Everyone Likes Shopping!
Multi-class Product Categorization for e-Commerce

Zornitsa Kozareva
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
zornitsa@kozareva.com

Abstract

Online shopping caters the needs of millions of users on a daily basis. To build an accurate system that can retrieve relevant products for a query like "*MB252 with travel bags*" one requires product and query categorization mechanisms, which classify the text as *Home&Garden>Kitchen&Dining>Kitchen Appliances>Blenders*. One of the biggest challenges in e-Commerce is that providers like Amazon, e-Bay, Google, Yahoo! and Walmart organize products into different product taxonomies making it hard and time-consuming for sellers to categorize goods for each shopping platform.

To address this challenge, we propose an automatic product categorization mechanism, which for a given product title assigns the correct product category from a taxonomy. We conducted an empirical evaluation on $445,408$ product titles and used a rich product taxonomy of 319 categories organized into 6 levels. We compared performance against multiple algorithms and found that the best performing system reaches .88 f-score.

1 Introduction and Related Work

Over the past decade, e-Commerce has rapidly grown enabling customers to purchase any product with a click of a button. A key component for the success of such online shopping platforms is their ability to quickly and accurately retrieve the desired products for the customers. To be able to do so, shopping platforms use taxonomies (Kanagal et al., 2012), which hierarchically organize products from general to more specific classes. Taxonomies support keyword search and guarantee consistency of the categorization of similar products, which further enables product recommendation (Ziegler et al., 2004; Weng et al., 2008) and duplicate removal.

Shopping platforms like Amazon, e-Bay, Google, Yahoo!, Walmart among others use different taxonomies to organize products making it hard and labor-intensive for sellers to categorize the products. Sometimes sellers are encouraged to find similar products to those they sell and adopt this category to their products. However, this mechanism leads to two main problems: (1) it takes a lot of time for a merchant to categorize items and (2) such taggings can be inconsistent since different sellers might categorize the same product differently. To solve these problems, ideally one would like to have an automated procedure, which can classify any product title into a product taxonomy. Such process will both alleviate human labor and further improve product categorization consistency in e-Commerce websites.

Recently, a lot of interest has been developed around the induction of taxonomies using hierarchal LDA models (Zhang et al., 2014) and the categorization of products using product descriptions (Chen and Warren, 2013). Despite these efforts, yet no study focuses on classifying products using only titles. The question we address in this paper is: *Given a product title and a product taxonomy, can we accurately identify the corresponding category (root-to-leaf path in the taxonomy) that the title belongs to?*

The main contributions of the paper are:

- We built multi-class classification algorithm that classifies product titles into 319 distinct classes organized in 6 levels.
- We conducted an empirical evaluation with $445,408$ product titles and reach .88 f-score.

- During the error analysis we found out that our algorithm predicted more specific and fine-grained categories compared to those provided by humans.

2 Product Categorization Task Definition

We define our task as:

Task Definition: Given a set of titles describing products and a product taxonomy of 319 nodes organized into 6 levels, the goal is to build a multi-class classifier, which can accurately predict the product category of a new unlabeled product title.

The algorithm takes as input a product title "*MB22B 22 piece with bonus travel/storage bag*" and returns as output the whole product category hierarchy "*Home and Garden >Kitchen&Dining>Kitchen Appliances>Blenders*" as illustrated in Figure 1.

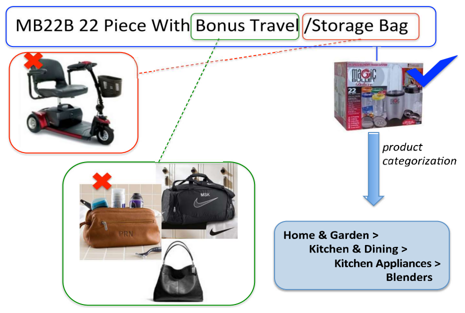

Figure 1: Example of Product Title Categorization.

3 Classification Methods

We model the product categorization task as classification problem, where for a given collection of labeled training examples P, the objective is to learn a classification function $f : p_i \rightarrow c_i$. Here, p_i is a product title and $c_i \in \{1, ..., K\}$ is its corresponding category (one of 319 product taxonomy classes).

We learn a linear classifier model f (parametrized by a weight vector $\mathbf{w}$) that minimizes the misclassification error on the training corpus P:

$$\min_{\mathbf{w}} \sum_{p_i \in P} \delta(c_i \neq f(\mathbf{w}, p_i)) + \lambda ||\mathbf{w}||_2^2$$

where, $\delta(.)$ is an indicator function which is 1 iff the prediction matches the true class and λ is a regularization parameter.

For our experiments, we used two multi-classification algorithms from the large scale machine learning toolkit Vowpal Wabbit (Beygelzimer et al., 2009): one-against-all (OAA) and error correction tournament (ECT). OAA reduces the K-way multi-classification problem into multiple binary classification tasks by iteratively classifying each product title for category K and comparing it against all other categories. ECT also reduces the problem to binary classification but employs a single-elimination tournament strategy to compare a set of K players and repeats this process for $O(\log K)$ rounds to determine the multi-class label.

