

## **5.2 Learning relational schema.**

The first task is to learn how many meronyms and holonyms there are. The (strong) hypothesis is that if the data set is large enough, we can discover at least one instance of the Cartesian product of all attributes that define the feature. Colloquially, there may not be a single phrase that includes an instance of all feature attributes, but transitive relationships (edges between different phrases) could complete the relation.

We search for a relational instance, one element of the Cartesian product, by searching for a maximal clique within the token graph for each hyponym. Because the general problem of finding a maximum clique is NP-complete, we adopt a simple heuristic for finding a maximal clique that is presented as Figure 6 [4, 35]. We iterate repeatedly, selecting the largest such maximal clique.

```

# input    g: the graph as a list of vertex edge pairs
#         v1: as the first vertex in the clique
gen_maximal_clique(v1, g):
    v_list = edge_list(v1, g) # pairwise adjacent vertex of v1 ordered by degree.
    c_list = edge_list(v1, g) # candidate vertices for addition to the clique
    clique = [v1] # initialize the clique
    return iterate_clique(clique, c_list, v_list, graph)

iterate_clique(clique, c_list, v_list, graph)
    if there are no candidates to add and no vertices to check, return c
    v_car is the first vertex in v_list
    v_cdr is the remainder of v_list
    e_car is the edge list of v_car
    if v_car is in c_list, v_car is adjacent to everything in the clique so far
        add(v_car, clique)
        update c_list to adjacent nodes of the new clique: intersect(c_list, e_car)
        recurse w/ v_cdr
    else recurse w/ v_cdr

```

**Figure 6. Algorithm to find a maximal clique [35]**

As an example, one of the clusters in Figure 4 corresponds to the hyponym *PC support*. The corresponding graph in Figure 5 yields a maximal clique "compat window 2000". The maximal clique (which happens to be the maximum clique in this case) is represented by a token for whether a particular PC operating system is supported, the operating system (windows) and the version (2000, xp, 98). The clique is represented in the first row of Table 7.

Note that the process of discovering a clique does not assign names to each attribute so the clique is not precisely a schema. In many cases, concept names will naturally emerge from the assignment to follow. Note also a limitation of the strong assumption regarding the maximal clique. In Table 7, we can see how ‘support’ is logically forced into the same concept/column as ‘compatibility.’ We take up this issue in the assignment algorithm next.

Maximal clique			compat	window	2000
Meronym/holonym concepts names	Pros and Cons	Token phrases	supported	OS	version
Token assignment	does not work with windows xp	work window xp	work	window	xp
	not support on windows 2000	support window 2000	support	window	2000
	not compatible with windows 2000	compat window 2000	compat	window	2000
	works with windows 98	work window 98	work	window	98
	no windows xp support	window xp support	support	window	xp
	windows xp compatible	window xp compat	unsupport	window	xp

**Table 7. Meronym/holonym relationships for features of the hyponym PC support**

### 5.3 Assigning to meronyms and holonyms

Having derived a representative  $k$ -tuple by searching for a maximal clique, we now seek to construct the corresponding meronyms and holonyms by assigning each token to its corresponding attribute domain. Given a cluster of phrases, we first generate the set of all tokens in the cluster and then find a maximal clique for the corresponding token graph. We assign tokens to attributes subject to the mutual exclusivity constraints represented by each phrase.

Each phrase represents  $\binom{n}{2} = \frac{n!}{2!(n-2)!}$  pairwise constraints where  $n$  is the number of tokens in

```

process_phrases(p_list)
[1]  schema = find_maximal_clique(p_list)
[2]  order phrases by length
[3]  for each phrase p:
[4]      # initialize data structures
[5]      mutual_ex_d - for each tok, all mutually exclusive tokens
[6]      boundary_d - for each tok, valid candidate assignments
[7]      assign_d - for each tok, the category assignment
[8]      # propagate the constraints for each successive phrase
[9]      boundary_d, mutual_ex_d, assign_d =
[10]         propagate_bounds(phrase, boundary_d,
[11]            mutual_ex_d, assign_d, schema)

```

**Figure 8. Process\_phrases as a CLP**

the phrase and each pair ensures that two tokens that co-occur in the same phrase may not appear in the same attribute domain. These pairwise constraints are consistent with disjoint clustering and assume the absence of multi-valued attributes within a single phrase.

Our approach (see Figure 8) is to frame the problem as a constrained logic program and to resolve the problem using a bounds consistency approach. We define the assignment problem using a maximal clique. In the bounds consistency approach, we invert each mutual exclusivity constraint (*mutual\_ex\_d*) and express the complementary constraint as a set of candidate assignments (*boundary\_d*). If the phrase constraints, taken together, are internally consistent, then the candidate assignments (*assign\_d*) for a given token are simply the intersection of all candidate assignments as defined by all phrases in the cluster containing that token.

