Applying Machine Learning to Product Categorization

Sushant Shankar and Irving Lin
Department of Computer Science, Stanford University

ABSTRACT

We present a method for classifying products into a set of known categories by using supervised learning. That is, given a product with accompanying informational details such as name and descriptions, we group the product into a particular category with similar products, e.g., 'Electronics' or 'Automotive'. To do this, we analyze product catalog information from different distributors on Amazon.com to build features for a classifier. Our implementation results show significant improvement over baseline results. Taking into particular criteria, our implementation is potentially able to substantially increase automation of categorization of products.

General Terms
Supervised and Unsupervised Learning, E-Commerce

Keywords
Amazon, Product Categorization, Classifier

1. INTRODUCTION

Small to medium sized businesses who sell products online spend a significant part of their time, money, and effort organizing the products they sell, understanding consumer behavior to better market their products, and determining which products to sell. We would like to use machine learning techniques to define product categories (e.g. 'Electronics') and potential subcategories (e.g., 'Printers'). This is useful for the case where a business has a list of new products that they want to sell and they want to automatically classify these products based on training data of the businesses' other products and classifications. This will also be useful when there is a new product line that has not been previously introduced in the market before, or the products are more densely populated than the training data (for example, if a business just sells electronic equipment, we would want to come up with a more granular structure). For this algorithm to be used in industry, we have consulted with a few small-to-medium sized companies and find that we will need an accuracy range of 95% when we have enough prior training data and a dense set of categorizations.

2. PRIOR WORK

There has been much millions of dollars spent developing software (e.g., IBM eCommerce) that maintain information about products, buying history of users for particular products, etc. As such there is much work being done on how to create good product categories. However, while large companies (like Amazon) can afford to use such expensive software and sophisticated methods, most companies that sell goods online do not have such resources. We propose to use machine learning as a way to automatically classify products.

3. DATA

3.1 Initial Dataset

For our project, we have acquired multiple datasets with thousands of products and their respective data from Ingram Micro, the leading information technology distributor, which list of product descriptors and resulting categorizations in the form of a detailed CSV document separating multiple features seen in Table 1. The datasets we have chosen to use for the project comprises of the categories seen in Table 2.

Table 1: Product Descriptors

Product Descriptor
SKU
Asin
Model Number
Title with Categorization
Description
1-10 Tech Details
Seller
Unspecified Number of Images
URL

Table 2: Company A (left) and Company B (right) Catalogs

Category	#
Electronics	3514
Camera & Photo	653
Video Games	254
Musical Instruments	212
Kitchen & Dining	173
Computer & Accessories	61
Automotive	48
Health & Personal Care	32
Office Products	30
Cell Phones & Accessories	29
Home & Garden	26
Home Improvement	24
Sports & Outdoors	17
Patio, Lawn & Garden	11
Software	6
Movies & TV	5
Toys & Games	2
Beauty	2
Everything Else	1
Clothing	1
Baby	1

Category	#
Health	768
Beauty	645
Electronics	585
Home	542
Toys	271
Office	255
Sports	238
PC	232
Personal Care Appliances	204
Books	203
Kitchen	160
Wireless	154
Grocery	113
Pet	112
Video	106
Apparel	91
Lawn	81
Baby	68
Automotive	52
Shoes	32
Jewelry	22
Watches	21
Software	8
Music	6

3.2 Parsing the Data Set

Each product comes with a large amount of accompanying data that may not be relevant or may add unnecessary

complexity that we feel would be detrimental to our classification purposes. Of the numerous product descriptors, we decided to keep SKU as a way to uniquely identify products, title with categorization, the description, and tech details. The title in particular contained a high concentration of important product information such as brand and categorization. As a result, we were able to parse out the categorization, but decided to temporarily leave the brand as part of the title and treat it as another word. As a side note, we decided not to incorporate the images as we felt that the task would be more suited to computer vision and image processing.

3.3 Splitting Training and Test Sets

Before creating our training and test sets, we decided to merge all of the categories with less than 0.5% of the overall products into one "Other" category, and then decided to train on a 25% random subset of the data and testing our error rate on the rest.

4. NAÏVE IMPLEMENTATION

Our predominant source of data is in the form of text descriptors about a product; thus, we decided to implement a simple Naïve Bayes classifier and focus on features described in Section 5. The simplicity of Naïve Bayes makes for a great testing environment and feature building, which helped to define the scope of the overall project.

