SIG Proceedings Paper in LaTeX Format*

Extended Abstract†

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1]

ACM Reference format:
. 1997. SIG Proceedings Paper in LaTeX Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK'97),* 2 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. A more efficient alternative is to make use of negative sampling (cite),

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

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 $c \in C$. 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. Now 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 parameters consist of an 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,

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

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.

The features we use contain two parts, the unigram and n-gram. We use all unigram and hashes n-gram to a preset number of buckets to avoid overfitting and 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

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

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

WOODSTOCK'97, July 1997, El Paso, Texas USA

© 2016 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

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 all training examples no matter 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.

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 DATA PIPELINE

5 RESULTS

6 CONCLUSIONS

REFERENCES