We transform the mutual exclusivity constraint represented by each phrase into a set of candidate assignments using the algorithm in Figure 9. Note that we need only propagate the mutual exclusivity of tokens that are previously unassigned. Accordingly, for each unassigned token in a given phrase, the set of candidate assignments is the intersection of the possible assignments based upon the current phrase and all candidate assignments from earlier phrases containing the same token. We maintain a list of active tokens *boundary\_list* to avoid rescanning the set of all tokens every time the possible assignments for a given token is updated.

```

propagate_bounds(phrase, boundary_d, mutual_ex_d, assign_d, schema)
[1] # marshall prior assignments
[2] unassigned_tok = {t|t∈phrase ∧ t∉assign_d}
[3] unassigned_attr = {a|a∈schema ∧ ∀t(t∈phrase ∨ a∉assign_d[t])}
[4] for each t in unassigned_tok:
[5]     mutual_ex_d[t] = (t × (unassigned_tok - t)) ∪ mutual_ex_d[t]
[6]     possible_assign = {a|a∈(unassigned_attr ∩ boundary_d[t])}
[7]     boundary_list = {(t, [possible_assign])} ∪ boundary_list
[8]     recurse_boundary(boundary_list, mutual_ex_d, assign_d)

```

**Figure 9. Propagate boundary constraints**

For simple cases, the search for a clique and the subsequent assignment is quite straightforward. The corresponding meronym or holonym relationships for the phrases of the hyponym *PC support* are included in Table 7. Beginning with the literal phrases in column 2, the table illustrates how logical phrase constraints naturally cluster tokens into meronyms and holonyms.

As another example, table 10 shows excerpted results from applying the maximal clique and the assignment to the hyponym cluster for *memory*. Rather than showing Pros and Cons as well as the token phrases, Table 10 simply lists the token assignments to each meronym or holonym. The tabular representation emphasizes the Cartesian product interpretation of this clustering. Even though a specific *k*-tuple may not correspond to a literal phrase from a customer review, it still implies a semantically meaningful phrase. For example, customer comments about a "4 mb compactflash card" may fail to mention whether the external memory is "included" or not. The

inclusion or exclusion of external compactflash cards is still a valid parameter when discussing compactflash. As illustrated in the last two rows of Table 10, it is also possible for phrases to omit values for one or more meronyms or holonyms. This is consistent with the notion of NULL values in the corresponding relational attribute domains.

Maximal clique	memori	capac	compact	flash	stick	suppli	camera
Meronym/holonym concepts names	subjective	unit	quantity	form	type	Included	residual
Token assignment	memori media skimpi excess	capac size wimpi mb	compact 8 4 16 32	stick card	flash compactflash smart meg standard	suppli include come inadeq	camera 4mb small

**Table 10. Meronym/holonym relationships for hyponyms of the feature Memory**

Unfortunately, the data are not always internally consistent. There are many valid reasons that semantically-inconsistent or contradictory constraints might appear: Poor parsing, the legitimate appearance of one token multiple times within a single phrase (e.g. the phrase ‘digital zoom and optical zoom’ duplicates the token ‘zoom’) or even inaccuracies by the human reviewers who write the text that is being automatically processed. For example, in Table 10, the phrase ‘compact flash’ reduces to two tokens: ‘compact’ and ‘flash’ while ‘compactflash’ is a single token. The implications range from the noisy clusters in Table 10, to logically unsatisfiable assignment problems.

Conversely, the problem may be too unconstrained. If there are too few phrases in the cluster, the assignment could have multiple valid solutions. Our approach would certainly accommodate the introduction of additional constraints such as matching synonyms or checking data-types; but adding such constraints would require domain-specific knowledge, limiting the generalizability of our approach.

Finally, it is worth noting the vulnerability of our CLP approach to noise from the initial clustering of phrases into hyponyms. We first attempt to cluster all phrases that define the same



product feature together; but we rely upon a token vectors representation. These tokens include homonyms and certain adjectives (e.g. ‘great’, ‘good’) that may modify multiple features. Consequently, a single cluster may include multiple features. The assignment of tokens to meronyms and holonyms must therefore account for logical inconsistencies in the constraints as well as noise from multiple product features within the same cluster.

### 5.4 Refining the assignments

To address the problem of robustness in the face of logical inconsistencies and noisy phrase clusters, we adapt our CLP approach to generate phrase sub-clusters in the process of logical assignment. Our intuition is to use a CLP to cluster a set of phrases into logically consistent, fully-constrained, satisfiable sub-clusters. Each sub-cluster represents a distinct hyponym within which we find a maximal clique and assign tokens.

For example, the reader may have noticed that our assignment in Table 7 omitted those Pros and Cons related to Macintosh system support. The problem is that the CLP of Figure 8 does not provide unambiguous assignments for all tokens in the "computer support" graph of Figure 5.

