Proceedings of 8th ACM Conference on Electronic Commerce (ACMEC 2007), San Diego, CA, June 2007, pp. 182-191.

Red Opal: Product-Feature Scoring from Reviews

Christopher Scaffidi[1], Kevin Bierhoff[1], Eric Chang[2], Mikhael Felker[2], Herman Ng[1], Chun Jin[1]

[1] School of Computer Science
[2] Heinz School of Public Policy and Management
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

{ cscaffid, kbierhof, wenchiec, mfelker, ngherman, cjin } @ andrew.cmu.edu

ABSTRACT

Online shoppers are generally highly task-driven: they have a certain goal in mind, and they are looking for a product with features that are consistent with that goal. Unfortunately, finding a product with specific features is extremely time-consuming using the search functionality provided by existing web sites.

In this paper, we present a new search system called Red Opal that enables users to locate products rapidly based on features. Our fully automatic system examines prior customer reviews, identifies product features, and scores each product on each feature. Red Opal uses these scores to determine which products to show when a user specifies a desired product feature. We evaluate our system on four dimensions: precision of feature extraction, efficiency of feature extraction, precision of product scores, and estimated time savings to customers. On each dimension, Red Opal performs better than a comparison system.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Search–*Relevance feedback, retrieval models, search process*

General Terms

Algorithms, design

Keywords

E-commerce, feature extraction, product reviews, search

1 INTRODUCTION

A hypothetical user named Art wants to give his granddaughter a book of ghost stories. He searches Amazon for books containing "ghost story," sorts the 16620 matches by Average Customer Rating, and browses the results. The first is an erotic book about vampires—not the gift that Art was looking for! The next concerns a murder by the Ku Klux Klan. When Art finally finds books about ghosts, he delves into reams of users' reviews to determine which books are worth reading. After 15 frustrating minutes, he gives up and calls his daughter, asking her to suggest a specific book that he can give to her.

Art's experience is all too common. Though most online shoppers are goal-oriented and want to finish shopping as fast as possible [23], picking a product in an e-commerce site requires 10 to 15 minutes [2][9]. A major reason is that sites' search results often overwhelm users [2][25].

This is unfortunate for both the seller (who has created a barrier to sales) and the user (who would like to invest less time in the whole process). In response, researchers have developed various recommendation systems that try to guide customers quickly to products that are likely to satisfy the users' needs. Some of the most promising systems, such as [4] and [7], take advantage of other customers' product reviews, which are theoretically less biased than the manufacturer's product description.

Our system, Red Opal, resembles these systems and makes two contributions. First, it improves on existing systems by using baseline statistics of words in English and probability-based heuristics to more accurately identify features of product categories. In addition, it scores each product on *each* feature of products in the category. As a result, users can select a category (e.g.: fiction books) and quickly retrieve products that are highly rated on a feature of that category (e.g.: ghost story). Our system is fully automatic and not restricted to specific product categories.

We evaluate Red Opal on four dimensions:

- *Feature extraction precision:* As rated by prospective end users, our extraction algorithm has a precision of 88% (compared to 64% for a comparison system [7]).
- *Feature extraction efficiency*: Feature extraction execution time is proportional to the number of reviews.
- *Product scoring precision*: Our scoring algorithm has a precision of 80% for high-scoring products.
- *Time savings to end users*: Our system cuts the time to find items from 10-15 minutes to 3 minutes.

After discussing related work in Section 2, we describe Red Opal in Section 3, then evaluate it in Section 4 and examine its limitations in Section 5.

2 RELATED WORK

Red Opal helps users find products based on product features. Achieving this involves two tasks, each with its own collection of related work.

2.1 Extracting Feature Terms

Unlike classic document retrieval, we do not index reviews according to terms that place documents "far apart" in an abstract space [19]. Instead, we seek a short list of terms corresponding to product features that customers care about. We use this short list to populate our user interface with shortcuts to products with those features.

In their intent, the algorithms of Hu et al [7] [11] and Popescu et al [16] are most comparable to Red Opal. Like our system, theirs apply part-of-speech (POS) tagging to identify nouns and noun phrases, based on the observation that features are generally nouns [15]. Hu et al use an association-mining tool called CBA to identify frequent noun phrases ("item sets"), each of which is a possible feature. Popescu et al extract features by computing pairwise mutual information between noun phrases and a set of meronymy discriminators associated with the product category. Both approaches mine the co-occurrences of the features and products within the review databases.

Two similar algorithms that use word co-occurrences are [13] and [14], but these do not target product features per se. For example, [14] identifies descriptive words, such as "slow," "doesn't work," and "no problem(s)."

A major difference between the four algorithms above and our own is that whereas those algorithms use word co-occurrence, we apply a language model approach with the assumption that product features are mentioned more often in a product review than they are mentioned in generic English. Our evaluation in Section 4 demonstrates that our approach leads to identification of better features.

Like ours, several other algorithms use words' baseline occurrence rates in generic English to identify terms, though none of these algorithms seem to have been applied to product reviews. For example, these include algorithms for extracting technical terms from discussions of cold war politics [3] and physics [18]. A number of similar algorithms are reviewed in [8].

2.2 Scoring Products

Some product scoring algorithms use lexica such as WordNet to identify opinion words in the reviews in order to assign scores [7] [15] [21]. Other algorithms build a context-aware model of opinion words [4] [14] [16]; such models cope with the limited set of words in lexica, but learned models may not generalize across product categories. In general, scoring products based on opinion words does not always work well due to the intrinsic complexity of languages, such as negation, synonym, polyseme, and long-distance correlation. For these and other reasons, opinion extraction is still a hard problem [1].

