Automated Product Profiling through NLP

Jiho Han, Dowon (Ronny) Ko

***Abstract*—We design and experiment with an innovative way to automatically generate product profiles from Amazon reviews. Using NLP, we extract *opinions* from each review, clusters them by their *orientation* through an unsupervised learning (k-means). From these clustered opinions, we estimate the product profiling kernel θ and the pricing kernel λ. Finally, we optimize/update the word polarity by minimizing the prediction error (supervised learning). While the trained model perform only slightly better than random guessing, the interim outputs and the estimated parameters seems to provide useful information while showing a possibility for improvement.**

I. Introduction

In this paper, we propose an unconventional approach to summarize Amazon online product reviews. In contrast to previous researches which attempt to simply predict the overall product rating, our model aims to: (1) extract and evaluate the sentence-level *opinion*s, (2) aggregate them into the relevant clusters (i.e., evaluation criteria), and finally, (3) predict the product rating by averaging the scores from the evaluation criteria.

Undoubtedly, this approach has much complicated internal structures, which make parameter estimation process highly difficult. Nonetheless, there are two immediate benefits that can be gained from this approach. First, this model generates a product profile, or a scorecard for each evaluation criteria. Second, we can estimate the degree of an average consumer's consideration on each evaluation criteria for each market product.

This approach, as far as we know, has not been suggested in the literature. Therefore, our effort lacks theoretical support and the fruitfulness of the research is never warranted. Even then, we believe this approach suggests a novel framework for unsupervised – or minimally-supervised – information extraction and is worth the effort to explore.

This paper was originally written in joint effort with Hyungoo Kang. Although he decided to pursue a different project, I recognize and deeply appreciate his contribution for this research.

The rest of the paper is organized as follows. Section 2 goes over the previous literature and provides the formal model specification. Section 3 describes the data which we work with. In Section 4 we discuss the estimation procedures in detail and point out developmental issues. Section 5 shows the model's outcome and the performance result. Lastly, Section 6 concludes and suggests the directions for future research.

II. Literature and Model Overview

A. Literature Review

There are several researches conducted on the topic of document-level classification and review-level rating prediction. Simply because documents have a lot more words than sentences, most of document-level classification models can utilize an N-gram model combined with NER to extract keywords while being agnostic about the syntax structure of each sentence. [1], [3] On the other hand, when evaluating sentences, we do not have such luxury of using a large number of words, and thus do not have to deal with semantic-level of natural language processing!

This is where our model can contribute to the literature. Given a minimal piece of information (i.e., product ratings), we devise an algorithm where sentence-level information extraction is dependent on semantic relations as little as possible. In other words, by jointly estimating the *polarity* and the *orientation* of *opinions* (these terms will be defined soon in the next section) and assuming a certain market structure (what we call the *pricing kernel* and the *orientation kernel*, also explained below), we can hopefully bypass semantic issues to some degree.

In our analysis, the previous work that is most closely related to ours is that of Popescu and Etzioni. [2] In fact, their objective and subtask procedures are almost identical to our model. However, the main difference lies in the use of product ratings and, in general, the assumption of market structures. For example, Popescu and Etzioni extensively use NLP techniques – e.g.,

defining 10 rules to represent semantic relations – to extract relevant product features and the corresponding sentiments while discarding the information from the product ratings.

In the next section, we present our model specification and show how we can leverage the market structure assumption.

B. Model Overview

The fundamental building block for our model is the *opinion*s we extract from each sentence. We define *opinion* as a tuple consisting of *orientation* and *polarity*, i.e.,

$$opinion = (orientation,\, polarity)$$

While *orientation* is an abstract entity, we represent it by a m-dimensional vector x_i for the ith-opinion. On the other hand, *polarity* of an opinion is simply a scalar. Given a set of opinions, we model the predicted rating $\hat{r}$ as follows:

$$\begin{aligned} \hat{r} \quad &= \quad \sum_{\forall i} (\lambda^T \theta x_i) p_i \qquad \text{(II.1)} \\ \text{where} \quad & \lambda \in \mathbb{R}^k \ (pricing\, kernel) \\ & \theta \in \mathbb{R}^{k \times m} \ (profiling\, kernel) \\ & p_i = \text{polarity of opinion } i \\ & k = \text{the number of evaluation criteria} \end{aligned}$$

Profiling kernel θ maps each opinion x_i into k-dimensional space where each dimension represents a criterion with which consumers use to evaluate the product (e.g. price or design). *Pricing kernel* λ is then the weights - or the importance - that each evaluation criterion has on determining the overall product rating. (e.g., How *important* is the battery life when evaluating a cell phone?)