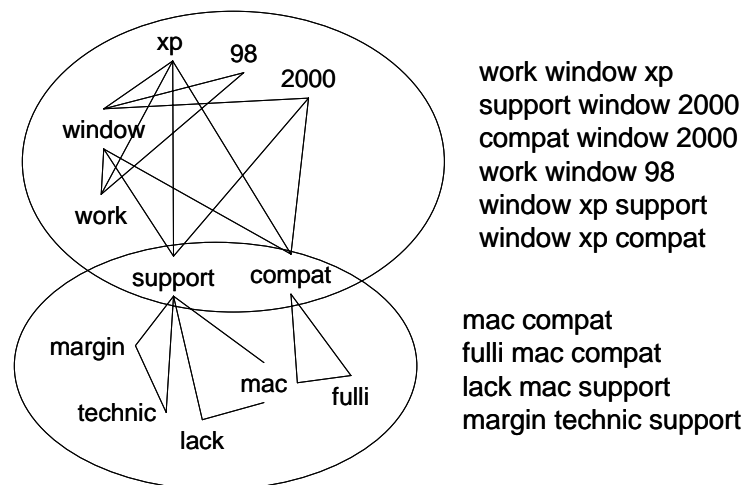


Figure 11. Clustering the token phrases of one hyponym using CLP

By clustering all tokens phrases containing one or more ambiguous tokens, we can define a token sub-graph, find a new maximal clique, and generate new token assignments. As depicted in Figure 11, the CLP not only assigns tokens but effectively partitions the graph into sub-graphs by discovering fully-constrained token phrase subsets. In Figure 11, the CLP produces one partition for Macintosh computers and one for Windows-based PCs.

Specifically, we extend our general approach from Figure 9 recursively. On every iteration, we process all remaining phrases. Any token phrase that proves logically inconsistent, contradicts a prior assignment, or is partially unconstrained, is appended to a *residual\_list* upon which we recurse. Every subsequent iteration produces a new sub-cluster.

There are three logical inconsistencies that we account for. When the boundary conditions are empty, the active token has no valid assignment. This occurs when the prior boundary conditions have no intersection with the constraints from a new phrase. The second inconsistency entails when a token assignment violates a prior assignment. A token can violate a prior assignment in at least two ways. First, the same token might appear twice in a phrase. Second, different phrases might transitively assign the same token. In either instance, a single token is forced into different assignments. The third inconsistency involves transitivity. A phrase may transitively assign a token into a violation of pairwise mutual exclusivity established by an earlier phrase(s). In addition to an extra check for prior assignments in *propagate\_bounds* (see: Figure 9), we introduce checks for logical inconsistencies into *recurse\_boundary* depicted in Figure 12 and called from *propagate\_bounds*.

Having subdivided our initial clusters based upon logical constraints, we still need to separate meaningful sub-clusters from noise. We assume that sub-clusters represent distinct product

```

recurse_boundary(boundary_list, boundary_d, mutual_ex_d, assign_d)
[1] if boundary_list = ∅: return
[2] else:
[3]     (active_tok, possible_assign)= car(boundary_list)
[4]     boundary_list = cdr(boundary_list)
[5]     if len(possible_assign) > 1: # > 1 possible assignment
[6]         boundary_d[active_tok] = possible_assign ∩ boundary_d[active_tok]
[7]     elif len(possible_assign)≤∅: RAISE ERROR # over constrained
[8]     else: # assumes len(possible_assign) == 1
[9]         # check for constraint violations e.g. mutual ex, prior assign
[10]        if mutual_ex_d[active_tok] ∩
[11]            {t|assign_d[t]= possible_assign}≠∅: RAISE ERROR
[12]        delete boundary_d[active_tok]
[13]        update assign_d[active_tok] = car(possible_assign)
[14]        for each t in mutual_ex_d[active_tok]:
[15]            boundary_d[t]= boundary_d[t]- possible_assign
[16]            b_list_prime = [(t, [boundary_d[t]])]∪ boundary_d[t]
[17]            # if t is already on boundary_list, keep newest(t,boundary)
[18]            boundary_list = boundary_list ∪ b_list_prime
[19]    recurse_boundary(boundary_list, boundary_d, mutual_ex_d, assign_d)

```

**Figure 12. recurse\_boundary: checking for logical inconsistencies**

features. Accordingly, though drawn from the same initial cluster, meaningful sub-clusters should have only minimal token overlap.

With this in mind, we apply a two-stage statistical filter to further filter noisy clusters. First, because we will ultimately separate each sub-cluster into token sets of meronyms and holonyms, we do not want to pick sub-clusters that represent too small a percentage of the overall number of tokens. Second, we assume that each sub-cluster comprises a (predominately) disjoint token subset. If the tokens in a sub-cluster appear in no other sub-cluster, then the sub-cluster token frequency should match the frequency of the initial cluster; likewise, the sub-cluster token order should match the relative order of the initial cluster.