In most online reviews, users assign a numeric rating to the product, and then discuss specific features in the review. However, most of the algorithms cited above do not take advantage of this numeric rating. In contrast, Red Opal uses this rating rather than opinion words to compute product scores for features mentioned in the reviews. As a result, our system has the advantage of not requiring training phases for building an opinion word model. However, our decision not to include opinion words involves certain trade-offs, which we discuss in Section 5.

Our product-scoring algorithm differs from these opinion-words algorithms in another major respect. We recognize that different customers look for different features in products, so our algorithm generates a distinct score for each product on each feature. For example, if we find 10 features for a certain product category, then we generate 10 corresponding scores for each product in the category. In contrast, most of the opinion-words algorithms mentioned above (with the exception of [7] and [16]) only generate a single overall score for each product, which makes it hard for users to shop for products based on specific features.

This difference is a significant advantage for our system, since different customers value products for different reasons. Whereas some customers might be entirely happy with a product, leading them to brag about the product in online reviews, other customers may value other features and decide to moan about the product in reviews. This may lead to a bimodal distribution of ratings in reviews, as reported by [6]. Our search system allows users to locate products based on features, and we display a meta-score so that users are aware of variance in the reviewers' opinions on each product's support for the feature.

Our work on product-feature scoring is related to text information retrieval [19], since Red Opal retrieves reviews that mention a feature of interest and uses those reviews to compute scores. We weight reviews based on how often they mention feature terms. Our weighting resembles the well-known TFIDF method for ranking search results [24], though as we discuss in Section 3.2, we cap the maximum weight of each review to prevent any one review from dominating the overall product score.

Finally, once any system has calculated products' scores for each feature, how should it present scores to users? Hu et al use graphical bar graphs [11], while Dave's system presents review snippets [4]. Our goal is to help users find good products as quickly as possible, so we simply present a list of hyperlinks to the products that scored best on each feature. This minimalist approach also will simplify the challenge of integrating our user interface with larger sites, such as Amazon.

3 RED OPAL

Red Opal's input is a database of reviews. Each review has text discussing a particular product's qualities, as well as a rating for that product. Within the database, products are uniquely identified by category name and a number.

Our system has three main components: a Feature Extractor, a Product Scorer, and a User Interface. For each product category c, the Feature Extractor examines the review texts for that category's product set P_c, and it identifies terms that probably describe features of products in P_c. It writes this set of features F_c back into the database.

Then, for each category c, the Product Scorer retrieves P_c and F_c. For each product $p \in P_c$ and each feature $f \in F_c$, the system looks for reviews of p that mention f. It combines those reviews' product ratings into a weighted product score $s(p,f)$. It also computes a confidence metascore and writes this along with $s(p,f)$ to the database.

Red Opal completes the processing discussed above prior to a customer's arrival. The customer view our User Interface web site and selects a product category c. The system responds by displaying the associated features F_c. After the user selects a feature f, the system retrieves P_c and sorts them in order of decreasing score $s(p,f)$.

We now discuss each of these three components in detail.

3.1 Feature Extractor

3.1.1. Pre-processing and configuring

Our prototype currently is not integrated with any e-commerce site. Instead, before executing our Feature Extractor, we pre-populate our database by downloading reviews from an XML feed that Amazon provides. In Section 5, we discuss the possibility of integrating our system with an e-commerce site.

We run each review through a part-of-speech (POS) tagger [12]. This appends the respective POS to each word, [1] and then appends the dictionary form of the word, known as its "lemma." For example, "I played this game for hours" becomes "I/PRP/I played/VBD/play this/DT/this game/NN/game for/IN/for hours/NNS/hour /././" The POS-tagged text is cached in our database. (When Red Opal is eventually integrated with an e-commerce site, we could tag each review when it is initially created. Other researchers' algorithms also use POS-tagging [7].)

Finally, before feature extraction, we configure Red Opal with statistics on how often each lemma appears in generic text. These statistics must come from text written in the same language and roughly the same style as the reviews to be processed, so we configure Red Opal with lemma-frequency data derived from a 100 million-word corpus of spoken and written conversational English [10].

Prior research indicates that technical terms and product features are generally nouns [15]. Therefore, we use the configuration to determine a list of lemmas that should be ignored because they are more likely to occur as non-nouns than nouns (such as "roll"); such words are unlikely to be good feature terms.

In short, for each noun x, the configuration specifies the probability p_x that a randomly selected noun in English text equals x. (We discuss non-dictionary words below.)

After pre-processing and configuration, the main portion of our algorithm can begin. The Feature Extractor currently looks for two kinds of features (single nouns and composite nouns), each of which we now discuss.

3.1.2. Examining single noun lemmas

What the reviews and the baseline corpus have in common is their language and conversational style. What sets them apart is the fact that the reviews focus on a specific topic, which is the product category c under discussion. Consequently, some words occur far more frequently in the review text than would be expected in a random section of English text of equal length. That observation forms the basis for Red Opal's feature-selection criterion.

The Feature Extractor begins by counting the total number of times n_x that each noun x appears in reviews of products in category c. The system also computes the total number of noun occurrences in the category's reviews:

$$N = \sum_x n_x \qquad \textit{Eq 1}$$

Next, the algorithm calculates the probability that lemma x would occur n_x times in random English text containing a series of N noun occurrences $<w_1, w_2, \ldots, w_i, \ldots, w_N>$. Two assumptions facilitate this calculation:

- We assume that the occurrence of lemma x in position i is independent of whether lemma x occurs in any other position j, so $P(w_i = x \mid w_j = x) = P(w_i = x)$.
- We assume that the occurrence of lemma x in position i is independent of i, so $P(w_i = x) = p_x$.