4.1 First Implementation – Naïve Bayes

To begin, we implemented a simple multinomial event model Naive Bayes model with Laplace smoothing on just the 'title' specification. Our accuracy using this implementation was around 70%.

4.2 Modified Implementation – Naïve Bayes

We realized that our classifier was classifying everything as 'Electronics' because the probability of the 'Electronics' category was astronomical compared to everything else. Thus, we decided to remove the prior class probability out of the equation in Naive Bayes.

However, when we break up errors by category, we find the less training data we have on this category, the worse the accuracy. Thus, from our previous observation, we decided to only use categories that have 0.5% of the total number of products (so > ~25 products).

With these changes, our results immediately improved to an accuracy of about 75%.

4.3 Precision / Recall

We found that for Product Catalog A, there was a large bias towards classifying towards the large categories, e.g. 'Electronics'. In other words, for the large categories there was high recall but low precision; for the smaller categories there was low recall but high precision. This effect was also seen in Product Catalog B, but to a much lesser degree as the skewness of the catalogs is much less than in Product Catalog A.

4.4 Filtering Items

One issue we noted for the this classification scheme was that often times, an item just didn't have enough information to classify it properly, or that the words just didn't provide enough information to really provide a situation such that even a human would be able to properly classify one way or another. In the prior case, if there weren't enough words, we just removed the item from consideration, and didn't incorporate that into the classification accuracy. This was especially true for the Company B's catalog, which included books whose titles were often 3-4 words or less in length, and were absolutely failing in their categorizations. The latter situation leads up to the next subsection.

4.5 Multiple Classifications / 'Strong Accuracy'

Upon manual inspection of miss-classifications, the classified found arguably equivalently good classifications. In this case, the classification was 'Electronics' but we classified it as 'Office Products', which also makes sense:

```
Ex: Fellowes Powershred MS-450C Safe Sense 7-sheet
Confetti Cut Shredder
```

We decided to redefine success if the top three categories the classifier spit out were the 'correct' categorization. For each SKU, we took up to three categories that were greater than or equal to the range of the probabilities (categories) (max - min).

Accuracies were low when looking at Product Catalog B (around 50%). *Including multiple classifications increased the accuracy to around 91%*. This is not just a fudge statistic, but actually makes sense in the context of these catalogs. When a company gets a user interested in an analogous category, we want to be able to show him a product to buy if it the analogous category also makes sense.

Using multiple classifications, if only one category was returned, this means our classifier is reasonably certain that this is the correct categorization as the probability this belongs to other categories is much less (in fact, less than 50% of the top categorization according to our heuristic). We term this the *strong accuracy*. Indeed, if we look at our strong accuracy for Product Catalog B for multiple category, our accuracy is 72.7% while our *strong accuracy is 84.5%*.

5. FEATURES

As we use the human generated text content to define a product, we decided on using the bag of words model as

features for each item. However, because the text descriptors are human generated, we encounter a lot of human eccentricities and differences in word choice semantics for similar words. Thus, we incorporate a couple of domain specific ideas to help with classification. The first is per feature processing, and the second is determining which features to use.

5.1 Feature Processing

We apply some standard techniques in text processing to clean up our feature set: stemming/lemmatization, lowecasing, and stop word, punctuation, and and number removal.

5.1.1 Stemming/Lemmatization

We use stemming to narrow our overall feature space so as to help with the lack of features per item.

```
Ex: case, casing, cases -> cas
```

Overall, we see mixed results for stemming. As seen in Table 3, while stemming shows improvement for both companies over the baseline without any feature processing, it clearly negatively affects the overall accuracy when combined with other feature processing for Company A, while it positively affects overall accuracy for Company B.

5.1.2 Stop Word Removal

We found as in much text, that there were many common, short function words, often prepositional terms and other grammatical syntax fillers that were found in many of the product descriptions. While we have not yet attempted to build a domain-specific (i.e. for product descriptions), we used a standard set of Porter stem words.

```
Ex: the, is, at, who, which, on
```

Stop words turned out to be by far the most impressive feature processing technique we used, in so much as dominating both catalogs for accuracy boosts (Table 3). In fact, for Company A, it resulted in nearly a 10% increase in overall accuracy.

