# Automatic Product Classification and Clustering Solutions in a Retail Context

Rohit Kaul and Rajiv Bhateja

*Abstract – In this report we propose a methodology to automatically classify products and cluster similar products together. This enhances user interaction and performance metrics for e-commerce (product shopping) websites, with applications in product search, site navigation, product comparison, etc. For this project, a variant of the multinomial Naive Bayes algorithm was used for classification. We present our findings of key aspects that demonstrate its accuracy. A clustering methodology was developed to handle large data-sets, overcoming the $N^2$ complexity, in a two-step approach. The first step of clustering utilizes Locality Sensitive Hashing, the second step uses a Shingles matching method. We describe a methodology that allows trading cluster-size versus similarity of items within a cluster and propose a way to use clustering to increment classification accuracy.*

## 1 Introduction

The topic of product classification and clustering is central to providing a good user experience, intelligent search, and revenue for e-commerce web sites. Companies like eBay and Amazon do not require the merchants to provide a product category – it is automatically deduced, allowing scalability and reducing deliberate or unintentional mis-classification, thus enhancing user experience, customer satisfaction and higher transaction rates.

There are several challenges that need to be overcome in order to build a robust product classifier and clustering algorithm, starting with developing a consistent taxonomy and a data set sufficient for validation. Clustering is an $O(N^2)$ problem, and requires computationally efficient solutions for large data sets. Small clusters, though very strongly tied, are not as interesting (or useful), but larger clusters run the risk of introducing odd-balls that make the cluster inconsistent.

## 2 Data Collection

Product data was obtained by crawling an existing e-commerce website until we obtained approximately a million items from downloaded web-pages. After writing parsers to extract title, description and product categorization from the downloaded pages, the extracted data was then cleaned (stemmed, converted to lower case, and cleansed of stop-words). Downloaded products were not unique since multiple merchants could provide the same products, and also due to overlap of products across different URLs. All the downloaded products (including duplicates) were used for clustering in order to verify performance on a large dataset. For classification, we limited the data-sets to approximately 20,000 items for training and 70,000 products for cross validating the classifier accuracy.

## 3 Classification

### 3.1.i Methodology

For classification, our objective was to extract the product taxonomy from the downloaded products, form the taxonomy tree, and build a classifier to map products to the top category level (L1) with 16 classes and to the second level (L2) with 526 classes. We started with a simple multinomial Naive Bayes algorithm [KIB1] and evolved to an extended model [PUU1]. With each experimental step, we updated the feature set to incrementally improve classifier accuracy. Our simple multinomial Naive Bayes model takes the form,

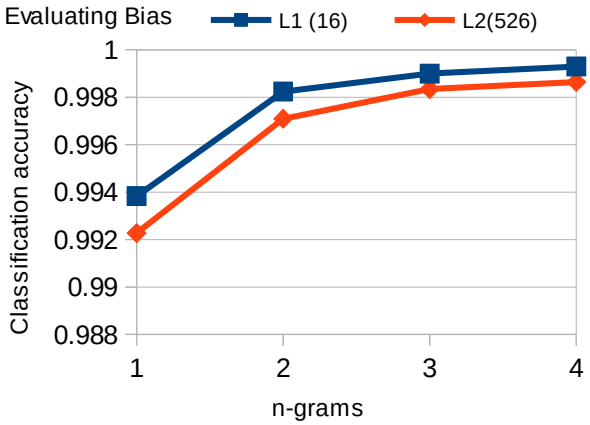
$$p(t_i|c) \propto \prod_n p(w_n|c)^{f_n}$$

Where, $p(t_i|c)$ is probability of a test document $i$ in class $c$, $p(w_n|c)$ is the probability of a word $n$, given class $c$, and $f_n$ is the count of word n in the test document. The extended model uses TF-IDF (term frequency, inverse document frequency) in the form, $f_n = \log\left[1 + \frac{w_n}{s(w)^{a_1}}\right] \frac{1}{s(w)^{1-a1}} \log\left[max\left(1, a_2 + \frac{m}{m_n}\right)\right]$ ,

where, $s(w)$ is the number of unique words in the document, $m$ is the number of training documents, $m_n$ is the

number of documents in which word *n* occurs; and $a_1$ effects length scaling and $a_2$ affects IDF weights. For the results below, $a_1$ and $a_2$ were set to 1 and 0, respectively.