Strictly speaking, these assumptions are false. For example, English speakers almost never use the same noun twice in a row, so the first assumption is false. In addition, some nouns like "hotness" almost never occur as a sentence's first noun, so the second assumption is false.

Despite these qualms, researchers often accept these assumptions because the assumptions do not necessarily harm the quality of results [24], and the assumptions help simplify the problem so it is computationally tractable [8].

[1] MontyLingua uses the same tags as the Penn TreeBank. Refer to http://www.cis.upenn.edu/~treebank/ for a POS tag manual.

Under these assumptions, the binomial distribution represents the probability that lemma x occurs n_x times in a set of N nouns. For large N and small p_x, the Poisson distribution approximates a binomial distribution [22]:

$$P(n_x) \approx \frac{(p_x N)^{n_x} \cdot e^{-(p_x N)}}{n_x!} \qquad \textit{Eq 2}$$

Taking the logarithm (to avoid numeric underflow), applying Stirling's approximation, and using the fact that $p_x << 1 << N$ yields Equation 3.

$$\ln(P(n_x)) \approx (n_x - p_x N) - n_x \ln(\frac{n_x}{p_x N}) - \frac{\ln(n_x)}{2} \qquad \textit{Eq 3}$$

If $n_x > p_x N$ and $\ln(P(n_x))$ is small, then it is unlikely that x occurred so often by chance. In actual use, the probability in Equation 3 is very small for actual features. For example, in one sample of 1690 digital camera reviews, which included N = 58543 noun occurrences, the lemma x = "lens" occurs n_x = 1174 times, even though "lens" constitutes only p_x = 3.1E-5 of all noun occurrences in English. As a result, $\ln(P(n_x))$ = -6429. This is an astonishingly small probability, leading Red Opal to identify "lens" as a feature term for products in this category.

When p_x cannot be determined because lemma x does not appear in the baseline lemma-frequency table, Red Opal defaults p_x to the average of all p_x values among English nouns. This sets a relatively high bar for how many times a word must be misspelled before it appears to be a product feature. Yet this is not too high of a bar to overcome in the case of non-dictionary terms that truly do deal with the product category. For example, "megapixel" occurs 195 times in a sample of 1690 digital camera reviews, even though it does not appear in the baseline lemma-frequency table at all. Using the default p_x of 1.05E-4 yields $\ln(P(n_x))$ = -488. From this, it might be suspected that "megapixel" is a feature of products in this category.

3.1.3. Examining lemma bigrams

Researchers have noted that many technical terms are compound nouns [15]. An adaptation of the above derivation estimates the probability that two nouns would occur successively a certain number of times as a bigram.

If x and y are nouns, let ρ_{xy} denote the rate of occurrence of bigram <x, y> among noun bigrams in English. (For example, if ρ_{xy} = 1E-3, then 1 out of every 1000 noun bigrams would be <x, y>.) Viewing a bigram as a 2-noun segment of English text, and using the second assumption in Section 3.1.2, the probability is p_x that x is the first noun of a randomly selected bigram, and the probability is p_y that y is the second noun of the bigram. Then, using the first independence assumption above, the probability of the bigram as a whole is $\rho_{xy} = p_x p_y$.

If a bigram <x, y> occurs η_{xy} times in the corpus, then the total number of bigram occurrences H is:

$$H = \sum_{x,y} \eta_{xy} \qquad \textit{Eq 4}$$

Carrying over the derivation in Section 3.1.2 leads to the logarithm of the probability $P(\eta_{xy})$ of observing η_{xy} occurrences of <x, y> in H randomly selected English bigrams.

$$\ln(P(\eta_{xy})) \approx (\eta_{xy} - \rho_{xy} H) - \eta_{xy} \ln(\frac{\eta_{xy}}{\rho_{xy} H}) - \frac{\ln(\eta_{xy})}{2} \qquad \textit{Eq 5}$$

If $\ln(P(\eta_{xy})) << 0$, and if $\eta_{xy} > \rho_{xy} H$, then it is unlikely that <x, y> could have occurred so often by chance. This implies that <x, y> might be an interesting feature of products in the category.

3.1.4. Selecting features for the product category

After calculating probabilities for each lemma and bigram in reviews of products in P_c, Red Opal identifies a final feature set F_c. To do this, it discards features that simply repeat the category name c (e.g.: "digital camera" is not a good feature of digital cameras). Then it combines the single-noun features and bigram features into a short list, sorts them by probability, and returns the best features.

3.2 Using Reviews to Score Products

Our goal is to enable users to search for products that are particularly good in a particular feature. The fundamental idea of Red Opal is to use customers' reviews to infer how products perform in the features F_c associated with that product's category c. A review consists of freeform text and a user's product rating. In the case of Amazon, users can rate products with 1 to 5 stars, where more stars indicate higher ratings. We not only compute a product score for each of the product's features, but also a meta-score that expresses confidence about the score.

3.2.1. Selecting reviews and calculating scores

We make the simplifying assumption that a user's product rating correlates with the quality of the features that the user discusses in the review text. Under this assumption, the mean of all ratings where the review mentions a given feature is a reasonable first estimate of the product's quality in that feature. We improve on this first guess by introducing a weighting scheme to compute a better product score for each applicable feature.