5.1.3 Number Removal

The decision to remove numbers was a result of looking at our data and realizing that many numbers had little to no relevance to a given category, and that the scatter (and range) of numbers only would adversely impact a given categorization. Because there are a limited and finite number of training examples, but an infinite number of numbers, we believed that we could not fully capture the information of numbers for a given category.

```
Ex: 5-10, 2, 6, 15.3
```

The slight performance increase for Company A is offset by a slight performance decrease for Company B (Table 3). We attribute this to the fact that for certain categories, specific numbers may show up more, and thus affect the resulting categorization.

5.1.4 Punctuation Removal

Removing punctuation was another result of looking at our data and noticing that humans have a very large variance in phrasing and word choice. This is especially true for words that may have multiple accepted forms, or words with punctuation in them. Also, because we parse the item descriptors on spaces, any punctuations that are in the phrase are left in, including ellipses, periods, exclamation points, and others. In addition, words that are concatenated are often used differently.

```
Ex: don't, dont, 3D, 3-D, period., period?
```

Punctuation removal was the second most effective feature normalization method used for Company A, while it was a (close) third best for Company B (Table 3).

5.1.5 Lowercasing

Sentences in the English language tend to have the first word capitalized. In addition, different people will capitalize different words intentionally or otherwise, depending on their intent interpretation of the word, of choice of capitalizing acronyms. Clearly, this is not always a useful normalization, and it can go very wrong especially when dealing with product names vs. generic objects.

```
Ex. President, president, CD, cd, Windows, windows
```

As expected, lowercasing doesn't help very much (Table 3), most likely because any benefits we receive from lowercasing occurrences such as "Knives are great" to "knives are great" are offset by "Windows Vista" to "windows vista", which have very different meanings. Perhaps it would've been more useful to just lowercase the word of the item if there's no chance that it's a product name, but that would've been particularly difficult given the infinite amount of product names out there.

Table 3: Showing Feature to NB Classification Accuracy

Feature Normalization	Company A NB Accuracy	Company B NB Accuracy
Stemming	.775	.519
Stopword Removal	.843	.521
Number Removal	.794	.492
Punctuation Removal	.803	.515
Lowercase	.778	.507
Include Certain Descriptors	.829	.498
None	.747	.497
All	.796	.524

5.2 Choosing Features

Secondly, we use domain specific knowledge about the product catalog andwhat is contained in it. In addition to the title, items have brands and other descriptions such as specifications that we may use. We felt many of the

specifications such as size and weight wouldn't benefit the overall classification, but we felt that the brand might have a large impact, and we when we tested it, we saw that the impact really depends on the dataset used (Table 3).

5.3 Calculating the importance of features

In order to 1) understand what features we can add and 2) and weighting the features we have by how important they are in distinguishing between categorizations. When we used the conventional MI formula, we found that two of the four terms in the summation were actually skewing the MI calculation for Company Catalog A -- in particular, the case of calculating the term of not this feature and in this class and not this feature and not in this class. This is due to the skewness in the distribution of categories (where Electronics categories are ¾ of the catalog) and the modification is not applicable for Company Catalog B as the distribution of categories is more uniform. The modified MI formula we used for this:

$$MI(x_i, y) = \sum_{x_i \in \{1\}} \sum_{y_i \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

Table 4: Most and Least Informative Words in 'Health & Personal Care' for Company A's Catalog

Most informative words	M.I.	Least informative words	M.I.
tim	0.00160	M30p20	2.5e^-8
beard	0.00084	pz462	2.5e^-8
wahl	0.00070	superspee	2.5e^-8
hair	0.00049	stick	2.5e^-8

Table 5: Most and Least Informative Words in 'Health & Personal Care' for Company A's Catalog

Most informative words	M.I.	Least informative words	M.I.
black	0.1437	rapid	0.1291
pack	0.1357	wet	0.1291
set	0.1356	orthot	0.1291
cas	0.1355	champ	0.1291

5.4 Feature Selection

As we are using a modified set of the words to describe products as our features, not all these features will be useful for classification or differentiation between categories. We considered forward or backward search to select our features, but as our features run in the order of thousands, we decided to try filter feature selection instead. To do this, we first looked at the distribution of MI in our two catalogs. In Company Catalog A, the spread between the most and least informative features was large. We had some interesting findings (both with using max of 3 possible categories):

- Using the most informative features increased the accuracy in both Company catalogs.
- For the Company A product catalog, not using the least informative features along with the most informative helped. We found a slightly significant jump of 2% accuracy features. When we do use the least informative features, the accuracy halves.
- For the Company B product catalog, when we do not include the least informative features, the accuracy drops. However, when we do include the least informative features, there is a slightly insignificant jump of 0.5% accuracy.

