SIG Proceedings Paper in LaTeX Format[*]

Extended Abstract[†]

ABSTRACT

Purchase item categorization is both a critical step in mining user online shopping behavior and a key enabler of many downstream analytical tasks such as user personalization and trend prediction. Training a categorization system, however, often involves large amount of parameters due to the large-scale nature of both the feature (ngram) and the output space (in-house taxonomy). Moreover, using supervised learning alone sometimes is not enough to achieve a satisfactory accuracy due to the high cost of obtaining labeled data and the noisy labeling process, e.g., crowd sourcing.

We therefore propose an end-to-end categorization system that requires only weak supervision by incorporating domain knowledge and business rules through generative modeling. We show that, by explicitly modeling the way domain knowledge generate labels, we are able to make use of the large amount of unlabeled data during training in a more cost-efficient way. During training, the system obtains latent representation of both the item and the category in a low-dimensional dense vector form. We show that this not only helps reduce the trainable parameters from O(|feature| x |output|) to O(|feature| + |output|) but also leads to high accuracy by capturing the syntactic and semantic feature / category relationships. We then show how to train the system efficiently by exploiting sparsity and multi-threading with momentum. Experimentally we apply the system to machine-generated receipts from millions of user accounts with hundreds of categories. The results show state-of-the-art categorization performance.

ACM Reference format:

. 1997. SIG Proceedings Paper in LaTeX Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK'97)*, 4 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

2 END TO END LEARNING FOR ITEM CATEGORIZATION

Category Taxonomy. Multi-class classification with a targeted taxonomy structure among categories is often considered as hierarchical classification. In this case the targets are often understood as sets of categories with closely related categories grouped in the same set. A standard approach then is to decompose the problem into a sequence of multi-class classifications in a recursive top-down manner: first discriminate the subsets of categories at the top level of the hierarchy before going down the next level to discern the categories (or sets of categories) in those subsets. This process is repeated until reaching the bottom level. There are however two major shortcomings about the approach: 1. *error propagation* where the mistakes made at the top-level will be carried over all the way to the leaf categories, and 2. *data scarcity and imbalance* where as the category become more specific the less training data there is for that category.

In this work we propose a more elegant and efficient approach with end-to-end learning from feature representation to multi-class classification. First of all, we simplify the problem by only considering the leaf categories in the taxonomy. In other words we flatten the hierarchy so that the problem becomes classic multi-classification. The trade-off is that while in the top-down approach the problem is decomposed into smaller subproblems each with only a few categories, now we consider 1000+ leaf categories all at once. We will see that this enables us to achieve a globally optimal solution in a sense that given an item a personalized total ranking can be learned among all leaf categories. This also present challenges to the problem due to the large scale output space.

Knowledge sharing (cite) has been shown to be important in multi-task learning scenarios like this where tasks are closely related and knowledge gained from one task can be used to facilitate the training of another one. We will also see that knowledge sharing is now possible by solving a unified single problem whereas before each subproblem is trained independently.

Negative Sampling With Log Loss. A standard approach in multi-class classification is to maximize the log probability estimated using softmax. This can become computationally expensive when the output space is large due to the need to compute the normalization factor. In this work we make use of a more efficient alternative based on negative sampling [2],

$$\mathcal{J}(\omega) = -\log \sigma(f_{x,c_x}(\omega)) - \sum_{i=1}^{k} E_{c_i \sim p_n(c)} \log \sigma(-f_{x,c_i}(\omega)) \quad (1)$$

Here, the logit function $f_{x,c}(\omega)$ is parameterized by ω and applying the sigmoid function $\sigma(f_{x,c}(\omega))$ approximates the probability of the item $x \in \mathcal{X}$ being in the category $c \in \mathcal{C}$. By minimizing (1) we increase the probability of x being in the ground truth category c_x while decreasing the probability of x in the wrong category sampled from the noise distribution $p_n(c)$. In this work we approximate it using n_c/n which is the number of category c in the training set divided by the total size of the training set. Note that (1) unlike Noise Contractive Estimation (cite) does not constitute an approximation to log probability of the full softmax. But it serves the purpose of classification where we only care about the top scores.