The first stage of our statistical filter is evaluation of the  $\chi^2$  statistic, comparing each sub-cluster to its corresponding initial cluster. If the initial cluster has a cumulative token count  $N_1$ , a sub-cluster has a cumulative token count  $N_2$ , and a token  $t$  has respective counts  $o_{t1}$  and  $o_{t2}$ , then

for cluster  $i = [1,2]$  we may write  $e_{ii} = \frac{N_i \times (o_{t1} + o_{t2})}{N_1 + N_2}$ . Then  $\chi^2 = \sum_{\forall t,i} \frac{(o_{ii} - e_{ii})^2}{e_{ii}}$ .

Although there is no hypothesis to be tested per se, there is a history of applying the  $\chi^2$  statistic in linguistics research to compare to text corpora with a measure that weights higher-frequency tokens with greater significance than lower frequency tokens [20]. In our case, we set a minimum threshold on the  $\chi^2$  statistic to ensure that sub-clusters reflect an appropriate percentage of tokens from the initial cluster.

After filtering out sub-clusters that do not satisfy the  $\chi^2$  threshold, we use the same cluster token counts to calculate rank order statistics. We compare the token rank order from each sub-cluster to that in the corresponding initial cluster using a modified Spearman rank correlation coefficient ( $r_s$ ). Spearman is the normalized sum of the squares of the differences in the rank orderings. For the  $n$  tokens in the sub-cluster:

$$r_s = 1 - \frac{6 \sum_{i=1}^n i - \text{rank}(c_2, c_{1i})}{n(n^2 - 1)}$$

where  $\text{rank}(c_2, c_{1i})$  is the rank in cluster 2 of the  $i$ -th ranked token in cluster 1. As a minor extension, where  $c_2$  represents the initial cluster, we use the relative token rank meaning we maintain token order but keep only tokens in both  $c_1$  and  $c_2$ . We refer to this as the relative Spearman rank  $r'_s$ . Because of minimal token overlap the relative token ordering should not match exactly but should exhibit significant positive correlation e.g.  $0.5 < r'_s < 1$ .

Note that when comparing text corpora, a difference in rank of one place amongst the top ten tokens should have more significance than a difference in rank of one place among tokens in the hundredth rank. Because Spearman weights these differences equally, it is considered less useful for evaluating corpus similarity [20]. However, recall that we assume that sub-cluster token order should closely match the relative order of the initial cluster because the sub-cluster tokens

are largely disjoint.  $r'_s$  therefore seems well-suited to our task. We select as significant those sub-clusters that maximize  $r'_s$ .

## **6. EVALUATION**

Early research in ontology induction limited evaluation to the subjective assessments of the researchers and/or domain experts [22, 27, 30]. Assessment might simply entail publishing representative results. Only more recently has research in ontology induction begun to develop more objective metrics. In this section, we report results from the application of our automated process to a real domain.

### **6.1 Experimental data set**

Our data set consists of 8,226 online digital camera reviews downloaded from Epinions.com on 5 July 2004. The reviews span 575 different products and product bundles that range in price from \$45 to more than \$1,000. The digital cameras range in resolution from 1MP to more than 6MP and vary in size from pocket-size to single lens reflex (SLR).

### **6.2 Reference data for evaluation**

For an automated system that is intended to capture knowledge from multiple, heterogeneous product domains, soliciting input from domain experts is not scalable. Instead, we compare our induced ontologies to publicly available expert-generated ontologies: online buying guides. We borrow from the literature to define metrics with respect to both the product features as well as the meronyms and holonyms by which individual product features are composed.

Modica, Gal and Jamil [28] use precision and recall to systematically document the effectiveness of different strategies at correctly incorporating new concepts from a representative data source into an existing hierarchy ontology. Popescu, Yates and Etzioni [32] develop

multiple techniques to extract and construct sub-class structures from text. They define their reference standard for correctness as the union of all sub-class structures from all techniques and use precision and recall as an internal consistency check.

We gather data from six online buying guides to serve as reference metrics for evaluation. The reference sources list a minimum of 14 product features and a maximum of 27. The average number of product features is 18. After processing our experimental data set, we compare the induced features and their corresponding attributes with each reference source. Precision asks, "how many of the induced features and components are actually used in professional reviews and online buying guides." Recall asks "how many of the features and components used in practice are automatically induced?" Finally, the F-measure, the weighted harmonic mean of precision and recall, provides a single, aggregate measure.