We surmise that this weird behavior is due to the mutual information not properly describing or capturing the 'least informative features', due to the skewness of the distribution of categories in Company A product catalog.

6. DIFFERENT CLASSIFIERS

We decided to continue working with our version of Naive Bayes after trying a few different classifiers. We used the Orange (http://orange.biolab.si/) machine learning library for Python to try out a few different machine learning algorithms. We tried this on Company A's catalog and the table below has the accuracy and time taken for each algorithm

Table 6: Different Classifiers and Accuracies

Algorithm	Company A Catalog Dataset Accuracy	Time Taken
Naïve Bayes	79.6%	~3 seconds
K nearest neighbors (k = 5)	69.4%	~4 minutes
Tree classifier	86.0%	~8 hours

Here in Table 6, we can see that the Tree Classifier performed the best, but took a few hours to complete. In a real commercial situation, we see an algorithm like the Tree Classifier (or a more sophisticated algorithm as Random Forests) as a back-end deamon. In addition, this tree gives a potential to more deeply understand a product catalog (more so than a fixed set of categories). However, Naive Bayes achieved almost the same accuracy as the Tree Classifier in significantly less time. We thus stuck with our Naive Bayes classifier to develop features for (see Section 5.2) and select features from (see Section 5.3). We tried the kNN (k-nearest algorithm) for different values of k, but they all performed poorly, with k=1 to 5 performing the best.

7. CONCLUSIONS

We found two main difficulties working with two datasets. In Company A's product catalog, the distribution of categories was severely skewed towards one category ('Electronics') and we tended to always classify products into the larger categories. As such, the larger the category the better the accuracy. In Company B's product catalog, we found many of the products had very short descriptions and there many fewer words than in Company A's catalog.

Along with implementing regular text processing techniques (stemming, etc.) we found that using mulltiple classifications, the notion of strong accuracies, and filtering products that had greater than a certain number of words -- helped the

accuracy significantly and made sense in the context of categorizing products. In particular, we believe that using multiple classifications is one of the main ways this can be extended to incorporating user data from these company websites and offering collaborative filtering capabilities.

We plan to continue this project in the context of a startup and we would like to work towards a collaborative filtering platform for small and medium sized businesses by first making our classification and categorization of products better for the current catalogs and robust for different types of product catalogs. We would also like to implement unsupervised learning to obtain sub-categorization of existing categories.

7.1 Product-Dependent Features

So far, we have use the bag of words model and have not incorporated features that are domain-specific:

- Bi-words: Many brand names and descriptions have words that occur together, e.g., 'digital camera'. We would like to capture these.
- Upweight the title and pick the best percentage of upweighting through a model selection method.
- Use our mutual information features on how to weight the features (the ones that differentiate between classes are more important). We already have this, but this will be important when we have different types of features.
- Add spelling correction as we noticed there are many misspelled words in the catalog.

7.2 Explore More Sophisticated Methods

We chose Naive Bayes as it is the simplest method we could implement. When we tried a few other algorithms using Orange, the Tree classifier looked promising. However, since it took around 8 hours on the entire set of data, we did not pursue working with this algorithm for this project. We would like to pursue variants of the Tree classifier (such as Random Forest) by exploring a subset of the data.

7.3 Unsupervised

Supervised classification with product catalogs is limited to the categories that we train on. To find more granular sub-categories, we need to be able to discover structure within categories of products. We are considering using an unsupervised hierarchical clustering approach. In addition, this would allow us to categorize and give structure to catalogs where there is no existing product catalog (for example, products for a new or sparsely explored market).

8. REFERENCES

[1] Witschel, Hans F., and Fabian Schmidt. "Information Structuring and Product Classification." TSTT 2006 Beijing. Web. 2011.

[2] *UNSPSC Homepage*. 2011. <http://www.unspsc.org/>.

[3] "eCl@ss - The Leading Classification System." *Content Development Platform*. Paradine, 15 June 2011. Web. 2011. <http://www.eclass-cdp.com>.