Intuitively, it matters if a review mentions a feature only once or multiple times. Multiple occurrences of a feature in a review indicate that the review more thoroughly discusses this feature, and therefore the review's rating should have a greater influence on the product score.

Thus, we assign weights to each review and use these weights in calculating the mean. A simplistic approach would let the weight of a review be proportional to the number of occurrences. However, this approach might let a particular review be arbitrarily influential to the score. (This would also be a problem if we used TFIDF [24].)

Consequently, we bound the maximum weight *w(r,f)* of a review on a product-feature score:

$$w(r,f) = \sum_{i=0}^{o(r,f)-1} 2^{-i} = 2 - 2^{1-o(r,f)} \qquad \text{Eq 6}$$

Here, *o(r,f)* is the number of occurrences of feature *f* in review *r*. The key property of this weighting is that every occurrence of a feature in a review influences the product score twice as much as any successive occurrence of the feature in that review. The net effect is to restrict the total weight *w(r,f)* of a review between 1 and 2.

With this weighting scheme, for each product *p* and each feature *f*, we compute the product's score *s(p,f)* as:

$$s(p,f) = \frac{\sum_r w(r,f) \cdot rating(r)}{\sum_r w(r,f)} \qquad \text{Eq 7}$$

Here, *r* ranges over all reviews that discuss feature *f* of product *p*. (This linear scoring formula is admittedly ad hoc, and future versions of Red Opal may incorporate a menu of more sophisticated scoring functions.)

3.2.2. Calculating a score confidence meta-score

A score *s(p,f)* can result from various rating distributions. For instance, a score of 3 could result from ratings 1 and 5 or from ratings 3 and 3. In both cases, it is reasonable to assume a score of 3, but our *confidence* in this score should be smaller in the first case. Also, it intuitively matters whether the score is derived from 1 or 10 reviews. Fewer reviews should imply lower confidence.

Thus, we provide a meta-score to summarize how well the set of reviews agree on a product-feature score *s(p,f)*. We convey this information to the user in form of a confidence bar next to the actual score *s(p,f)*.

The *standard error* $e = \sigma / \sqrt{n}$ is commonly used to establish a confidence interval around a sample-based point estimate of a population mean [22]. This statistic uses the sample size *n* and the standard deviation σ (typically estimated from the sample standard deviation).

The standard error ranges between 0 and 2 for sets of Amazon ratings. We can thus normalize the standard error of a sample of ratings to a confidence value between 0 and 1, where 1 expresses the highest confidence:

$$confidence(p,f) = 1 - \frac{\sigma_w(p,f)}{2\sqrt{\sum_r w(r,f)}} \qquad \text{Eq. 8}$$

Here, $\sigma_w(p, f)$ is the weighted standard deviation of ratings for reviews of product *p* that mention feature *f*. Instead of *n*, we use the sum of weights *w(r,f)*; this sum is ≥ n.

Note that confidence is always interesting to compute, even with more elaborate scoring schemes, since individual reviewers can have different opinions about a feature (just as different reviewers can have different opinions about the product as a whole, as reported in [6]). In fact, in our evaluation, we found many products where different reviews expressed completely different experiences with a feature. This is sometimes due to problems in the production or improvements made by the manufacturer after the product was available for some time. Therefore, we are optimistic that showing a confidence meta-score to users will help them evaluate the reliability of a supplier, resulting in better overall buying decisions.

3.3 User Interface

The preceding computations yield a list of product features F_c for each product category *c*, as well as a score *s(p,f)* for each product $p \in P_c$ on each feature $f \in F_c$. Red Opal caches these results in the database and uses them to provide a highly intuitive user interface.

A demonstration of our system (Figure 1) is viewable at http://redopal.ntelligentsolutions.net This standalone PHP web application runs from a cache of Amazon data: we extracted 10 features for each of 8 categories, which together contain a total of over 700 products and 5000 reviews. (In Section 4.2, we evaluate system scalability.)

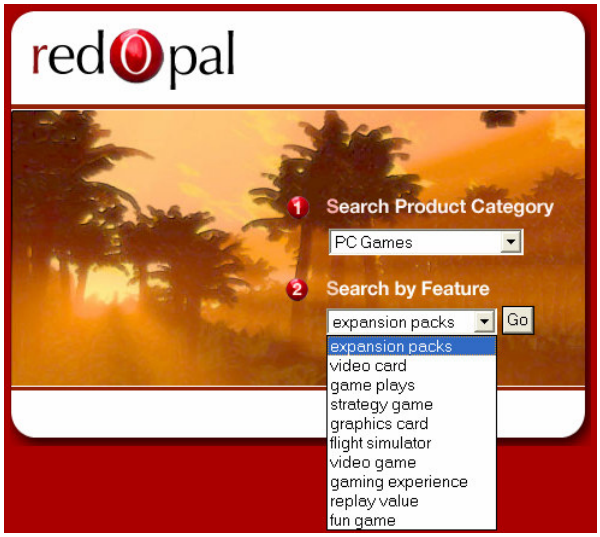

Figure 1. Red Opal search screen

In Figure 1, the user selects a product category *c* ("PC Games"), and Red Opal immediately displays the corresponding list of features F_c. The user selects a feature ("expansion packs") and clicks the "Go" button. (Art, the example customer in Section 1, would probably choose the "Fiction Books" category and then "ghost story.")

As shown in Figure 2, Red Opal displays a paged result set of products, sorted according to scores on that feature. The score confidence appears alongside each score.