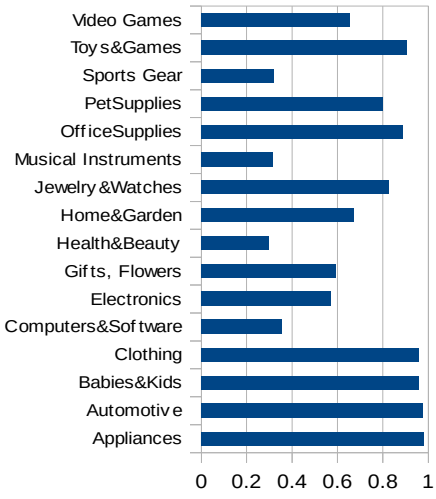
Using a training set of ~20k, we evaluated bias, and the classifier showed excellent accuracy on the training data. Adding higher number *n-grams* was marginally useful, so we limited our validation to bi-grams, based on concerns of over-fitting and computational cost.




### 3.1.ii Validation

For validation, we started with a set of ~70,000 products. The most notable trials to improve L1 and L2 classification accuracy are described below. A key factor was the fraction of tokens not seen during training.

| *Trial* | *Description* | *L1 acc.* | *L2 acc.* | *Test size* |
|---|---|---|---|---|
| 1 | Single tokens, title, simple MN Bayes (SMNB) | 0.72 | 0.38 | 71360 |
| 2 | Unigrams, bigrams, title, simple MN Bayes | 0.73 | 0.38 | 71360 |
| 3 | Unigrams, bigrams, title, description, SMNB | 0.77 | 0.39 | 71360 |
| 4* | Unigrams, bigrams, title, description, MN Bayes TF-IDF | 0.79 | 0.39 | 71360 |
| 5 | Same as 4, but limit to products with unknown unigrams, bigrams less than 50% | 0.85 | 0.47 | 50243 |
| 6 | Same as 4, but limit to products with unknown unigrams, bigrams less than 30% | 0.95 | 0.84 | 10014 |
| 7 | Same as 4, but limit to products with unknown unigrams, bigrams less than 10% | 0.99 | 0.97 | 1489 |




L1 classification accuracy for Trial 4.

## 4 Clustering

$N^2$ clustering over a million items is prohibitively expensive, therefore we researched various approaches that can first segment the products into large, reasonably similar clusters. In order to segment the data into first level clusters (*super-clusters*), we experimented with using *k*-bit Locality Sensitive Hashing (LSH) [ULL1] on product titles to produce a consistent hash even if some of the terms used to produce the hash were different. LSH represents similarity (*S*) between product titles ($T_i$) using probability distributions over hash functions, $h(T_i)$:

$$S(T_{1,}T_2)=P_{h\in H}[h(T_1)=h(T_2)]$$

LSH is computed by combining hashes for different tokens at a bit level. For LSH, we used each keyword as a token. Each bit place-holder is multiplied by +weight if the bit = 1 and -weight if the bit = 0. For this project, we used a 64-bit hash. This resulted in an array of doubles of size 64, which we then mapped back to an 8-byte integer (sum(weight) < 0 = 0 bit and sum(weight) > 0 = 1 bit for all 64 bits). The Hamming Similarity [VIL1] between two n-bit vectors *x* and *y* is defined as, $HS(x,y) = Count(1\{x_i = y_i\})$ and the Hamming Distance is given by {1 – HS(x, y)}. Our goal was to minimize the Hamming distance for related product titles so that they could be clustered easily. In an ideal case, the Hamming distance between related products would be zero.

For computing LSH, a Gaussian weight function based on the token frequency (*f*) across all products was used.

