

# Large Scale Product Categorization using Structured and Unstructured Attributes

Abhinandan Krishnan  
akrishnan@walmartlabs.com  
WalmartLabs  
USA

Abilash Amarthaluri  
aamarth@walmartlabs.com  
WalmartLabs  
USA

## ABSTRACT

Product categorization using text data for eCommerce is a very challenging extreme classification problem with several thousands of classes and several millions of products to classify. Even though multi-class text classification is a well studied problem both in academia and industry, most approaches either deal with treating product content as a single pile of text, or only consider a few product attributes for modelling purposes. Given the variety of products sold on popular eCommerce platforms, it is hard to consider all available product attributes as part of the modeling exercise, considering that products possess their own unique set of attributes based on category. In this paper, we compare hierarchical models to flat models and show that in specific cases, flat models perform better. We explore two Deep Learning based models that extract features from individual pieces of unstructured data from each product and then combine them to create a product signature. We also propose a novel idea of using structured attributes and their values together in an unstructured fashion along with convolutional filters such that the ordering of the attributes and the differing attributes by product categories no longer becomes a modelling challenge. This approach is also more robust to the presence of faulty product attribute names and values and can elegantly generalize to use both closed list and open list attributes.

## KEYWORDS

neural networks, text classification, extreme classification, eCommerce, structured datasets

### ACM Reference Format:

Abhinandan Krishnan and Abilash Amarthaluri. 2019. Large Scale Product Categorization using Structured and Unstructured Attributes. In *KDD '19: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 04–08, 2019, Anchorage, Alaska*. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## 1 INTRODUCTION

Categorizing products into a hierarchical taxonomy has become a central part of the organizational efforts of eCommerce companies. It is a critical step that acts as a precursor to multiple downstream

systems like search, facets etc. In the eCommerce context, it is posed as an extreme multi-class classification problem and is relatively well studied in both academia and industry.

There are several challenges that are unique to product categorization in the eCommerce domain that are not observed in traditional extreme classification challenges like ImageNet. Product catalogs are in constant flux with **old products being retired** and **new products being added** on a daily basis. Due to the dynamic nature of the product catalog, the hierarchical taxonomy against which they are categorized is also in constant flux although not at the same rate as the items in the catalog. Hence, we need to classify products with acceptable performance **against a non-stationary dataset** where both the sample space and the label space change at the same time. This also means that both the training and validation sets need to keep changing to reflect the latest snapshot of the distribution of the catalog. Acquiring labeled data for this changing catalog is also an extremely expensive process and hence intelligent sampling strategies need to be employed to reuse as many previously labeled examples as possible. With marketplace platforms such as Amazon, Walmart, eBay etc., there are new sellers and vendors being added everyday which results in a wide distribution of data quality levels for the products being setup. While **most products have some common attributes like title, description, image etc.**, every product also has a **unique set of structured attributes** describing it depending on the category the product belongs to. The total set of unique attributes in the catalog is in the tens of thousands ( $N$ ) while an individual product might only possess a few attributes ( $k \ll N$ ) that are relevant to it. In addition, the quality of product attributes widely varies by seller. Each product attribute also has its own value space that presents a modelling challenge. All these added complexities make product categorization an extremely challenging problem to tackle.

Next, in section 2, we briefly review related work in extreme multi-class classification and product categorization in particular. Section 3 outlines our preliminary approach to classification using hierarchical multi-class models, our move to a **flat classification scheme** using **two different flavors** of Deep Learning models, a **baseline architecture for structured attributes using word averaging** and finally an innovative way to use all available structured attributes in a product in an unstructured format along with **convolutional filters**. In section 4, the experimental setup is described, specifically the details about the dataset, preprocessing, training schedule, dictionaries, embeddings, models and their deployment. Finally in section 5, we provide some results comparing the different approaches we have experimented with.

Permission to make digital or hard copies of all or part 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 components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*KDD '19, August 04–08, 2019, Anchorage, Alaska*

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## 2 RELATED WORK

Many websites, especially in eCommerce, organize their product catalog into concept hierarchies or taxonomies. Some common examples are the Wordnet hierarchy, Google's Product Taxonomy, the Open Directory Project (ODP) etc. Extreme multi-class classification against such taxonomies has been worked on for a relatively long time and several approaches have been explored over the years. The two most common approaches that have been adopted are flat single-step classifiers and hierarchical multi-step classifiers.