The overall procedures for training the model are summarized in Algorithm 1. Note that the key insight of our model is that we update the polarity and the underlying market structure so that the prediction error is minimized.

Of course, the question is: How exactly can we estimate each entity mentioned above? In Section IV, we will answer that question in detail.

Algorithm 1 Overall training process

Given the orientation extraction rule $\mathcal{R}_O$ and
the polarity extraction rule $\mathcal{R}_P$,
$O \leftarrow extractOpinions(R)$
$P_0 \leftarrow extractPolarityWord(R'; \mathcal{R}_P)$
$O \leftarrow extractOrientations(R,\, P_0; \mathcal{R}_O)$
Until converge {
  $CO \leftarrow clusterOpinions(O)$
  $K_i \leftarrow extractMarketStructure(CO,\, P_i)$
  $P_{i+1} \leftarrow optimizePolarity(CO,\, K_0; \mathcal{R}_P,\, \mathcal{R}_O)$
  $K_{i+1} \leftarrow updateMarketStructure(CO,\, P_i)$
}
Return $\{K = (\lambda, \theta),\, P; \mathcal{R}_O,\, \mathcal{R}_P\}$

III. Data

We used the product reviews from Amazon, which is stored in XML format and is available to public on the web.[1] Ideally, we would like to obtain a dataset where the ratings for specific evaluation criteria are given along the review texts. Unfortunately, we were unable to collect such dataset, and hence the sentence-level polarity evaluation can only be performed through human inspection. In other word, even though a few examples will be provided, there is no summary statistics for opinion-level polarity evaluation.

Our Amazon dataset includes 8 product categories from which we choose the reviews of electronic devices for training. The reason is that compared to other product categories, electronic devices have reasonably clean dimensions for evaluation – such as processing speed, price, battery life, etc. The key information we extract from each review is the review text and the rating, which ranges from 1 to 5.

The training set consists of 100 reviews comprising of 643 sentences in total. We test our trained model on three different testing sets: (1) 50 reviews from the same electronics domain, (2) 50 reviews from books, and (3) 50 reviews from kitchen and hardwares.

IV. Model Parameter Estimation

The very first step in our model is to read in the XML review files, parse each sentence within the review into a syntax `tree` structure, and store it in a `Vector<Opinion>`. (So, each sentence now transforms into an *opinion* whose *orientation* and *polarity*

[1] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

are not yet initialized.) Fortunately, Stanford NLP group provides a Java syntax parser, which utilizes Viterbi algorithm internally to extract the most likely syntax structure of the input sentence. We now proceed to the problem of extracting orientation and polarity.

A. Extracting Orientation

4.1.1 Choice of Orientation Dimension

In order to build the set of orientation words (i.e., the words to represent the orientation of an opinion), we first collect all unigram and bigram tokens from all review texts, treating each unigram, bigram token to be one orientation dimension. The words with no or very little information about the orientation of an opinion (e.g. "a" or "the") are eliminated from these N-gram sets. The next step is to transform words as an original form, if necessary. For example, “cars” will be transformed as “car”, and “brought” or “bringing” will be transformed as “bring”. Also, when a token comprising purely of numbers appears, it will be stored as `<NUM>`. Then, we merge filtered N-grams applying various rules using `wordnet` library: For example, we merge all synonyms into one "bucket" of orientation – "price" and "cost" can be recognized as representing an identical orientation dimension. Another rule is to merge all antonyms – treating any opinion containing "fragility" or "robustness" as expressing the same orientation (but with opposite polarity) dimension. Other examples include merging all hypernyms/hyponyms, which defines the hierarchy of words - "chair" is a type of "furniture." It is also possible to treat the pair of the current word and the following word’s part of speech as a type of orientation dimension. Finally, the extracted orientations is stored as a binary vector of dimension m.