In addition, because the product feature space is inherently hierarchical, product features in one reference source are sub-classed in a different source. For example, *optical* and *digital zoom* are distinct product attributes in one reference and sub-classes of *zoom* in another. We define precision containment ( $P_C$ ) to include induced attributes that are subsets of the reference attribute. Likewise, recall containment ( $R_C$ ) includes reference attributes that are subsets of an induced attribute. We can also calculate the F-measure using  $P_C$  and  $R_C$ .

### **6.3 K-means phrase clustering**

Beginning with our set of customer reviews, we parse the Pro and Con lists as described in Section 3 to produce a phrase  $\times$  token matrix that is 14,081 phrases by 3,364 tokens. We then set  $k = 50$  and iterate k-means 10 times, selecting the best resulting cluster based upon  $QC$ .

The selection of  $k = 50$  ensured that, after applying the CLP, the number of clusters approximated the upper bound on the number of product features in our reference data.

However, in general, determining an optimal value of  $k$  is an open research question.

Practitioners appear to rely upon subjective measures of what is most appropriate for the domain at hand [37]. However, one can also apply quantitative measures. For example, we could calculate a quality metric that balances the number of clusters against the internal quality of each cluster [14; 15].

#### **6.4 CLP clustering and assignment**

Given an initial set of 50 clusters (from k-means), our next step is to search for cliques from which we can assign and subsequently sub-cluster. The CLP sub-clustering, which initializes a new sub-cluster upon encountering any logical contradictions, generates a total of 672 sub-clusters from the original 50. Applying the  $\chi^2$  threshold at 0.001 reduces the number of sub-clusters to 194. In addition, for convenience and efficiency, we eliminate any sub-cluster whose schema is of length less than or equal to two.

After applying  $r'_s$ , we are left with 47 sub-clusters. Though we might have expected 50 sub-clusters, one for each of the initial clusters, this is not the case. Some initial clusters contain only a single phrase and hence are not semantically meaningful. Moreover, other initial clusters yield multiple sub-clusters with the maximum  $r'_s$ .

Of the 47 sub-clusters, 11 are noisy clusters with no single, discernable product feature. We set these noisy clusters aside and revisit them below. The remaining 36 clusters are reasonably coherent based upon manual inspection. Of those, eight clusters are duplicates and a ninth cluster involves three similar clusters. This leaves 26 somewhat distinct features. We treat the duplicate clusters and the distinct features each in turn.

The existence of duplicate clusters is particularly interesting because they derive primarily from different initial clusters. It suggests both the existence of synonymous tokens within the

underlying data set as well as the need for more refined first-order clustering techniques. At the same time, duplicates also introduce the additional challenge of merging sub-clusters which is akin to the broader question of database or ontology integration. We revisit this question below.

The existence of distinct product features and the necessity of merging duplicates raises the question of naming these features. In this work, we apply a naïve naming convention: scan the sub-cluster assignment for singleton meronyms or holonyms. If there are multiple singletons, label the sub-cluster with a list of those singletons. Conversely, if there are no singletons, name the sub-cluster by the two or three tokens in a single meronym or holonym. Sometimes the resulting names are not entirely illuminating or interesting. For example, we would name the

red eye (flash)	manual (v. automatic)	view finder	options	features
shot (modes)	stick card (memory)	battery life	battery	lens cap
iso set noise	flash lag time	photo quality	cheap bit (design)	indoor
slow serial (pc)	relatively learn (easy to use)	easy read	take clip sound	zoom
body durable metal	hand shoot fit (ergonomic)	software	doesn't power (adapter)	
interface*	pocket size*	*name not taken directly from the cluster		

**Table 13. Induced digital camera features from online customer reviews**

sub-cluster in Table 10 as ‘stick card.’ Such a name appears more appropriate when we consider the additional context token ‘memory’ such as in ‘memory card’ and ‘memory stick.’

## 6.5 An ontology of digital camera features

The 26 features from our test data are listed in Table 13. Parentheticals in Table 13 are insertions to provide context to the feature name. The asterisked product features did not have a label that we could derive from the cluster itself.

It is worth noting that the application of  $r'_s$  as a filter does occasionally favor noisy clusters over more semantically meaningful ones. For example, digital camera *resolution* is missing from our feature list for this reason as is *battery type* and *PC support* (see Table 4). The absence of certain features is not necessarily significant in that they may be either so common or so outdated that consumers no longer ask after them. *Digital zoom*, for example, does not appear in



our induced data. At the same time, additional measures for better distinguishing meaningful clusters are still necessary.

To better evaluate the significance of the automatically induced features, we compare our induced features to those used by several prominent online buying guides. Table 14 exhibits performance that is not strong when compared to supervised information retrieval standards, where precision and recall can exceed 0.90. However, it is worth noting that ours is not an information retrieval task and that we use unsupervised, domain independent methods.