Yu et. al. [2] explored several word level features in conjunction with linear classifiers like SVMs for classification. Kozareva [7] used several word features (n-grams, LDA, Word2Vec embeddings etc.) followed by linear classifiers. Ha et. al. [5] proposed multiple LSTM blocks, one for each unstructured or structured attribute followed by fully-connected layers for classification. Xia et. al. [12] proposed a variation of CNNs called Attention CNNs applied on Japanese product titles for classification.

Weigend et. al. [1] used a hierarchical classification scheme with one meta-classifier to determine the broad topic followed by individual topic level classifiers to distinguish nuances within each topic. They also observed that using neural networks in place of standard logistic regression resulted in improved performance. Shen et. al. [9] reformulated this into a two level classification problem ignoring the prior hierarchy and distributing the leaf nodes fairly evenly across top level categories. They used a kNN classifier at the top level followed by individual SVM classifiers for each second level node.

Yundi Li et. al. [13] used a Machine Translation approach to generate a root-to-leaf path in a product taxonomy. Gupta et. al. [4] trained path-wise, node-wise and depth-wise models (with respect to the taxonomy tree) and trained an ensemble model based on the outputs of these models. Zahavy et. al. [14] use decision level fusion networks for multi-modal classification using text and image inputs.

Flat classification models have more parameters per model but perform inference only once per product and hence have a lower latency. We can also batch products together to improve throughput. Hierarchical models on the other hand need to deal with products individually since each product may trace a different path in the taxonomy tree.

All of the above approaches use unstructured product attributes like title, description etc. to perform classification. Ha et. al. [5] used a limited set of structured attributes but used an independent LSTM block for each structured attribute which does not scale when each category of products contains different sets of attributes and there are thousands of product attributes overall that need to be considered.

## 3 OUR APPROACH

At Walmart, we experimented with multiple approaches to tackle this problem. Before delving deeper into all our experiments, we will present an overview of the information that a Walmart product contains. These are the attributes that will feed into the Machine Learning models.

### 3.1 What does a product contain?

A product is any commodity that may be sold by a seller. Within our catalog, every product contains a mix of unstructured and structured attributes which describe the product. Examples of unstructured attributes include product name, product short description, product long description, shelf description, synopsis etc. Examples of structured attributes include screen size, color, gender, fabric material, hard drive capacity etc. Structured attributes may have a closed list or an open list of values. For example, gender has a closed list of values while brand is an open list. Every product may also have multiple product images associated with it. These structured and unstructured attributes along with product images can all potentially be used to categorize the product.

### 3.2 Hierarchical Classification Approach

We first experimented with a hierarchical classification scheme similar to Weigend et. al. [1] using a bag-of-words hash feature on titles and descriptions followed by an entropy maximization based multinomial logistic regression classifier at each node in the taxonomy tree. The training data would vary based on the node the classifier was being trained for but the extracted features and classification scheme remained the same. This approach had several advantages and disadvantages.

- Advantages:
  - This architecture lent itself to parallelization at training time since each of the individual models could be trained in parallel.
  - Considering the rate at which the product catalog changes, we could focus specifically on retraining the parts of the model that needed attention rather than retrain the entire model at every iteration.
  - We could also target specific models for improvement without adversely affecting the performance of the rest of the models in the hierarchy.
- Disadvantages:
  - Top-1 predictions were very fast but top-k predictions were extremely slow since multiple paths (potentially all paths) along the hierarchy needed to be explored before returning the top-k predictions.
  - Large fraction of errors were made at the root level and the performance of the overall model was bounded above by the performance of the model at the root.

This hierarchical model is easy to deploy and could run efficiently without requiring GPUs. However, the taxonomy hierarchy itself had a significant impact on the performance of the model at the root. For example, product types Athletic Shoes and Dance Shoes appear under the category Sports & Outdoor while product types Casual & Dress Shoes and Safety Shoes & Boots appear under the category Clothing, Shoes & Accessories even though all 4 product types may be close to each other with respect to product content. Such confusions make it harder for the root node classifier to distinguish between categories at a high level for several types of products. After conducting an analysis, we found that this indeed contributes the largest fraction to the drop in accuracy.

taxonomy hierarchy itself  
had significant impact

{ closed list - gender.  
open list brand.