Joint Training of Item and Category Vector. Here we discuss the specific structure we use for the logit function $f_{x,c}(\omega)$. The trainable

[*]Produces the permission block, and copyright information
[†]The full version of the author's guide is available as `acmart.pdf` document

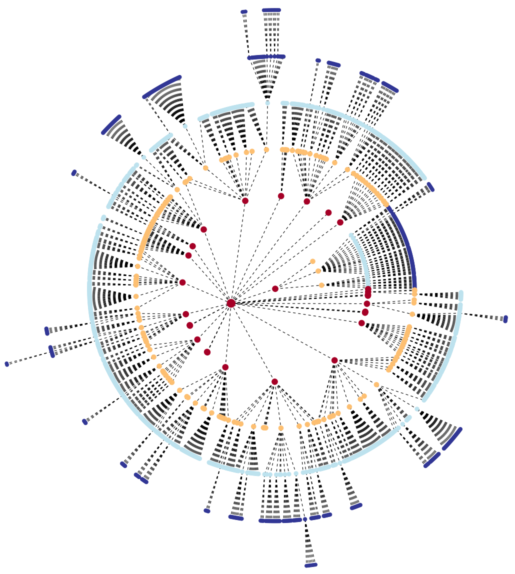


Figure 1: An illustration of in-house taxonomy.

parameters consist of the item matrix U and a category matrix V. Each row in the item matrix corresponds to a feature we extract from the item description, and each row in the category matrix represents a particular category. The logits for all categories can then be obtained as,

$$f_{x,C}(U,V) = x^T UV^T$$

Note that if letting $W = UV^T$ the above equation becomes $x^T W$ which is a standard form of multi-class classifier. By splitting W into two matrices U and V we are essentially imposing a low-rank structure onto the classifier. This greatly reduces the number of learning parameters and has been widely used in recommendation system, multi-task learning, etc, to improve generalization.

In feature extraction we include both unigram and n-gram. We use all unigram and apply hashing tricks [4] to n-gram, e.g., hashing each n-gram to a preset number of buckets. This has been shown to avoid overfitting and to improve system efficiency. In other words, if there are 1000 unique tokens and the bucket size is set to 500, then the dimension of x will be 1000 + 500 = 1500 where each entry is 0 or 1 denoting whether the corresponding token or n-gram is present in the item description or not. In practice the size of the vector x will be millions but very sparse. For efficient implementation we only store the indices of the non-zeros entries in x and the matrix-vector multiplication between x and U takes only a loop through the non-zeros entries which are only the number of tokens in the item description.

Now let us look at how we enable knowledge sharing in this context. Specifically, each task, or category in this case, corresponds to one row of the category matrix V, which captures the uniqueness of the task. The common knowledge shared by all task is reflected in the item matrix U. During training U will be updated by examples regardless of which categories they belong to, while the rows in V are only updated with the examples in the corresponding categories. In this way U can help improve the accuracy of the category even though there are only a few training data in that category. This addresses the aforementioned *data scarcity and imbalance* issues in the top-down approach.

Asynchronous SGD with Momentum. The embedding matrices U, V are trained using stochastic gradient descent with momentum and the updates are applied asynchronously in multiple threads. The major procedures are presented in Algorithm 1. The main update rules are described in Line 7-11. Here v_{c_x} denotes the vector representation of category c_x and $u_i, i \in nnz(x)$ denotes the feature vector corresponding to the ith non-zero entry of x. We will see that when the training dataset is highly sparse Algorithm 1 is the same as if the operations were executed in some sequential order of block coordinate descent. In fact, consider two training examples (x_1, c_1) and (x_2, c_2) such that $nnz(x_1) \cap nnz(x_2) = \emptyset$ and that $c_1 \neq c_2$. According to Algorithm 1 (x_1, c_1) and (x_2, c_2) will result in update on $\omega_1 = (v_1, U_{nnz(x_1)})$ and $\omega_2 = (v_2, U_{nnz(x_2)})$, respectively. Note that ω_1 and ω_2 does not overlap because the categories and the non-zeros entries of training examples do not overlap. The computations of their partial derivatives are also independent of each other. Hence the update procedures in Line 7-11 can happen simultaneously without the need for a lock. Finally we make use of momentum in the update (Line 7 and 10). The descent direction is obtained by a linear combination of current gradient and a running average of previous gradients. This has been shown to accelerate SGD in the relevant direction and dampen oscillations especially during navigating ravines [3].