More importantly, our features are induced directly from customer reviews. Our product features may not closely align with those in the marketing materials for manufacturers and retailers; but rather than being indicative of an unsuccessful process, the inconsistencies may also signal a managerially significant disconnect between buyers and producers.

	Attributes	P	C <sub>P</sub>	R	C <sub>R</sub>	F	F <sub>C</sub>
Epinions.com	14	0.500	0.571	0.429	0.500	0.462	0.533
Cnet.com	17	0.211	0.263	0.211	0.316	0.211	0.287
megapixel.net	19	0.350	0.500	0.400	0.400	0.373	0.444
pcmag.com	14	0.429	0.571	0.429	0.643	0.429	0.605
viewz.com	27	0.371	0.407	0.370	0.481	0.370	0.441
BizRate.com	16	0.187	0.250	0.063	0.313	0.094	0.278

**Table 14. Comparing induced features to buying guide features**

While inconsistencies between induced features and those from online buying guides may still offer value, 11 of our final clusters are still too noisy to offer any immediate use. Tracing these uninformative clusters backwards reveals failures in the original clustering as well as conflicts within the constraints.

## 7. DISCUSSION AND CONCLUSIONS

In this paper, we have presented a system for automatically processing the text from online customer reviews. Phrases are parsed from the original text. The phrases are then tokenized and

clustered. A logical assignment problem further separates phrases into sub-clusters and then assigns individual tokens to separate categories. The system relies upon domain-independent (shallow natural language processing and constrained logic programming) techniques to learn product-specific feature ontologies from customer comments. While our preliminary results show some promise, they suggest the need for additional work to evaluate the extensibility and scalability of the existing technique as well as further research on a number of conceptual refinements.

### **7.1 Extensibility and scalability**

The evaluation presented in this paper is currently limited to a single product category: digital cameras. Although the evaluation is intended to illustrate the viability of our approach, it leaves open the question of extensibility; whether the approach is limited to a single product domain or even to products that share similar latent characteristics such as "all portable consumer electronics" or "all image-based systems."

We are currently applying the technique across product categories ranging from the newest consumer electronics to mature product categories such as home appliances. In this way, we can test the sensitivity of our shallow approach. Our goal is general applicability across a wide-range of product categories, capable of distinguishing highly diverse features, without relying upon product-specific domain knowledge. It is worth noting that mature products such as vacuum cleaners or toaster ovens tend to have fewer buying guides limiting the number of external sources available for external validation. Moreover, there are other domains where similar phrase-like text strings prevail as opposed to prose. Progress notes in medical records and government regulatory filings are two such examples that we are also exploring.

Closely related to the question of extensibility is that of scalability. While scalability typically refers to an ability to scale to larger collections of data, we need to assess the stability of our results with respect to smaller data sample sizes. We rely upon a large data set to yield the phrases from which we identify a maximal clique. The large data set is also a boon because we can liberally discard phrases to minimize the effects of naïve parsing (see above). Not all product categories are associated with large sets of reviews, however. To measure the sensitivity to sample size, we aim to cross-validate our results using smaller sets of review samples. We can plot the trade-off between sample size and ontology performance measures to identify diminishing returns and attempt to estimate a minimal number of required reviews. Care needs to be taken in ensuring sufficient heterogeneity in the sample selection with respect to different product features and the corresponding feature attributes.

## **7.2 Conceptual considerations**

To generate initial clusters, the system requires tokenized phrases. Even though Epinions customer reviews provide lists of phrases, variations in human input are a source of noise. Moreover, we assume that a phrase represents a single concept and that each token is a distinct meronym or holonym. This creates difficulties when managing feature lists (e.g. ‘digital and optical zoom’) as well as token sequences (e.g. ‘compact flash’ vs. ‘compactflash’).

One possible solution is to introduce more sophisticated NLP parsing before clustering. For example, rather than treating tokens as atoms, we could apply chunking techniques. A second thought is to substitute a different clustering technique. The phrase  $\times$  token matrix might be characterized by relative sparsity and comparatively high dimensionality. Because this can present problems for center-based clustering like cosine based k-means. We are also experimenting with a shared nearest neighbor density-based similarity measure [37]. We might

even consider altering the initial clustering step from an unsupervised to a supervised approach. It may be the case that customers are sufficiently distributed to successfully train a classifier on a small sample set.

Challenges such as atomic token chunks or parsing errors also appear in the CLP clustering and assignment. Even accurate parsing does not preclude issues with clique generation and token assignment. First, we make the very strong assumption that a maximal clique over a token graph equals a maximum clique. There is no guarantee that our heuristic process does not miss a larger maximal clique that more completely captures the corresponding concept attributes. Moreover, we make the even stronger assumption that a maximum clique captures the set of all meronyms and holonyms of a concept. In practice, this assumption may not prove reasonable. We might weaken the assumption by searching for dense subgraphs instead.