满足

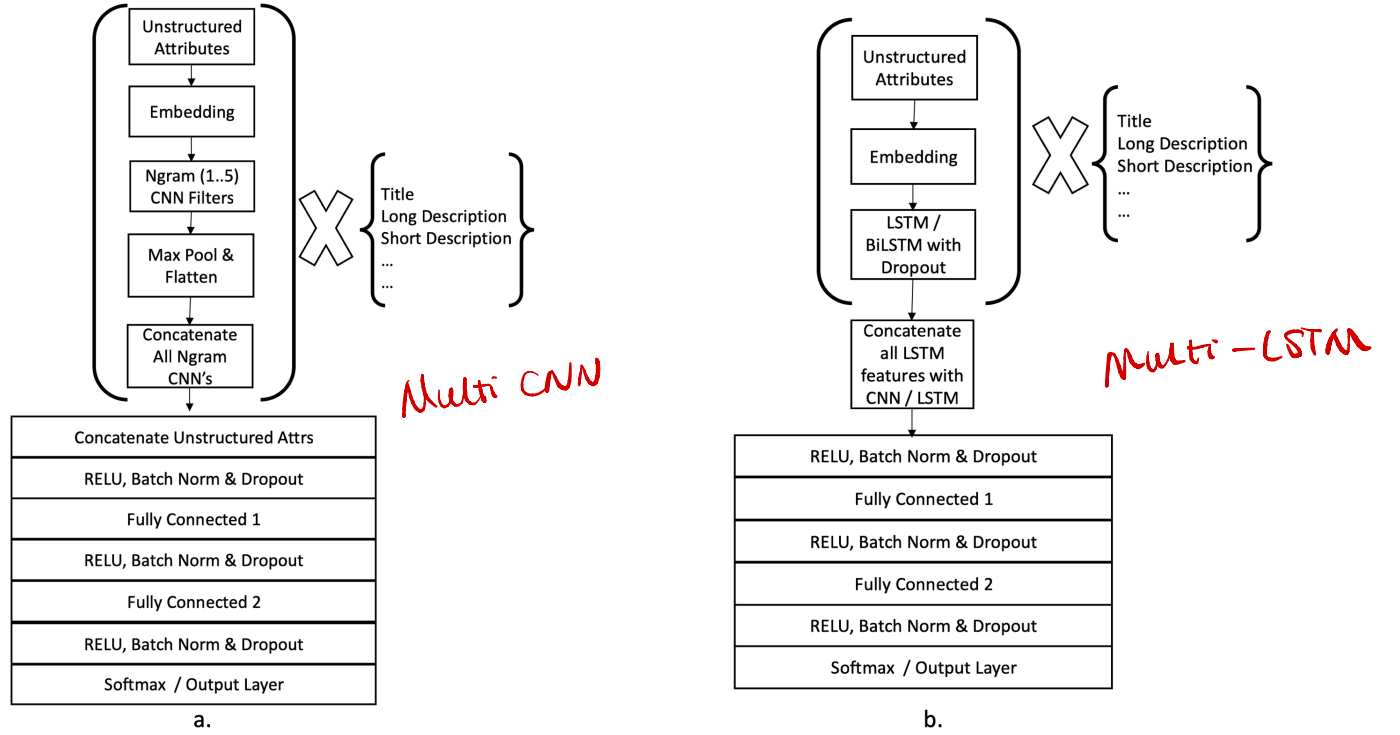


Figure 1: Model architectures used for unstructured attributes. Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.

### 3.3 Flat Classification Approach

Recently, we explored a single-step flat categorization approach using Deep Learning based models. These models seemed to benefit from the large amount of labeled data we had acquired and also demonstrated robustness to random label noise compared to the hierarchical model. In this paper, we describe a Multi-LSTM and a Multi-CNN based approach to perform product categorization. We also present a novel way to use structured attributes that can scale to any number of product attributes and any cardinality in the value space for a particular attribute. This method of using the structured attributes can be added on to any baseline model architecture and should provide a significant boost in classification performance. We observed that it achieves very similar improvements over both the Multi-LSTM and Multi-CNN architectures.

#### 3.3.1 Generating Word Dictionaries.

For each unstructured attribute, like *product name* or *product short description*, we go through the entire catalog and gather data for the attribute from every available product to build a word dictionary. We then trim this dictionary to retain a subset of words. The number of words we retain for each attribute depends on the original size of the dictionary and the variety of words found for that particular attribute. We use word embeddings with 200 dimensions for each attribute which are initialized randomly and updated over the course of training. We also experimented with Word2Vec and Glove embeddings with and without updates during

the course of training. However, using these embeddings degraded performance marginally and we also observed that the dictionaries associated with these pretrained embeddings did not capture a lot of very important tokens in our catalog. We also experimented with embeddings of size as low as 50 and found that these lower dimensional embeddings had comparable performance to embeddings with 200 dimensions.