Given the parsed sentence structure, we can extract all candidate phrases that can be an opinion. Any of noun phrase (NP), verb phrase (VP) and propositional phrase (PP) can express an opinion, and any of adjective, verb or noun can be an orientation (evaluation criteria) word. For example, consider that the verb “last,” the noun “longevity,” and the adjective “long” can all refer to the life span of a product. The question is how to represent them.

We consider all these possibilities and experiment with the various combinations of phrase types and compared their performance. For example, in one experiment we choose to represent an opinion only with noun phases (NP) and, in another experiment, with a combination of NP and VP, and so on. At the same time, we can apply the same combination technique among nouns, adjectives and verbs to pick the orientation words.

4.1.2 Customized Features

B. Extracting Polarity

We describe polarity extraction in two steps. The first section present the model for opinion-level polarity. The second part deals with the word-level polarity.

1) Extracting Opinion Polarity: Following the approach of Zagibalov and Carroll [4], the polarity p_i of an i-th opinion o_i is modeled as follows:

$$p_i = \mathcal{R}_P(o_i) = \sum_{\forall w \in o_i} (-1.0)^N P(w) \quad \text{(IV.1)}$$
$$\text{where} \quad N = \sum \mathbf{1}\{w = \text{negationWord}\}$$

where $P(\cdot)$ is a dictionary of word polarity. We explain more about P in the next section. Roughly speaking, the polarity of an opinion is the sum of individual words polarity whose value flips whenever negation words (such as “not” or “never”) appears. Since we will optimize/update the values of P later in the process, there are two key requirements for $\mathcal{R}_P$: (1) it has to be differentiable with respect to P and (2) the word polarity returned from P has be bounded by some value. The reason for these requirements will be clear in Section IV-E.

2) Extracting Word Polarity: We now describe the steps for extracting polarity at word-level. There are three major obstacles when constructing the word polarity dictionary P. The first is to decide which words to include. (Certainly, we cannot include the words we have not seen.) Second, in relation to the previous point, how to deal with unobserved words during the testing stage need to be addressed. Lastly, specifically to our model, the values of word polarity has to be bounded above and below by some value while preserving the differentiability of Eq. IV.1.

For the first problem, we use boosted decision tree algorithm to extract the best polarity words. In our implementation, each intermediate node in a decision tree represents a polarity word, and input for the tree

is one whole review text. Each input chooses the child node to traverse down based on whether a particular polarity word is contained within its text. While building the trees, polarity words are chosen such that it maximizes the entropy, which sorts the review texts by their rating scores as evenly and ideally as possible. The number of total trees to be built while the boosting procedure to be 100.

We gave higher frequency weights to those words that takes place closer to the root node in the tree, because the earlier that polarity word is chosen in the tree, the more efficiently the algorithm classify the ratings (and the polarity of the subnodes).

As the final step of building the tree, we set the initial polarity value of each word ranging between -1.0 and 1.0. The sign for this quantity is decided by comparing the average rating score of the entire training set and the average rating score of the review texts that contains the target word – if the latter is higher the sign is positive, otherwise negative. Likewise, the magnitude of the polarity is computed as the frequency of the target word divided by the cumulative frequency of all the other words.

Now, for the second problem, the solution is easy. For any unseen words, we simply examine the `wordnet` searching for synonyms. If a synonyms that also exists in our polarity dictionary, then we use that value. If no such word is found, then we simply return zero as polarity.

Lastly, the subtle point. We represent polarity as a value from $arctan(\cdot)$ function internally in our code. In other words, if the polarity of a certain word is set to be 1, we internally store that value as $tan(1)$. The reason is to preserve the eligibility for optimizing P. Note that $arctan$ is bounded by $(-\pi/2, \pi/2)$ and its derivative $arctan'(x) = 1/(1 + x^2)$. Imposing this internal structure, Eq. IV.1 continues to satisfy the two requirements for optimization.

C. Clustering of the Opinions

Now the orientation and the polarity of all the opinions are initialized, we can cluster them into k-groups. The purpose of this clustering is two folds: (1) to reduce the orientation dimension m into a smaller k-dimensional profile space, and (2) to find the "centroids" of different *opinion*s so we can estimate the pricing kernel λ in the next step.

Intuitively, k represents the number of product evaluation criteria. For example, if consumers only cares about two factors (e.g., price and design) when reviewing the electronics, we expect our model to classify the opinions into two groups. Unfortunately, this hypothesis cannot be tested as our input data does not specify the number of evaluation criteria (i.e., k is not known).