Figure 2. Red Opal search results page

The user can click on any of the product names in Figure 2 to access a detailed product page on Amazon. In one version of the site, we also experimented with providing links to retrieve the specific reviews that yielded that product's score on the selected feature. The current version does not include this feature, as we have not settled on an appropriate sorting order for product reviews.

In Section 5, we discuss desirable enhancements to this user interface. For example, we would like to include a tool that would enable users to search for multiple features simultaneously, and we would like to explore the tradeoffs associated with letting users type in a feature name, rather than select it from a pre-determined list. We anticipate that iterative prototyping and user evaluation will reveal costs and benefits of adding new functionality.

We could easily integrate the user interface into Amazon itself, or into another site. This task would be significantly eased by the fact that our site is implemented in XHTML, and most of the layout, colors, and so forth are factored out modularly using Cascading Style Sheets (CSS). This modularity will also facilitate iterative prototyping and evaluation.

4 EVALUATION

Red Opal outperformed comparison systems on precision of feature extraction, efficiency of feature extraction, precision of product scores, and time savings to users.

4.1 Precision of Feature Extraction

4.1.1. Evaluation overview

At present, one of the most widely-cited feature extraction algorithms is that of Hu and Liu [7], so we use a slightly simplified version of their algorithm as a baseline for evaluating our own Feature Extractor. Specifically, we compare our algorithms on precision (the fraction of algorithm outputs that are correct), as it is a widely accepted measure of a machine learning algorithm's quality [17].

To evaluate precision, we use each algorithm to generate a list of outputs (product features) and then mark each as correct or incorrect. In some evaluations (e.g.: [7] and [16]), people who helped to build the system perform the marking procedure. In contrast, in order to minimize bias, we recruited students at Carnegie Mellon University to grade the features. These students are representative of a highly desirable e-commerce demographic (in their twenties, well-educated, with substantial disposable income).

Machine learning algorithms can be tuned to increase precision at the expense of reducing recall (the fraction of all correct answers that the algorithms actually output). Thus, precision is usually reported as a function of recall.

However, our goal is not to return *every* feature of each product category, since this would make the drop-down lists in Section 3.3 virtually unusable. The resulting system would be no better than existing systems that overwhelm the user [2]. Thus, for this search system, a better way to report precision is as a function of the number of features displayed on the user interface.[2]

As a secondary goal, we hoped to assess if users preferred bigrams over single-noun lemmas. To ensure a supply of each type of features, we tweaked Red Opal's algorithm to return equal numbers of bigrams and single-noun lemmas. While this modification provided useful information about user preferences, it also forced Red Opal to return

[2] For the reasons outlined above, we did not implement the "infrequent feature identification" stage of the Hu and Liu algorithm, since this stage tends to reduce their algorithm's precision in order to raise its recall, so including that stage would have unfairly reduced their algorithm's score. We also did not implement compactness pruning (in order to improve their algorithm's computational efficiency) until after we had completed our evaluation, but we found in retrospect that omitting compactness pruning did not substantively affect results, since including it would only have affected a single feature generated by their algorithm. For a detailed description of our implementation of their algorithm, refer to [20].

some features with inferior log-probability values. That is, Red Opal would have returned better results if we had not constrained it in this way. Consequently, the precision reported here is a *lower bound* on the quality of features returned by our system.

4.1.2. Evaluation procedure

In our evaluation, we cached over 5000 reviews of 7 product categories from Amazon, shown in Table 1. We selected these categories because students are likely to be familiar with such products. We used each algorithm to generate 6 features for each category. Combining these two sets of 42 features yielded 8 to 12 distinct features per category (since some were generated by both algorithms), for a total of 70 features overall.

Table 1: The categories covered by our evaluation

Category	# of Products	# of Reviews	# of Features
Digital Cameras	110	1690	10
DVD Players	100	808	11
E-Commerce Books	110	321	10
Grills	15	208	12
PC Video Games	110	1277	8
PS2 Video Games	110	594	9
Watches	110	108	10

For each product category, we alphabetized the features (so there was no indication that they were generated by different algorithms) and then posted an online survey for students to evaluate the features. The survey instructed respondents to imagine that they were shopping at Amazon for a birthday gift for a friend. The questions asked whether searching for each feature term would be helpful for finding products. For example, in the case of digital cameras, the survey asked whether searching for each term "might help you pick out the right DIGITAL CAMERA for a friend." Respondents could grade each candidate feature using the following scale:

- Definitely Helpful (coded as a grade of 3)
- Probably Helpful (coded as a grade of 2)
- Probably Not Helpful (coded as a grade of 1)
- Definitely Not Helpful (coded as a grade of 0)

We sent email to 29 students, inviting them to take our survey. Although 12 took the survey (a response rate of 41%), 1 respondent entered a grade of "Definitely Helpful" for every single feature, and another respondent's answers correlated very poorly with those of the remaining 10 respondents (specifically, a correlation of 0.18 with the mean of the other 10 respondents' answers). Consequently, we dropped the data from these 2 respondents. The grades from the remaining 10 respondents were quite consistent (standardized Cronbach α =0.89).

We marked each candidate feature as "correct" if at least half of the respondents graded that feature as "Probably Helpful" or "Definitely Helpful." This allowed us to compute the precision of each algorithm. Moreover, because both algorithms generate features in sorted order, we could evaluate "what-if" scenarios to determine how the algorithms' precision would have varied if they had been used to generate 5, 4, 3, 2, or 1 features instead of 6.

Finally, as an additional measure in our evaluation, we computed an average grade for each algorithm. For this average grade, we only included features that were generated by that algorithm but not the other. This enabled us to perform a t-test to evaluate if differences in grades were statistically significant.