#### 3.3.2 Multi-CNN Model.

In this approach, we use an independent set of convolutional filters of multiple lengths (1,2,3,4,5) on the word embeddings for each unstructured attribute, as shown in Figure 1a. 128 filters of each length are used and each filter is of size  $n \times 200$  where  $n$  represents the filter size. This approach is similar in motivation to Kim [6]. The multiple length convolutional filters in essence capture n-gram features from pieces of text. The activations generated by each set of convolutional filters are passed through a max pooling layer along the last dimension to retain one activation per filter. These activations for each attribute are then concatenated and passed through multiple fully connected layers followed by a softmax layer at the end for classification. The performance of this approach was very similar to that of the Bidirectional Multi-LSTM model and led to an improvement of almost 20% over the hierarchical model.

#### 3.3.3 Bidirectional Multi-LSTM Model.

In this approach, we use one Bidirectional LSTM layer on the embeddings for each unstructured attribute, as shown in Figure 1b. The

*n: filter size*  
*n gram*

Flat classification : Text classification / Maschin Translation.  
 otherwise: hierarchical classification : Tree-

Table 1: Example product

Attribute Name	Attribute Value
product_name	Rails Womens Plaid Spread Collar Button Down Top
product_short_description	This Rails Button Down Top is guaranteed authentic. It's crafted with 100% Rayon.
assembled_product_weight	0.5 Pounds
color	White
clothing_size_type	Regular
clothing_size_group	Women
maternity	N
age_demographic	Women
brand	Rails
fabric_material	Jersey
clothing_size	S
style_sleeve	Long Sleeves
actual_color	White Navy Sky
country_of_origin_assembly	CN
personalizable	N

activations generated by each LSTM layer can either be concatenated directly or put through another bidirectional LSTM following which we add multiple fully-connected layers and softmax at the end for classification. This approach is similar to the one used by Ha et. al. [5], however, we found that using a second level LSTM that takes in the activations of the first level of LSTMs as time steps performs slightly better than just concatenating the activations and using fully connected layers on top. Using this approach led to an improvement of almost 20% over the hierarchical model.

### 3.4 Using Structured Attributes

Every product has a small set of structured attributes which varies based on the category. Some attributes are common across categories while others are specific to a particular category. For example, **assembled\_product\_width** is an attribute that is relevant to both **Cell Phones** and **End Tables** even though these product types appear under completely different categories. In this case, the value of the attribute may have some useful information regarding the category or product type to which the product could potentially belong. On the other hand, an attribute like **diaper\_size** is only relevant to diaper related product types. In such cases, just the presence of the attribute is a strong indicator about the product type.

#### 3.4.1 Traditional Use of Structured Attributes.

Guo and Berkhahn [3] proposed Entity Embeddings for Categorical Variables for the Rossmann Store Sales Kaggle competition. This approach helps avoid feature sparsity and captures semantic relationships between the entities in a euclidean space. This is the latest state of the art method using structured attributes. However, one of the limitations of this method is that separate entity embeddings are needed for each categorical variable which quickly becomes unmanageable when the number of attributes grows to a few thousand. In addition, lots of attributes whose value spaces are open list are not necessarily categorical and hence may need other representations.

Embedding : { avoid feature sparsity  
 capture semantic relationship

#### 3.4.2 Building Attribute Word Dictionaries.

We build a common word dictionary across all attribute names and values unlike our approach with unstructured attributes where we build a separate dictionary for each attribute. This common dictionary is then trimmed down based on the total number of words in it. It is recommended to explicitly ensure that all the tokens present in the complete set of product attribute names are present in the dictionary after it has been trimmed down. This is accomplished by building a separate dictionary for tokens in the attribute names alone and then merging it with the joint dictionary for attribute names and values. This enables the model to extract richer features from these attribute name value pairs.