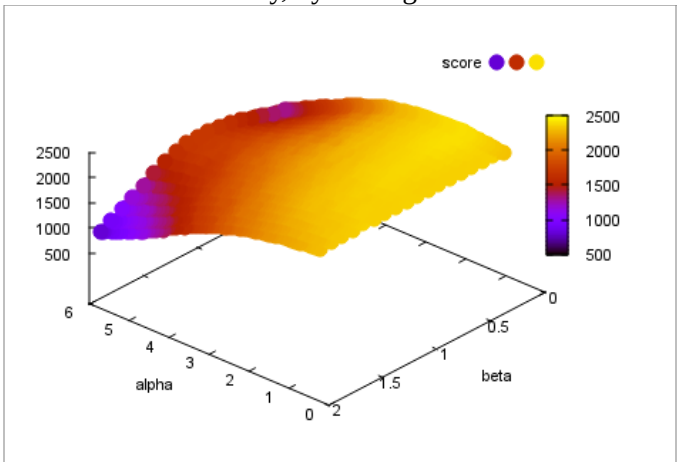
The intuition here is that if a token was very specific (e.g., measure of size, or material type) we did not want it to increase clustering distance from other similar items. At the same time, if a token was too common (e.g., color), that would also skew the clustering. Therefore, the weight of a token *x*, was chosen as $w(x) = k\, e^{-\left(\frac{(\log(f)-\alpha)}{2\beta^2}\right)}$, where *k* is a scaling constant, and $\alpha$ and $\beta$ affect the offset and the shape of the Gaussian curve, respectively. If the Gaussian weight function is fatter, then the LSH hash is influenced by a larger number of tokens, and if its narrower, then effectively a smaller number of tokens affect the final hash.

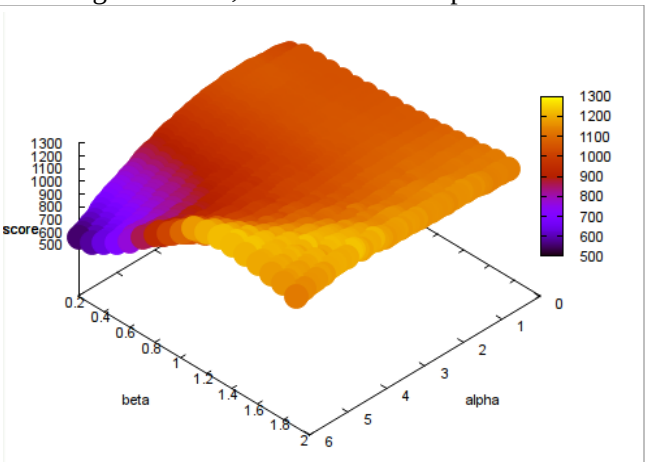
## 4.1 Top Level Clusters

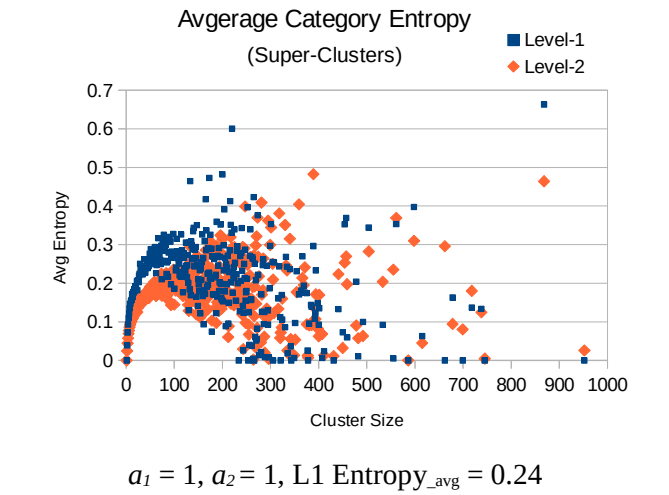
The primary motivation behind super-clusters is to reduce the $N^2$ problem from *N*=O(1M) to ~ O(1K). The clusters produced need to have related items, but produce as many large clusters as possible, which can then be subsequently used for producing sub-clusters. However, as the cluster size increases, the similarity of products inside the clusters reduces, and needs to be balanced with the desire for larger clusters. Therefore, $\alpha$ and $\beta$ in the above equation need to computed to maximize the following score: $\left(ClusterSize_{avg}^{a1} \times ClusterSimilarity_{avg}^{a2}\right)$