In our implementation, the optimal k is searched from the range from 2 to 8 so that the sum of the within-cluster distances are minimized. For each iteration, we initialize the centroids by averaging the input orientation vectors. In practice, if a prior expectation for the value of k is available, that number may be used as a initial value for clustering.

The resulting centroids of k-means clustering has a great implication for our model. First, these centroids represent the average sentiment over all the reviews in the training data. Second, they act as a set of "standard" evaluation-axis where opinions can be projected onto. Shrewd readers may notice that these centroids are essentially the mapping, or the *profiling kernel* θ, from the orientation vector to the product profile space. If we project the average opinions of one reviewer onto θ, we get the scorecard from that reviewer. In other words, letting x_i and p_i denote the orientation and the polarity of opinion i, the product profile f from the review R is:

$$f_R = \sum_{\forall o_i \in R} (\theta x_i) p_i \text{ where } \theta = centroids$$

Note that the success of this step is critically dependent on the quality of the orientation vectors. The more the orientation vector contains relevant informations, the better the performance of the clustering step. We believe that the orientation extraction based on decision tree has contributed significantly in this regard. (More discussion will follow in the result analysis section.)

Finally, we want to point out that, by no means, k-means clustering is the only algorithm that can be applied for this step. Any techniques for reducing the

dimensionality – e.g., PCA or ICA – can be applied here as long as the two objectives mentioned earlier are satisfied.

D. Estimating Pricing Kernel

At this moment, let us step back and take a look at the overall picture. In the end, our objective is to make a meaningful prediction for product rating. Revisiting Eq. II.1, we now have all the ingredients for make a prediction $\hat{r}$ – except λ. Looking closely, estimating λ turns out to be just a simple OLS regression problem. Therefore, the SSE-minimizing λ is:

$$\hat{\lambda} = (F^T F)^{-1} F^T \bar{r}$$

where$F = \sum \theta x_i$ is a matrix of profiles (averaged over each review) and $\bar{r}$ is the re-centered actual rating (meaning 3.0 is subtracted from the actual rating).

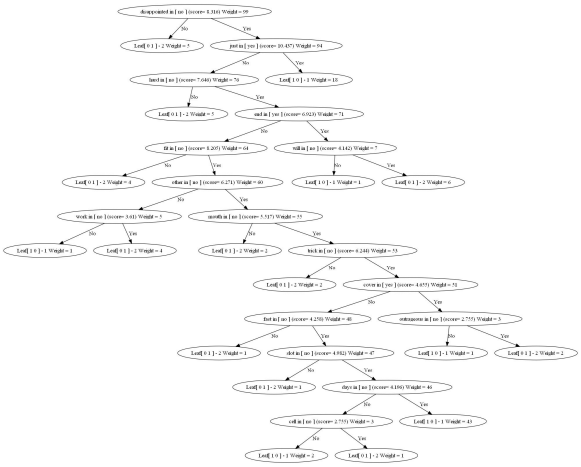
There is nothing theoretically complicated in this step, but there are a few practical issues we want to point out. (1) If there is a review with zero polarity, $F^T F$ becomes singular and thus the inverse cannot be found. We resolve this problem by assign a small polarity (0.1) on that training review. If the training size is large, this is not normally a problem. (2) Since the orientation vector is unscaled, the raw values of λ might seem out of the resonable ranges as a weighting vector at first. In this paper, we report a normalized λ unless specified otherwise.

E. Optimizing Polarity

The final step of our training is to optimize the word polarity P to minimize the prediction error. Since the extraction of a opinion polarity is performed by a differentiable (w.r.t. P) Eq. IV.1 and thanks to the internal structure of P, we can simply pass the arguments to any standard optimizer (such as LBFG optimizer included in the Stanford NLP packages) to find the optimal values for P in the sense that it minimize the RMSE of the rating predictions.