4.1.3. Evaluation results

As shown in Figure 3, our Feature Extractor yielded higher precision (85% to 90%) than the comparison algorithm (40% to 75%). The comparison algorithm gave particularly high precision when configured to return larger numbers of results, and the precision of our Feature Extractor was fairly constant. (These precisions for the Hu/Liu algorithm are comparable to those reported in [7], in which they ranged from 50% to 80%, depending on product category, level of pruning, and post-processing.)

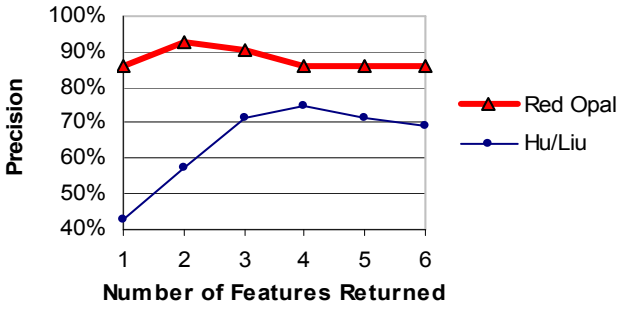

Figure 3. Red Opal feature extraction had higher precision than a comparison system.

Our algorithm also demonstrated a superior average grade. Of the 70 features, 28 were generated only by our algorithm, and 28 were generated only by the comparison algorithm. (The remaining 14 were generated by both.) Our algorithm's average grade was 1.69 (S=1.07), and the comparison algorithm's was 1.40 (S=1.03), a statistically significant difference (t=3.17, df=558, P<0.001).

Finally, to assess the relative utility of generating features comprising compound nouns, we compared the scores of single-noun features to compound nouns. Of the 70 features, 44 contained one noun and 26 were compound. The average of the single-noun grades was 1.43 (S=1.05), while the average of the compound-noun grades was 1.98 (S=1.02). The users' preference for compound nouns was statistically significant (t=6.73, df=698, P<0.001).

4.2 Efficiency of Feature Extraction

We discuss our scalability evaluation in brief and refer the reader to [20] for details. When we ran our Feature Extractor on a commodity desktop machine for each product category listed in Table 1, our algorithm averaged less than 30 ms per review. This does not include downloading reviews and POS-tagging, which required over 250 ms per review, but which could be performed in advance when reviews are initially created.

To assess throughput for higher numbers of reviews, we downloaded 32000 reviews of paperback fiction books from Amazon and ran the algorithm on increasingly larger subsets of reviews. Specifically, we ran the algorithm 4 times for each of the following 7 collection sizes: 1000, 2000, 4000, 8000, 16000, 24000, and 32000 reviews. A linear fit to the 28 points reveals that the time increases at an average of 0.9 ms per review (R^2=0.99, F=2227, df=1, P<0.001), which comes to 15 minutes for 1 million reviews. See [20] for a discussion of algorithmic complexity, which is *O(n)*, where *n* is the number of reviews.

4.3 Precision of Product Scores

An evaluator read reviews for products and assigned a score for each product on each feature. This enabled us to determine how well system-assigned scores agreed with human-assigned scores. Again, to reduce bias, an author who did not help build the system acted as the evaluator.

To reduce the burden of generating thousands of scores, we selected only a subset of products, features, and reviews. We chose two categories, digital cameras and e-commerce books, for feature diversity (physical versus conceptual features, respectively). We chose five digital camera and four e-commerce features (listed in Figure 4) that were highly valued by students and the evaluator.

For each category-feature pair, Red Opal returned a ranked list of products. Some result sets contained hundreds of products while others contained only a few. For each unique score appearing in the list, we selected up to five products with the highest confidence, so the test cases covered the entire score range with highly-skewed ones limited to five products at most. These criteria identified 114 product-feature test cases; some features such as "business model" and "technology" had small score ranges, as Red Opal did not generate low scores for any of these products. For each test case, the evaluator read all reviews and attempted to assign a score in the range of 1 through 5. In 16 cases, the evaluator was unable to assign a score because the reviews were too contradictory or did not provide any clear evidence; in these cases, the evaluator assigned a score of 3 (which we discuss further below).

To report results, we use a generalized precision metric that deals with these *numerical* scores to represent the agreement between system-assigned scores and human-assigned scores. (In traditional information retrieval, the precision is defined from *binary* scores [17]; the system score is 1 if an item is retrieved or 0 otherwise, and the human score is 1 if the item is relevant or 0 otherwise.)

The generalized precision, at rank cut of *n*, is a function of the score differences on the top *n* products from the system-generated ranked list. The score difference on the *i*th product is the normalized difference between the system-assigned score S_i and the human-assigned score E_i. Dividing by the larger of the scores normalizes the result.

$$precision = 1 - \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{|S_i - E_i|}{MAX(S_i, E_i)} \qquad \text{Eq 9}$$

We are primarily interested in the precision of high-scoring products, since when a customer values a certain feature, he or she seeks high-quality products on that feature [25]. For the products with a system-assigned score of 5, the average precision on the nine features is 80%.

Although we are mainly concerned with the precision for the top products, precisions at other score cuts, such as system *score* ≥ *4*, or *score* ≥ *3*, and so forth, provide a view over the entire score range. For each category x feature x cutoff combination, we get a set of test cases (one dot in Figure 4). For most test sets, precision exceeded 80%.