Algorithm 1: Asynchronous SGD With Momentum.

```
1  Input: $\gamma \in (0,1), \eta > 0, T > 0$, the training set $\mathcal{D} \subset \mathcal{X} \times \mathcal{C}$
2  initialize randomly the trainable parameters $U, V$
3  initialize to zero the momentum parameters $\bar{U}, \bar{V}$
4  for each thread do // asynchronously
5      for $t = 1, 2, ..., T$ do // main iteration
6          sample $(x, c_x) \in \mathcal{D}$
7          $\bar{v}_{c_x} \leftarrow \gamma \bar{v}_{c_x} + \nabla_{v_{c_x}} \mathcal{J}(U,V)$
8          $v_{c_x} \leftarrow v_{c_x} - \eta \bar{v}_{c_x}$
9          for each $i \in nnz(x)$ do
10             $\bar{u}_i \leftarrow \gamma \bar{u}_i + \nabla_{u_i} \mathcal{J}(U,V)$
11             $u_i \leftarrow u_i - \eta \bar{u}_i$
12 Output: model parameters $U, V$
```

3 RULE INTEGRATION THROUGH GENERATIVE MODELING

weak supervision, in which training labels are noisy and may be from multiple, potentially overlapping sources To address this, we model the labeling functions as a generative process, which lets us automatically de-noise the resulting training set by learning the accuracies of the labeling functions along with their correlation structure. In turn, we use this model of the training set to optimize a stochastic version of the loss function of the discriminative model that we desire to train.

4 RESULTS

4.1 De-noising Training Set

Items with incorrect labels will affect the qualities of the categorization system if many are present in the dataset used for training. Those noisy items get introduced into the training set either due to conflict in the business rules or because of the labeling process which is often labor-intensive and error-prone. Here we use a de-noising approach based on active learning [1], by alternatingly updating the least 'confident' items given the classifier and the classifier given the training data. We observe that de-noising helps improve the accuracy of the system. The main procedures are described in Algorithm 2. An illustration of the intermediate results from one iteration of Algorithm 2 is shown in Figure 2a. The training data confidence scores for nine categories are presented. Outliers are shown as black dots, for example, the obvious one at the left bottom corner of Figure 2a. The central rectangle contains the scores between the first and the third quantile. A thin rectangle with high median score indicates a goodness of fit. In the experiments we find that the outliers are either labeled incorrectly or the classifier failing to learn the item sufficiently. For the former we either relabel or remove them from the training set. and for the latter we duplicate and add them back to the training set to increase their exposures to the training algorithm so the system learn them better.

Algorithm 2: Training Set De-noising Procedures.

```
Input: $\mathcal{X}_o = \emptyset, \mathcal{D} = \mathcal{X} \times C, U, V, \eta > 0$
repeat
    retrain $U, V$ from $\mathcal{D}$
    compute the logit matrix $L = XUV^T, L \in \mathbb{R}^{N \times |C|}$
    normalize $L$ such that all entries are between $(0, 1)$
    create the vector $l \in \mathbb{R}^N$ where $l_i = L_{ic_i}$
    set $\mathcal{X}_o$ to be empty
    for each $c \in C$ do
        rank $l_c \in \mathbb{R}^{N_c}$ in descending order
        compute the range of variation ($IQR$) from $l_c$
        collect points that are $\eta \times IQR$ below the first quantile
        add those points to $\mathcal{X}_o$
    update $\mathcal{D}$ by either removing or relabeling $x \in \mathcal{X}_o$
until $\mathcal{X}_o \neq \emptyset$;
Output: $U, V, \mathcal{D}$
```

4.2 Feature Representation And Selection

We demonstrate that our end-to-end system does not require feature engineering or preprocessing in the sense that it automatically extracts and selects important features from raw item descriptions. Furthermore, it does not require the removal of stop words or punctuations. On the contrary, it is observed that using raw text without preprocessing improves the performance. Table 1 shows the different features selected by different categories for prediction. The features are ranked by their relative importance to the given category vector. The importance scores are obtained by taking the inverse of the cosine distance between the category and the feature in the vector space. Notice that many world-famous brands are selected as important features, such as levi's for *jeans*, fitbit, jawbone for *wearable devices*, and ferrero, hersheys for *chocolates*, etc. This makes sense since those brand words might very well be the most frequent words in that category. However, notice also that no stop words make it to the list although they are quite frequent as well. Moreover, note that each category automatically selects its own set of features for prediction, although features are shared among all categories. For example, it is interesting to see that the system figures out on its own that, for jeans, 501 is more popular with men while women like 524. Finally Figure 2b demonstrates the discriminative property of the item vectors. In this experiment, we take a subset of items from eight categories and project them onto the 2-dimensional space. It can be seen from the figure that eight clusters are easily identified and well separated. It is interesting to notice that categories with similar meaning are close to each other, e.g., electronic devices (wearable and phones) on the top and apparel (men and women jeans) on the right side.