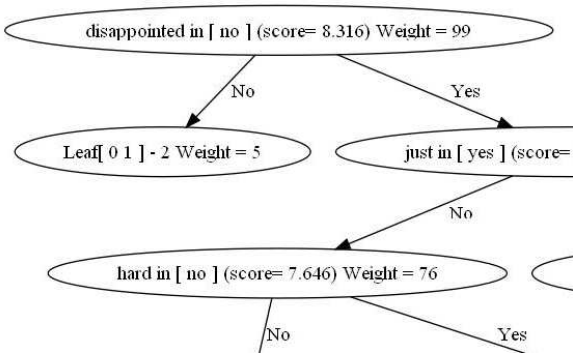
We iterate through the entire steps in this section until the in-sample RMSE value converges. Between each iteration, all the parameters except the orientation and polarity extraction rules (Eq. IV.1, respectively) are being updated. This completes the training or our model.

Figure V.1. An example of a decision tree for orientation extraction



Each node/word represents a dimension of orientation vector

Figure V.2. A zoomed-in version of the Decision Tree



V. Result Analysis

While the previous section focuses on explaining the underlying theories behind the model, we present the concrete examples/results from each step and discuss the findings.

A. Outputs from Initialization

1) Initial Polarity Word Extraction: Figure V.1 shows one of the decision trees generated by our orientation extraction algorithm (Max-Entrophy). As mentioned earlier, we use the 100 review articles, which is comprised of 699 opinion phrases/sentences. Each intermediate node in the tree represents a polarity word, and the traversing-down direction is chosen by whether the word exists in the particular opinion. In this training data, the total number of polarity word candidates (all NP and VP words) are 1500; after running the 10 iterations of building the decision tree, the number

Table I
TOP 7 POSITIVE POLARITY WORDS

Positive Word	Polarity
great	100.0
easy	90.0
many	67.0
trick	53.0
fit	64.0
fast	48.0
outrageous	3.0

Table II
TOP 7 NEGATIVE POLARITY WORDS

Negative Word	Polarity
disappointed	-99.0
hard	-76.0
end	-71.0
careful	-52.0
cover	-51.0
poor	-30.0
refund	-29.0

reduces to 65 (so now $m = 65$ effectively).

The weight of each intermediate node is directly used as a polarity magnitude of that word, and the sign is decided by analyzing its children: if the average rating of child reviews who contains the word is lower than that of all reviews that reached the node, the sign of the polarity is positive; otherwise, it's negative. To put it loosely, the polarity is positive only if the existence of the word in reviews increases the average rating score.

We iteratively build the trees until we get more than a certain number of polarity words. This is necessary for later procedure because the less the number of extracted polarity words is, the fewer tools we have to analyze and predict the rating score of a new article. When only 1 tree was used, processing 50 reviews gave only 23 polarity words. On the other hand, as we increased the number of trees being built up to 10 we could collect as many as 56 polarity words.

Table I and II show the top 7 positive and negative *initial* polarity words generated by decision trees. The most positive words are 'great', 'easy', 'many' trick', 'fit', 'fast' and 'outrageous', whereas the most negative words are 'disappointed', 'hard', 'end', 'careful', 'cover', 'poor' and 'refund'. As expected, any articles containing 'disappointed' or 'poor' give a review rating score lower than 3 while the most articles containing a word like 'great' or 'easy' is given a almost full rating score. The words that do not seem to have an ideal polarity value are re-calibrated in the later training process.

Figure V.3. Performance on various choices of POS for the selection of orientation words

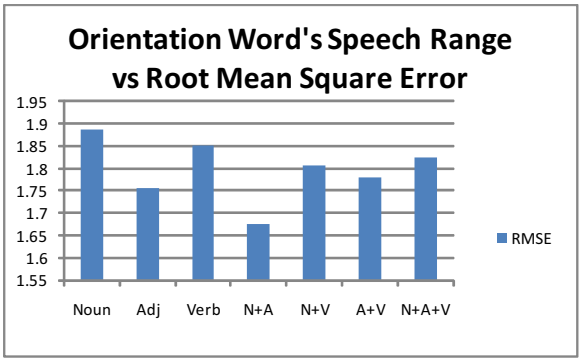
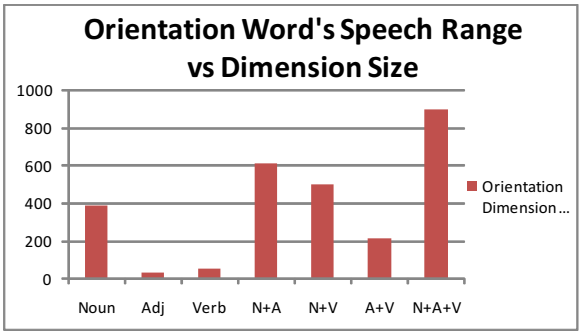