### *4.1.i Training*

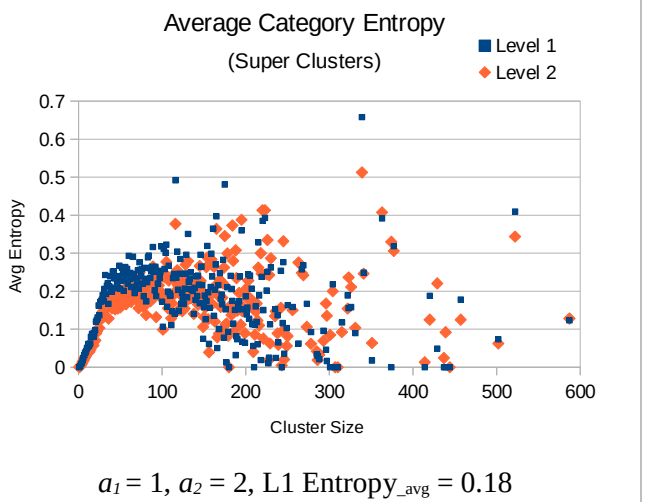
If ($a_1$ >> $a_2$), it will produce super-clusters that are large with fewer similar items, and vice-versa. As a measure of cluster-similarity, we use average *shingles* based similarity (see 4.2 below). The values of $\alpha$ and $\beta$ to use is determined numerically, by finding the maximum score on a training data of 50,000 random examples .




$a_1$ = 1, $a_2$ = 1, L1 Entropy$_{_avg}$ = 0.24




$a_1$ = 1, $a_2$ = 2, L1 Entropy$_{_avg}$ = 0.18

### *4.1.ii Testing*

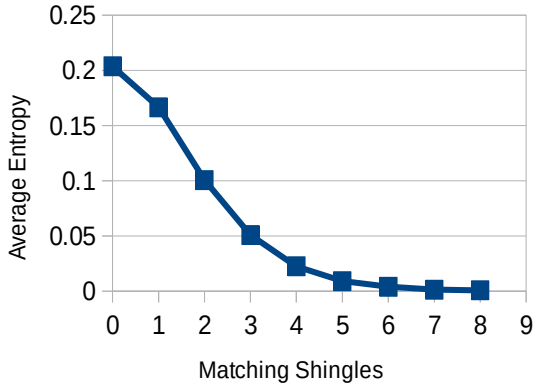
In order to measure the outcome of super-clusters, we can compare the entropy of category distribution, $\sum_{i \in category} -p_i \log(p_i) / Entropy_{max}$ . A higher entropy implies lesser similarity within a super cluster. We test two cases, $a_1 = a_2 = 1$ and in the second case, $a_1 = 1$ and $a_2 = 2$. The max cluster size in first case is ~ 2000 (not shown in graph to reduce clutter) compared to ~ 600 for second with 33% higher entropy than the second case. Given the primary motivation behind super-clusters, we decided to use $a_1 = a_2 = 1$.

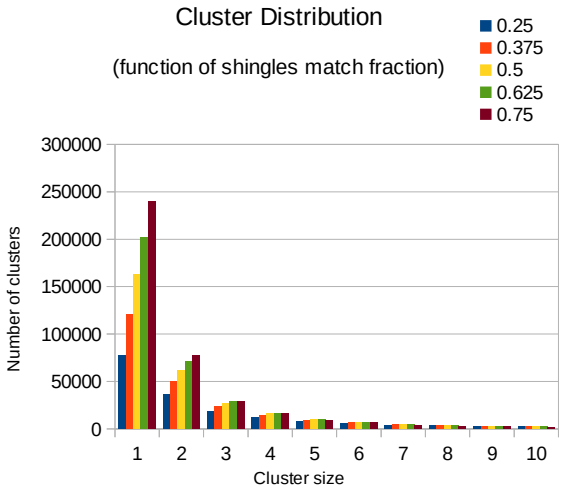
## 4.2 Second Level Clusters (sub-clusters)

The sub-clusters are the final clusters, and need to be very similar products. Subclusters are computed within each "super cluster" using *shingles matching* approach. Its is comprised of *n* min-hashes of a given title, run with a different hashing function each time. The similarity between two ShingleHashes is a measure of Jacccard Similarity |A∩B|/A∪B|. For this project, we used 8 hashing functions. All pairs of items within a super-cluster were compared, and the ones that exceeded a threshold were used to form a sub-cluster.