4 Feature Modeling

Next we describe the set of features we used to train our model.

4.1 Lexical Information

N-grams are commonly used features in text classification. As a baseline system, we use unigram and bigram features.

4.2 Mutual Information Dictionary

Lexical features require very large amount of training data to produce accurate predictions. To generalize the categorization models, we use semantic dictionaries, which capture the presence of a term with a product category. Ideally, we would like to use existing dictionaries for each product category, however such information is not available. For instance, WordNet provides at most synonyms/hyponyms/hypernyms for a given category name, but it does not provide products, brand names and the meaning of abbreviations.

We decided to generate our own dictionaries, by taking all product titles and estimating the mutual information $MI(w, C_i) = log\frac{f(w,C_i)}{(f(w,*).f(*,C_i)}$ of every word w and product category C_i. For the dictionary, we keep all word-category pairs with MI above 5. During feature generation, for each title we estimate the percentage of words found with each category C_i according to our automatically generated dictionary. The dimensions of the feature vector is equal to the total number of categories. The size of the generated lexicon is $34,337$ word-category pairs.

4.3 LDA Topics

We also incorporate latent information associated with product titles using topic modeling techniques.

We learn latent topics corresponding to terms occurring in the titles using Latent Dirichlet Allocation (David Blei and Jordan, 2003). We capture the meaning of a title using the learned topic distribution. For our experimental setting, we use the MALLET (McCallum, 2002) implementation of LDA and build it in the following manner.

Method: Given a set of titles and descriptions D of products from different categories, find K latent topics. The generative story is modeled as follows:

```
for each product category $s_k$ where $k \in \{1, ..., K\}$ do
    Generate $\beta_{s_k}$ according to $Dir(\eta)$
end for
for each title $i$ in the corpus $D$ do
    Choose $\theta_i \sim Dir(\alpha)$
    for each word $w_{i,j}$ where $j \in \{1, ..., N_i\}$ do
        Choose a topic $z_{i,j} \sim Multinomial(\theta_i)$
        Choose a word $w_{i,j} \sim Multinomial(\beta_{z_{i,j}})$
    end for
end for
```

Inference: We perform inference on this model using collapsed Gibbs sampling, where each of the hidden sense variables $z_{i,j}$ are sampled conditioned on an assignment for all other variables, while integrating over all possible parameter settings (Griffiths and Steyvers, 2002). We set the hyperparameter η to the default value of 0.01 and α=50. During feature generation, we take all words in the title and estimate the percentage of words associated with each topic s_k. The topic-word mapping is constructed from the word distribution learnt for a given topic. The number of features is equal to the number of topics.

Figure 2 shows an example of the different topics associated with the word *bag* for different product titles.

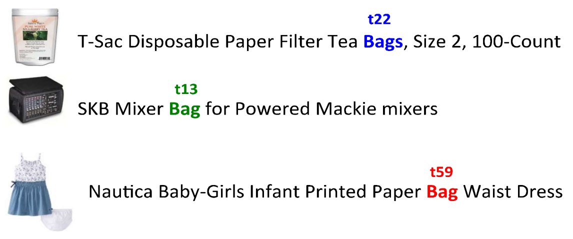
t22
T-Sac Disposable Paper Filter Tea Bags, Size 2, 100-Count
t13
SKB Mixer Bag for Powered Mackie mixers
t59
Nautica Baby-Girls Infant Printed Paper Bag Waist Dress

Figure 2: Learnt Topic Assignments for *bag*.

4.4 Neural Network Embeddings

While LDA allows us to capture the latent topics of the product titles, recent advances in unsupervised algorithms have demonstrated that deep neural network architectures can be effective in learning semantic representation of words and phrases from large unlabeled corpora.

To model the semantic representations of product titles, we learn embeddings over the corpus P using the technique of (Mikolov et al., 2013a; Mikolov et al., 2013b). We use a feedforward neural network architecture in which the training objective is to find word (vector) representations that are useful for predicting the current word in a product title based on the context. Formally, given a sequence of training words $w_1, w_2, ..., w_T$ the objective is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n, j\neq 0}\log p(w_t|w_{t+j})$$

where n is the size of the training context and $p(w_t|w_{t+j})$ predicts the current position w_t using the surrounding context words w_{t+j} and learned with hierarchical softmax algorithm.

Since word2vec provides embeddings only for words or at most two word phrases, to represent a product title p containing a sequence of M word tokens $(w_1, ..., w_M)$, we retrieve the embeddings of all words and take the average score.

$$\begin{aligned} p &= [e^1, ..., e^d] \\ where, e^i &= \frac{1}{M}\sum_{j=1}^{M} e^i_{w_j} \end{aligned}$$

Here, d is the embedding vector size, e^i and $e^i_{w_j}$ are the vector values at position i for the product p and word w_j in p, respectively.

To build the embeddings, we use a vector size of 200 and context of 5 consecutive words in our experiments. We then use the new vector representation $[e^1, ..., e^d]$ (d features per title) to train and test the machine learning model.