Figure V.4. Choice of POS for selecting orientation words and its effect on dimension size m



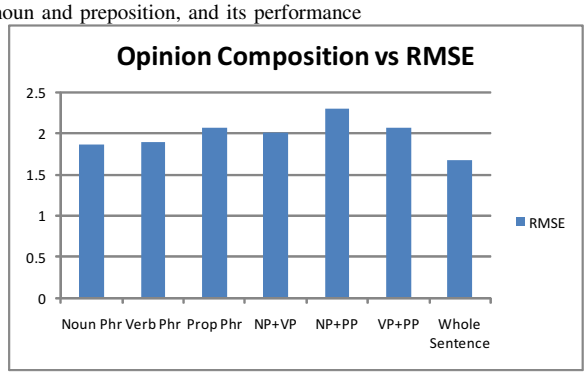
2) Deciding Appropriate Orientation Dimension: We use the polarity words extracted in the previous section to represent orientation. But, here is an dilemma. If we loosen our bar and includes lots of words, the quality of each orientation deteriorates and the estimation of pricing kernel λ will be very noisy. On the other hand, if we decide to choose only the highly-informative words, we then lose the dimensions for meaningfully characterizing a product evaluation dimension.

FigureV.3 shows our effort to balance out the performance by testing on varying choices of part of speech (POS) for orientation words. In general, we get a better performance when adjectives are included as a possible dimension for orientation vector. The best score of 1.673 is achieved when adjective and noun type are considered as an orientation word. In our training dataset, some of adjective and noun combination tokens

Table III
ORIENTATION BUCKET WORDS

Bucket Index	Words
[1]	blessed damned goddam darned blasted goddamned damn infernal blamed deuced goddamn blame
[2]	need requisite requirement essential indigence motive motivation necessity pauperism penury prerequisite pauperization necessary want demand
[3]	adaptor adapter transcriber arranger
[4]	tuner wireless radio radiocommunication

Figure V.5. Choice of various set of opinion compisition among verb, noun and preposition, and its performance



recognized by our orientation extractor include "long duration," "fancy features," "fixed price," etc. None of these paired words are recognized in the orientation extractors when using a single part of speech.

Obviously, the number of dimension for orientation is proportional to the size of orientation word candidates. The size of adjective and verb candidates are relatively small compared to nouns in the traning texts. This is indeed true in English language because many adjectives and verbs share the same/correlated meaning than nouns do. Consequently, our implementation includes the method of merging word groups that have any overlap in its meaning. All synonyms shares the same orientation bucket. We present several examples produced by our implementation as in Table III.

3) Applying Various Speeches on Orientation Words: In our implementation, we eliminate any opinions that yields zero orientation vector, meaning it does not contain any meaningful information on any evaluation criteria we estimate from polarity words. Consequently, as the number of words in an opinion decreases, the likelihood that its words fall into any of the orientation dimension decreases. Elimination of opinions implies reduction in training data size, which is not productive. For sentences like "it only costs $249 dollars," when only NP phrase are used for orientation extraction, it fails to recognize "costs" as a possible orientation word. When the whole sentence was treated as the domain to extract orientation words, the verb "costs" was classified into the orientation bucket representing prices, with other words such as "expense", "weigh", "price", etc. Therefore it was strongly recommended to use each whole sentence as an opinion to widen the possibility to recognize any orientation words in it.

Table IV
TRAINED MODEL PARAMETERS

Parameters	Value
Orientation dimension m	8092
Otimal k	2
Size of word polarity P	137

B. Clustering & Market Sentiment Extraction

In the previous section, we discuss the topics of initializing the model and the related findings. In this section, we report the interim outputs and discuss the result. Note that our trained model returns the following: the optimized polarity dictionary P, the profiling kernel θ, and the pricing kernel λ while the *rules* for extracting orientations and polarity are fixed.

Using the 100 electronics review as a training data, the resulting model parameters are summarized in Table IV. However, note that the estimated figures are not exactly replicable as our decision tree algorithm randomly selects polarity word candidates and use them as initial seed words.