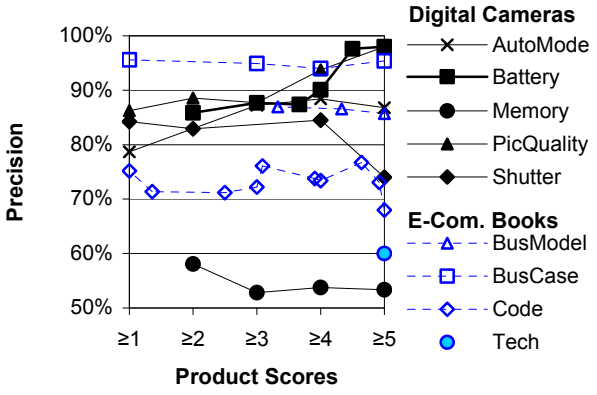

Figure 4 Product scoring precision mostly exceeded 80%

Although Red Opal performed well, there is some room for improvement.

First, reviews sometimes contained negative comments on the feature yet had high overall ratings. In these cases, Red Opal assigned high scores, but the evaluator assigned lower scores. Such cases violate our assumption that each review rating represents the reviewer's opinion about features mentioned in the review. Although our assumption is generally true (as reflected by high precision overall), these cases indicate the need for algorithm refinement.

Second, for 16 of 114 cases (14%), the reviews mentioned the features, but the reviewers' opinions were

ambiguous. In these cases, the evaluator assigned a score of 3, which often was lower than the score assigned by Red Opal. (Excluding ambiguous cases raises this precision to 89% for the highest-scoring products; letting E_i = 1 instead of 3 for ambiguous cases reduces this precision to 66%.) The issue is that Red Opal generally assigned *high confidence* to these scores, contrary to the evaluator's opinion that the reviews were ambiguous. Only 14% of test cases had this problem, but these cases indicate the need for refinement in our meta-scoring algorithm.

4.4 Overall Time Savings for Customers

To generate a preliminary estimate of the time that a customer would save by using Red Opal, our evaluator from Section 4.3 went to Amazon and measured the time needed to pick a good product for each of the 9 features.

The evaluator tried to pick a product as fast as possible by using Amazon's search tool, skimming reviews, reading only relevant sentences, and reading only explicit lists of Pros and Cons when available. The evaluator started by typing a feature into the search tool and selecting the category. The search for digital cameras allows the customer to sort the results by review rating; the tool for e-commerce books does not. In either case, the evaluator then read reviews for the first 10 products returned.

Overall, the search for 9 products took 80 minutes, for an average of 9 minutes per product feature. This is slightly low compared to previous studies, which found that users require 10 to 15 minutes to shop for a product, though this includes checkout, which we did not include [1][9].

Using Red Opal was much faster. The evaluator measured the time to select a category, select a product feature, and then browse through the reviews of the first page of results. On average, this took 3 minutes per product feature (again, not including checkout). This estimate is preliminary, pending confirmation with actual end users.

5 DISCUSSION AND FUTURE WORK

Empirical research suggests that customers have "heterogeneous tastes" [6]: different people value different product attributes or features. Our system automatically identifies the features of products in each category, then helps each user rapidly locate products that probably have good support for that specific user's desired feature.

Prospective users preferred features extracted by our algorithm over those extracted by a comparison system. To explore this, we qualitatively compared terms that were preferred by users to those that were less preferred by users. Users did not prefer vague, general terms like "thing" and "feature," and the comparison system returned more such terms than Red Opal. The baseline statistics of word occurrence rates in general English enabled our algorithm to recognize that the product reviews contained such words in roughly the same proportion as English text, allowing it to disregard such terms.

Compound nouns were preferred over single-noun features, so future versions of Red Opal may bias toward extracting more bigrams. (We did not measure recall in this study, but future evaluations may include measures of recall; if users consider bigrams to be important features, then biasing feature extraction toward bigrams may help to improve recall. Only a couple of trigrams were extracted (by the comparison system), and these were preferred about as well as bigrams, so it is as yet unclear whether we should add trigram features to Red Opal.

We have identified four more ways in which we could extend Red Opal. Each potential extension has tradeoffs.

First, we will consider integrating manufacturer product descriptions for feature extraction. For example, we could rule out words that do not appear in product descriptions (even if they occur often in reviews) to raise precision. Unfortunately, this could significantly reduce recall, since manufacturers write about products from a different viewpoint than customers, so they might not mention important features. For example, in our evaluation, "burger" was a highly-preferred feature term for grills (but did not seem to be mentioned much in product descriptions). In retrospect, this feature made sense, since some grills are excellent for cooking burgers, whereas other grills are indoor grills and entirely unsuitable for cooking burgers.

Second, we will consider finding opinion words in the review text and using them to increase or decrease the rating for features that are mentioned near to the opinion words. This enhancement might improve the precision of our product scoring algorithm, since this change would enable us to take advantage of more information. One complication is that we also would want to account for the presence of negation. For example, a reviewer might write, "This grill is not bad for burgers." Our system's precision might be reduced if we extract "burger" as a feature term, then discount this review's rating because the word "bad" occurs near "burger" (failing to notice the presence of the word "not"). In addition, we would want to find ways of reducing the human effort required to build models of product category-specific opinion words.

Third, we will consider allowing users to type features in our user interface, rather than selecting from pre-built lists. Users might find this to be more flexible. However, users may type words that do not match features identified by our system. To deal with this, we could map unrecognized words to synonyms that are feature terms; we will need to achieve this robustly in a category-agnostic manner. Alternatively, we could perform on-the-fly scoring of products for the specified feature, though this would impose significant online computational load.