5 CONCLUSIONS

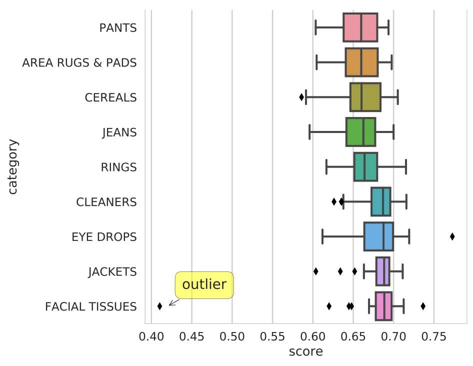
REFERENCES

[1] Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*.

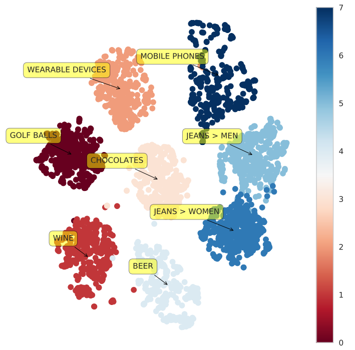
[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

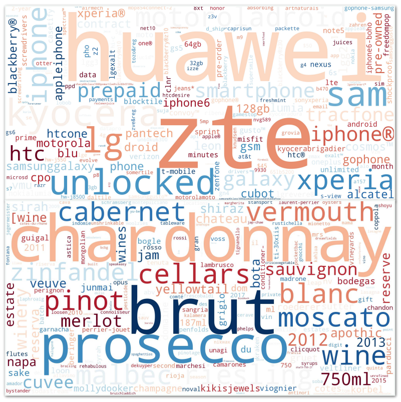
[4] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1113–1120.



(a) Noisy training sample detection.



(b) Visualizations of item vector clusters.



(c) Top feature clouds.

Figure 2: In (2a) it shows eight categories and their training sample 'confidence' scores distribution. Outliers are denoted by black dots. (2b) visualizes the dense 50-dimensional item vectors that are projected onto the 2-dimensional space using t-SNE with 50 perplexity and PCA initialization. (2c) is generated from top key words in categories *wine* (bottom) and *mobile phones* (above). Words with bigger sizes indicate stronger predicators for the category.

Table 1: Strongest Unigram Signals / Keywords for Category Prediction

JEANS > MEN	JEANS > WOMEN	WEARABLE DEVICES	BEER	WINE	CHOCOLATES	GOLF BALLS	MOBILE PHONES
brixton	jeans-	smartwatch	ipa	chardonnay	ferrero	titleist	zte
graduate	suki	misfit	ale	brut	m&m	srixon	huawei
normandie	super-skinny	fitbit	beer	prosecco	lindt	callaway	kyocera
fit-	rips	jawbone	budweiser	riesling	hersheys	maxfli	unlocked
rude	wedgie	forerunner	lagunitas	pinot	m&ms	volvik	sam
skinny-fit	straight-leg	vivofit	heineken	malbec	milka	taylormade	lg
levi's®	pajamajeans	tracker	coors	cellars	snickers	pinnacle	xperia
501	stevie	fitness	lager	wine	kisses	bridgestone	iphone
a&f	524™	nike+	modelo	vermouth	kinder	balls	motoroladroid
527	524	pebble	shiner	moscato	cadbury	dozen	smartphone
sartor	5pkt	garmin	peroni	blanc	ghirardelli	distance	iphone®
selvedge	jegg	withings	bottles	cabernet	brookside	wilson	galaxy
carpenter	rockstar	vivosmart	guinness	zinfandel	twix	golf	htc
stonewash	nouveau	activity	corona	sauvignon	butterfinger	cornmeal	prepaid
matchbox	seabreeze	smartband	sixpoint	750ml	godiva	flite	tracfone