Although we cannot transform the profiling kernel θ back to natural language spaces, we instead look for the orientation vectors closest to each centroid. Since we have $k^* = 2$, there are two centroids, which means there are two evaluation criteria that consumers think are important. We print the closest orientation vector here.

```
centroid[0]: another, one, looking, brand,
equal
```

Table V
WORD POLARITY CHANGE

Polarity at iteration:	1	2	3
`impressed`	-1.5822	0.0252	0.0467
`inexpensive`	0.00609	0.04166	0.0722
`wasted`	-0.00608	-0.0314	-0.0646
`fit`	0.01076	0.15423	0.0439

Table VI
PERFORMANCE ON RATING PREDICTION

	# reviews	# right direction.	RMSE Model	RMSE Bm
Electronics	50	33	1.6877	2.131
Kichenware	50	30	1.8620	2.017

```
centroid[1]: stupid, handle,
```

Unfortunately, it is difficult for us to infer what each evaluation dimension represents from the above. (Perhaps the second one may be related to the ease of use?) The changing value of word polarity is also interesting to watch. A few samples of the word polarity are shown in in Table V. Note that the polarity of each word converges as optimization iteration goes on. The sign of the polarity has no significance at this moment. Only when it is multiplied by λ, the value of polarity word has its conventional meaning.

Lastly, the value of λ is shown below:

$$\lambda^T = (-0.2726, 0.9621)$$

C. Rating Predicting

All of the extracted information from the previous sections, we aggregate them here to one cohesive number: a product rating. We report the result of predicting the ratings from our model in TableVI. As before, RMSE is used as a measure of performance.

Two benchmark values are to be compared. The first benchmark is obtained from the 10,000-rep simulations where the distribution of the actual ratings are used to make random guesses. The average RMSE is 2.131. Compared to this number, our model seems to have certain predictive ability. In fact, the percentage of picking the right direction is certainly above 50%, which is encouraging.

However, as for our second benchmark, one may argue that, from a blind reviewer's point of view, her optimal RMSE-minimizing guess for rating is 3. Indeed this statement is true and when 3 is used as the predicted rating consistently, the simulated RMSE turns out to be much lower at around 1.6 – even better than our in-domain model performance.

At first, this result seems to completely invalidate our work. However, be reminded that the real value of our approach is its insight to extract the market structure out of the review texts that is written in natural languages. As evidenced clearly in Section IV, we put a great amount of effort to derive such mechanism and the real output of our model are the estimated parameters λ, θ, and P. Although the RMSE of rating prediction is reported several times in the paper (mostly because it is the **only** measure we can compute), we believe that it is a distance measure for evaluating the value of our research.

VI. CONCLUSION AND FUTURE IMPROVEMENTS

In this paper, we devise an architecture where NLP can be used on the Amazon review data to discover useful market structures. The process is complex and involves the assumptions for the underlying structure for the rating mechanism. Although each step in our model involves a substantial amount of work, the resulting parameters have its own significance and interpretation. While the prediction performance does not meet our expectation yet, there are a few promising signs – such as sensible clustering of the orientation vectors – for the pursuit of research in this direction.

While we put our best effort to make our model as complete as possible, there is a plenty of rooms for improvement and future research. A few examples are (in the order that we think will improve the overall model performance/robustness):

- Resolve the meaning of the pronouns such as "it" or "that" using syntactic tree structure.
- Simplify or relax the underlying model specification
- A better way of extracting orientation/polarity (instead of a linear model currently being employed)

The biggest contribution of our research is not the prediction performance of our model. The idea of using NLP techniques to extract the hidden market structure/dynamics is, in our opinion, a less-studied area of research and seems to suggest a new type of (indirect) information extraction.

REFERENCES

[1] M. Dredze J. Blitzer and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Proceedings of the Association for Computational Linguistics (ACL)*, 2007.

[2] A.-M. Popescu and O.Etzioni. Extracting product features and opinions from reviews. *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005.

[3] J. Rothfels and J. Tibshirani. Unsupervised sentiment classification of english movie reviews using automatic selection of positive and negative sentiment items. *CS224N - Final Project*, 2010.

[4] T. Zagibalov and J. Carroll. Automatic seed-word selection for unsupervised sentiment classification of chinese text. *In Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1073–1080, 2008.