Finally, we may enhance Red Opal to let users search for many features simultaneously. We will need to determine how to weight product-feature scores and combine them into overall scores, and how to design a user interface that presents as much scoring information as possible without using excessive screen space and without confusing users.

6 ACKNOWLEDGEMENTS

We thank numerous colleagues for pre-testing our survey and our classmates for taking the survey. We thank Norman Sadeh and George Duncan for suggestions concerning our evaluation. We thank Mary Shaw and Norman Sadeh for allocating discretionary funds for this work. This research was also funded in part by the EUSES Consortium via the National Science Foundation (ITR-0325273), by the National Science Foundation under Grant CCF-0438929, and by the ARDA NIMD program under contract NMA401-02-C-0033. Any opinions, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the sponsors' views.

7 REFERENCES

[1] Allen, J. *Natural Language Understanding*. Benjamin/Cummings Publishing Company, Inc, 1995.

[2] Barnard, L., Wesson, J. Usability Issues for E-Commerce in South Africa: An Empirical Investigation. *Proc. 2003 Annual Research Conf. on South African Inst. of Comp. Scientists and Info. Technologists on Enablement through Tech.*, 2003, 258-267.

[3] Damerau, F. An Experiment in Automatic Indexing. *American Documentation*, *16*, 4 (Oct 1965), 283-289.

[4] Dave, K., Lawrence, S., Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *Proc. 12th Intl. Conf. on World Wide Web*, 2003, 519-528.

[5] Kim, S., Hovy, E. Determining the Sentiment of Opinions. *Proc. 21st Intl. Conf. on Computational Linguistics*, 2004, 1367-1373..

[6] Hu, N., Pavlou, P., Zhang, J. Can Online Reviews Reveal a Product's True Quality?: Empirical Findings and Analytical Modeling of Online Word-of-Mouth Communication. *Proc. 7th ACM Conf. on Electronic Commerce*, 2006, 324-330.

[7] Hu, M., Liu, B. Mining and Summarizing Customer Reviews. *Proc. 10th SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2004, 168-177.

[8] Kageura, K., Umino, B. Methods of Automatic Term Recognition: A Review. *Terminology*, *3*, 2 (1996), 259-289.

[9] Kohavi, R., Parekh, R. Ten Supplementary Analyses to Improve E-Commerce Web Sites. *Proc. 5th WEBKDD Workshop*, 2003.

[10] Leech, G., Rayson, P., Wilson, A. *Word Frequencies in Written and Spoken English: Based on the British National Corpus*, Longman Press, 2001.

[11] Liu, B., Hu, M., Cheng, J. Opinion Observer: Analyzing and Comparing Opinions on the Web. *Proc. 14th Intl. Conf. on World Wide Web*, 2005, 342-351.

[12] Liu, H. *MontyLingua: An End-To-End Natural Language Processor with Common Sense*, Available at: http://web.media.mit.edu/~hugo/montylingua.

[13] Matsuo, Y., Ishizuka, M. Keyword Extraction from a Single Document Using Word Co-Occurrence Statistical Information. *Proc. 16th Intl. Florida AI Research Society*, 2003, 392-396.

[14] Morinaga, S., Yamanishi, K., Tateishi, K., Fukushima, T. Mining Product Reputations on the Web. *Proc. 8th SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, 341-349.

[15] Nakagawa, H., Mori, T. A Simple but Powerful Automatic Term Extraction Method. *Proc. 19th Intl. Conf. on Computational Linguistics*, Morgan Kaufmann Press, 2002, 29-35.

[16] Popescu, A., Etzioni, O. Extracting Product Features and Opinions from Reviews. *Proc. Joint Conf. on Human Lang. Tech. / Conf. on Empirical Methods in Natural Lang. Processing*, 2005, 339-346.

[17] Raghavan, V., Bollmann, P., Jung, G. A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. *ACM Trans. Info. Sys., 7*, 3 (Jul. 1989), 205-229.

[18] Robertson, S., van Rijsbergen, C., Porter, M. Probabilistic Models of Indexing and Searching. *Proc. 3rd Annual ACM Conf. on Research and Development in Information Retrieval*, 1981, 35-56.

[19] Salton, G., Wong, A., Yang, A. A Vector Space Model for Automatic Indexing. *Comm. ACM*, *18*, 11 (1975), 613-620.

[20] Scaffidi, C. *Application of a Probability-Based Algorithm to Extraction of Product Features from Online Reviews*. Tech. Report CMU-ISRI-06-111, School of Computer Science, Carnegie Mellon University, 2006.

[21] Sista, S., Srinivasan, S. Polarized Lexicon for Review Classification. *Proc. Intl. Conf. on Machine Learning, Models, Technologies & Applications*, 2004.

[22] Wackerly, D., Mendenhall, W., Scheaffer, R. *Mathematical Statistics with Applications*, Duxbury Press, 2001.

[23] Wolfinbarger, M., Gilly, M. Consumer Motivations for Online Shopping. *Proc. 6th Americas Conf. on Info. Sys.*, 2000, 1362-1366.

[24] Wu, H., Salton, G. A Comparison of Search Term Weighting: Term Relevance vs. Inverse Document Frequency. *Proc. 4th Annual Intl. ACM Conf. on Information Storage and Retrieval*, 1981, 30-39.

[25] Zahir, S. Designing a Knowledge-Based System for the Web to Support Consumers' Online Decisions. *Proc. Intl. Conf. Comp. & Info. Tech.*, 2002, 381-386.