### *4.2.i Training*

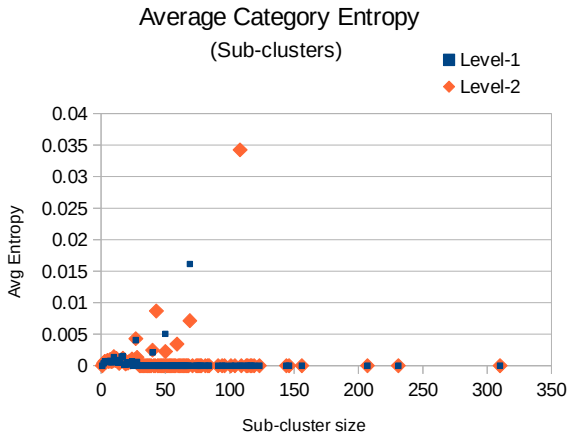
The key parameter is to determine the ratio of shingles match between two items. For this testing, sub-clusters from all super clusters were computed for a varying degree of shingle match, and the average *L2* category entropy across clusters was computed to choose minimum number of matching shingles such that the entropy is below a certain threshold. As the number of matching shingles decreases, the entropy increases and the cluster size distribution gets a fatter tail.



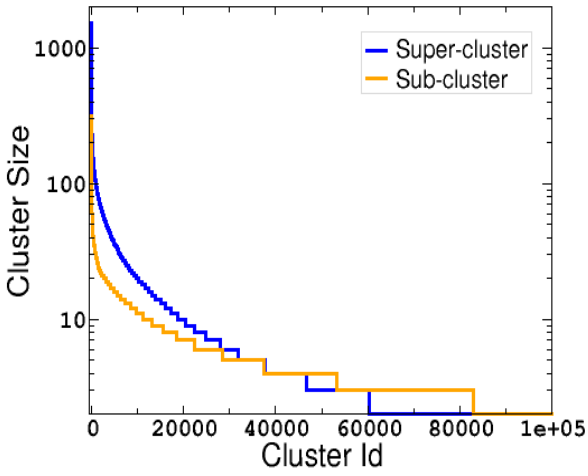




### *4.2.ii Testing*

Given the primary motivation behind sub-clusters, we decided to conservatively choose at least 6/8 to match. The testing approach was the same as for super-clusters, but from the graph below, it is clear that even the *L2* category entropy within each sub-cluster (on an average) is extremely low, which is a good measure of the quality of similarity produced.




*Average sub-cluster category entropy*




*Final cluster sizes (log scale)*

### 4.3 Future Work

For this project, clustering and categorization have been independent of each other. One of the key factors for accuracy of the classification are the unknown tokens present in the product title and description. One option is to increase the training data, or alternatively, to add related tokens from existing training set. Given that sub-clusters have near zero *L2* category entropy, the tokens within a cluster can be mined to reduce the *unknown token ratio* when required. Also, classifying a product into an L1 category first and then restricting L2 categories to only the subset of L1 category can also be explored.

## *5 References*

[PUU1] Puurula, Antti, “Combining Modifications to Multinomial Naive Bayes for Text Classification”, Springer Link, Lecture Notes in Computer Science Volume 7675, 2012, pp 114-125, http://www.cs.waikato.ac.nz/~asp12/publications/Puurula_12c.pdf

[KIB1] Kibriya, A.M., Frank, E., Pfahringer, B., Holmes, G.: Multinomial Naive Bayes for Text Categorization Revisited. In: Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence. AI'04, Berlin, Heidelberg, Springer-Verlag (2004) pp 488-499

[VIL1] Villaca, R.S., Paula, L.B., Pasquini, R. and Magalhaes, M.F., “Hamming DHT: Taming the Similarity Search” http://www.facom.ufu.br/~pasquini/artigos/ccnc_2013.pdf

[ULL1] Rajaramanhttp, Anand and Ullman, Jeffrey David, “Mining of Massive Datasets” http://i.stanford.edu/~ullman/mmds/ch3.pdf