Second are a number of challenges for CLP-based token assignment. Homonyms can link concepts inappropriately. Synonyms and misspellings can segregate concepts incorrectly. Poor parsing of multi-token chunks create inappropriate constraints (e.g. separating ‘mega’ and ‘pixel’ versus ‘megapixel’). These multi-token phrases manifest as multi-valued attributes given our relational interpretation. Likewise, insufficient constraints are akin to null values within a tuple. First, we are currently experimenting with a more traditional mixed integer programming formulation of the assignment problem. The introduction of a penalty function may address the problem of conflicting constraints as well as address unconstrained assignments.

We are also attempting to identify additional sources of constraints. For example, incorporating external data via a foreign key relationship that relates a review to the associated manufacturer’s product description.

### 7.3 Future work

In addition to further experiments to assess extensibility and scalability as well as conceptual refinements, there are a number of ways in which we might enrich the relationships that we are learning.

For example, we currently learn hyponym relationships between classes of features. However, with respect to a single instance, some subclasses are mutually exclusive while others are not. For example, the feature *camera type* might have the following subclasses: *slr*, *standard*, and *compact*. Digital cameras also have the feature *lens* of which *fixed focus* and *zoom* are two such subclasses. A single instance of a digital camera would only be categorized in a single *type* subclass but could support multiple interface types. From a marketing and recommendation perspective, it would be particularly useful to extend our induced ontology to distinguish between mutually exclusive subclasses and those that are not.

The two features of *camera type* and *lens* also exhibit a second dimension of the hyponym relationship. Some subclasses are actually ordinal in nature. *slr*, *standard*, and *compact*, are decreasing in size. Learning ordinal subcategories is particularly useful because of the closely related task of aligning orderings. From a recommendation standpoint, aligning orderings is critical because different customers may address a concept using parallel ordinal categories. For example, one customer might seek a 3.0 *megapixel* camera while another seeks a *resolution* appropriate for 8x10 enlargements. Is a *compact* camera *small*, *medium*, or *large*? Because of the relational assumption underlying our CLP approach, we can apply the concept clustering of [11] or [12] to group tokens from parallel categories.

From aligning parallel orderings we might extend to aligning or merging entire ontologies. There are many different sources of online customer reviews, and one of our end objectives is to

support use-centric mining across reviews gathered from heterogeneous sources. Doing so, however may also require learning ontologies unique to distinct sites and then aligning those ontologies [28]. Ontology alignment can help ensure that the review vectors derived from distinct sources are remain comparable.

While there are many buying guides that provide recommendations for specific products or services, most guides tend to rely upon domain-specific experts. Unfortunately, reliance upon experts is not scalable. Automated support for managing customer and product data is necessitated by the heterogeneity among both producers and users as well as the increasing complexity of products. A critical step in providing automated support lies in simply understanding the language used to describe a particular product category. In this paper, we present an unsupervised, domain independent approach to learning the ontology for specific product categories based upon consumer feedback in the form of online customer reviews. Reviews are first pre-processed using shallow NLP techniques. The resulting phrases are tokenized and then clustered into hyponyms by adapting traditional document clustering algorithms. Further decomposition into meronyms and holonyms is enabled by a novel bounds consistency approach to constraint logic programming that treats the clustering as an assignment problem. We applied the method to a set of several thousand online reviews and evaluated the results against a collection of online buying guides. Interestingly, though the automatically induced features do not perfectly align with the published ontologies, this is not necessarily an indication of poor performance. Because our features are drawn directly from customer comments, the differences may reveal a significant opportunity for better managing the consumer, producer relationship.