5 Data Description

To conduct our experimental studies, we have used and manually annotated product titles from Yahoo's shopping platform. For each title, we asked two annotators to provide the whole product category from the root to the leaf and used these annotations as a gold standard.

We split the data into a training set of $353,809$ examples and a test set of $91,599$ examples. Our product taxonomy consists of 6 hierarchical levels. Figure 3 shows the total number of categories per level. The highest density is at levels 3 and 4.

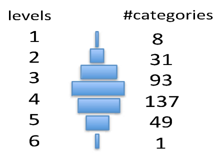

Figure 3: Product Taxonomy.

6 Experiments and Results

In this section, we describe the evaluation metric and the sets of experiments we have conducted.

6.1 Evaluation Metric

To evaluate the performance of the product categorization algorithms, we calculate *f-score* on the test set. The results are on exact match from top-to-leaf path of the gold and predicted categories.

6.2 Results

Table 1 shows the obtained results. For each feature we report the performance of the two machine learning algorithms one-against-all (OAA) and error correcting tournament (ECT).

features	OAA	ECT
unigram	.72	.63
unigram+bigram	.67	.58
MI Dictionary	.85	.77
LDA Dictionary	.79	.67
NN-Embeddings	**.88**	.80

Table 1: *Results on Product Categorization.*

The highest performance is achieved with the neural network embedding representation. Between the two classifiers one-against-all consistently achieved the highest scores for all different feature sets. We also studied various feature combinations, however embeddings reached the highest performance.

6.3 Error Analysis

We analyzed the produced outputs and noticed that sometimes the predicted category could be different from the gold one, but often the predicted category was semantically similar or more descriptive than those provided by humans. Figure 4 shows some examples of the errors we discovered.

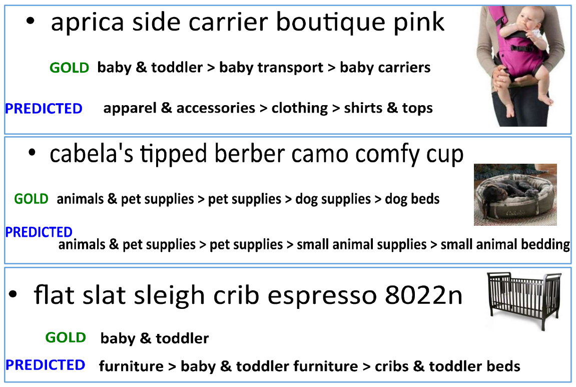
• aprica side carrier boutique pink
GOLD baby & toddler > baby transport > baby carriers
PREDICTED apparel & accessories > clothing > shirts & tops
• cabela's tipped berber camo comfy cup
GOLD animals & pet supplies > pet supplies > dog supplies > dog beds
PREDICTED animals & pet supplies > pet supplies > small animal supplies > small animal bedding
• flat slat sleigh crib espresso 8022n
GOLD baby & toddler
PREDICTED furniture > baby & toddler furniture > cribs & toddler beds

Figure 4: Examples of Categorized Products.

For instance, the title *cabela's tipped beer como comfy cup* was classified as *Small Animal Bedding*, while the gold standard category was *Dog Beds*. In our case we penalized such predictions, but still the two top level categories of *Animals* and *Pet Supplies* are similar. The major difference between the prediction and gold label is that the humans annotated *bed* as belonging to *Dog Beds*, while our algorithm predicted it as *Small Animal Bedding*. During manual inspection, we also noticed that often our classifier produces more descriptive categories compared to humans. For example, *flat slat sleigh crib espresso 8022n* had gold category *Baby & Toddler*, while our algorithm correctly identified the more descriptive category *Cribs and Toddler Beds*.

7 Conclusions

In this paper we have presented the first product categorization algorithm which operates on product title level. We classified products into a taxonomy of 319 categories organized into a 6 level taxonomy. We collected data for our experiments and conducted multiple empirical evaluations to study the effect of various features. Our experiments showed that neural network embeddings lead to the best performance reaching .88 f-score. We manually inspected the produced classification outputs and found that often the predicted categories are more specific and fine-grained compared to those provided by humans.

Acknowledgments

We would like to thank the anonymous reviewers for their useful feedback and suggestions.

References

Alina Beygelzimer, John Langford, and Pradeep Ravikumar. 2009. Error-correcting tournaments. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, ALT'09, pages 247–262.

Jianfu Chen and David Warren. 2013. Cost-sensitive learning for large-scale hierarchical classification. In *Proceedings of the 22Nd ACM International Conference on Conference on Information &#38; Knowledge Management*, CIKM '13, pages 1351–1360.

Andrew Ng David Blei and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Leaning*, 3:993–1022.

Thomas L Griffiths and Mark Steyvers. 2002. A probabilistic approach to semantic representation. In *Proceedings of the Twenty-Fourth Annual Conference of Cognitive Science Society*.

Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Jeff Yuan, and Lluis Garcia-Pueyo. 2012. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proc. VLDB Endow.*, 5(10):956–967, June.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. 2008. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *ICTAI (2)*, pages 113–120. IEEE Computer Society.

Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. 2014. Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 243–252.

Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. 2004. Taxonomy-driven computation of product recommendations. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 406–415.