## **REFERENCES**

- [1] "Epinions.com Customer Reviews," accessed: 5 July 2004, [http:// www.epinions.com/Digital\\_Cameras](http://www.epinions.com/Digital_Cameras).
- [2] P. Black, "Dictionary of Algorithms and Data Structures," 3 Jan 2005, accessed: February, 2005, [www.nist.gov/dads](http://www.nist.gov/dads).
- [3] C. Borgelt and R. Kruse, "Induction of Association Rules: Apriori Implementation," 15th Conf on Computational Statistics (Compstat), Berlin, Germany, 2002.
- [4] D. Brelaz, "New Methods to Color the Vertices of a Graph," *Communications of the ACM*, vol. 22, pp. 251-256, 1979.
- [5] K.-W. Cheung, J. T. Kwok, M. H. Law, and K.-C. Tsui, "Mining customer product ratings for personalized marketing," *Decision Support Systems*, vol. 35, pp. 231-243, 2003.
- [6] Consumer\_Electronics\_HQ, "Buying Your First Digital Camera: The Basics," accessed: 19 November, 2004, <http://www.digitalcamera-hq.com/hqguides/firsttime-buyer.html>.
- [7] Consumer\_Union, "Digital Cameras," Consumer Reports Buying Guide 2005, pp.33-36; 237-240.
- [8] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, Y.-K. Ng, D. Quass, and R. D. Smith, "Conceptual-model-based data extraction," *Data & Knowledge Engineering*, vol. 33, pp. 227-251, 1999.
- [9] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in KnowItAll (Preliminary Results)," WWW 2004, New York, New York, 2004.
- [10] M. Finkelstein-Landau and E. Morin, "Extracting Semantic Relationships between Terms: Supervised vs. Unsupervised Methods," International Workshop on Ontological Engineering on the Global Information Infrastructure, Dagstuhl Castle, Germany, 1999.
- [11] V. Ganti, J. Gehrke, and R. Ramakrishnan, "CACTUS - Clustering categorical data using summaries," ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, 1999.
- [12] D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering categorical data: an approach based on dynamical systems," 24th International Conference on Very Large Databases (VLDB), 1998.
- [13] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing, Stanford KSL-93-04," International Workshop on Formal Ontology, 1993.
- [14] M. Halkidi, Y. Batistakis, and M. Vazirigianis, "Clustering Validity Checking Methods: Part II," *SIGMOD Record*, vol. 31(3): p. 19-27, 2002.
- [15] J. Han and Y. Fu, "Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases," AAAI 94 Workshop on Knowledge Discovery in Databases (KDD94), 1994.
- [16] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers, 2001.
- [17] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," Fourteenth International Conference on Computation Linguistics (COLING), Nantes, France, 1992.
- [18] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," KDD04, Seattle, WA, 2004.
- [19] Jupitermedia, "UN: E-Commerce Could Bridge the Divide", ECommerce-Guide.com, 27 November 2001, URL: [www.ecommerce-guide.com/news/trends/article.php/929691](http://www.ecommerce-guide.com/news/trends/article.php/929691)

- [20] A. Kilgarriff, "Comparing Corpora," *International Journal of Corpus Linguistics*, Vol 6(1), 2001. 97-133.
- [22] Y. Lu, Concept hierarchy in data mining: specification, generation and implementation, Thesis for the degree of Master of Science, Submitted to School of Computing Science, Simon Fraser University, Vancouver, BC, p. 106.
- [23] A. Maedche and S. Staab, "Semi-automatic Engineering of Ontologies from Text," Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE'2000), Chicago, 2000.
- [24] A. Maedche and S. Staab, "Ontology learning for the Semantic Web," *IEEE Intelligent Systems*, pp. 72-79, 2001.
- [25] A. Maedche, G. Neumann, and S. Staab, "Bootstrapping an Ontology-based Information Extraction System," in *Intelligent Exploration of the Web*, P. Szczepaniak, J. Segovia, J. Kacprzyk, and L. Zadeh, Eds. Heidelberg: Springer / Physica Verlag, 2002.
- [27] M. Missikoff and R. Navigli, "Integrated approach to Web ontology learning and engineering," *IEEE Computer*, pp. 54-57, 2002.
- [28] G. Modica, A. Gal, and H. Jamil, "The Use of Machine-Generated Ontologies in Dynamic Information Seeking," CoopIS 2001, Trento, Italy, 2001.
- [29] T. Nasukawa and J. Yi, "Sentiment Analysis: Capturing Favorability Using Natural Language Processing," K-CAP'03, Sanibel Island, Florida, 2003.
- [30] B. Omelayenko, "Learning of ontologies for the Web: the analysis of existent approaches," ICDT 01 International Workshop on Web Dynamics, London, UK, 2001.
- [31] I. Peacock, "E-Commerce in Europe", Exploit Interactive, issue 3, October 1999  
URL: <http://www.exploit-lib.org/issue3/ecommerce/>
- [32] A.-M. Popescu, A. Yates, and O. Etzioni, "Class extraction from the World Wide Web," AAAI 2004 Workshop on Adaptive Text Extraction and Mining (ATEM), 2004.
- [33] G. Salton and M. McGill, *Introduction to modern information retrieval*. New York: McGraw-Hill, 1983.
- [35] S. Skiena, *The Algorithm Design Manual*. New York: TELOS, Springer-Verlag, 1998.
- [36] H. Suryanto and P. Compton, "Learning classification taxonomies from a classification knowledge based system," ECAI 2000 Workshop on Ontology Learning, Berlin, 2000.
- [37] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Education Inc., Boston, 2006.
- [38] J. Ullman, *Principles of Database and Knowledge-Base Systems, Vol 1*. Computer Science Press, Rockville, MD, 1988.
- [39] Y. Zhao and G. Karypis, "Criterion Functions for Document Clustering: Experiments and Analysis," technical report, Univ of Minnesota, Dept. of Computer Science, Minneapolis, 2002.