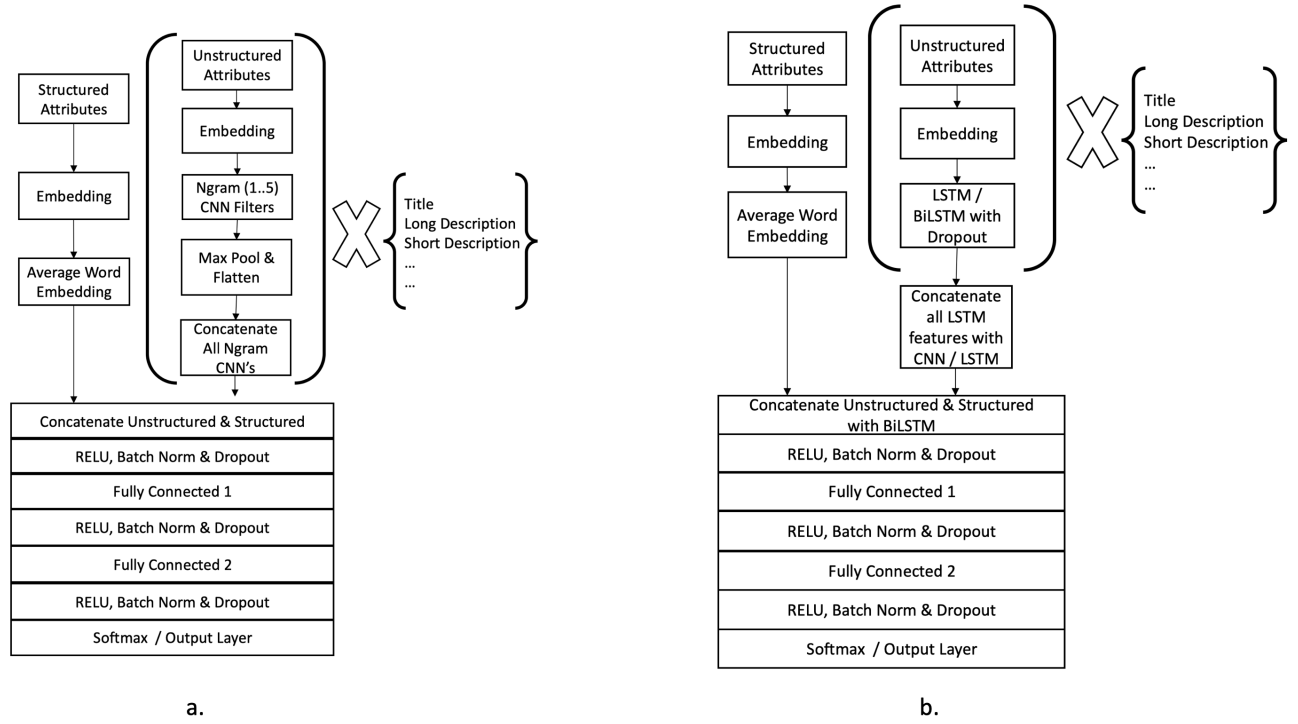
#### 3.4.3 Combining Structured Attributes.

Once the dictionary is built, we combine all the structured attributes associated with the product in an unstructured fashion. We use both attribute names and values to achieve the best performance. We observed that using just the attribute names improves performance marginally but using both names and values together provides the best results. Attribute names and values are broken down into natural language by stripping out underscores, dashes and other such special delimiters and then combined in the order below.

```
<attr_name> <attr_value> <separator> <attr_name>
<attr_value> <separator> ... <attr_name> <attr_value>
```

This is very similar in format to other unstructured attributes like **product\_name** or **product\_long\_description**. The custom separator token is added so that the convolutional filters have a marker depicting the end of one attribute and the beginning of the next attribute. In our experiments, we found that using the separator token improves classification accuracy over not using it. An example product is shown in Table 1. The combined structured attribute set for this product is shown below:





**Figure 2: Model architectures used for unstructured and structured attributes using averaged word embeddings . Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.**

```
assembled product weight 0.5 Pounds __sep__ color White
__sep__ clothing size type Regular __sep__ maternity N
__sep__ clothing size group Women __sep__ age demo-
graphic Women __sep__ brand Rails __sep__ fabric material
Jersey __sep__ clothing size S __sep__ style sleeve Long
Sleeves __sep__ actual color White Navy Sky __sep__ coun-
try of origin assembly CN __sep__ personalizable N.
```

#### 3.4.4 Structured Attribute Features using Word Averaging.

Wieting et. al. [11] showed that a simple **word averaging method performs well on sentiment classification**. We use the same method as a feature extractor and create a joint embedding for all structured attribute names and values in a given product. The joint embedding is simply the average of the word embeddings of the tokens found in the structured attribute names and values. This feature is then concatenated with the features extracted by the initial layers of the Multi-LSTM or the Multi-CNN models from the unstructured attributes. This is considered the baseline approach to incorporate structured attributes in our classification model. We observe that even this simple averaging of embeddings is an extremely useful feature and results in a lift in classification accuracy. The model architecture for this approach is shown in Figure 2.

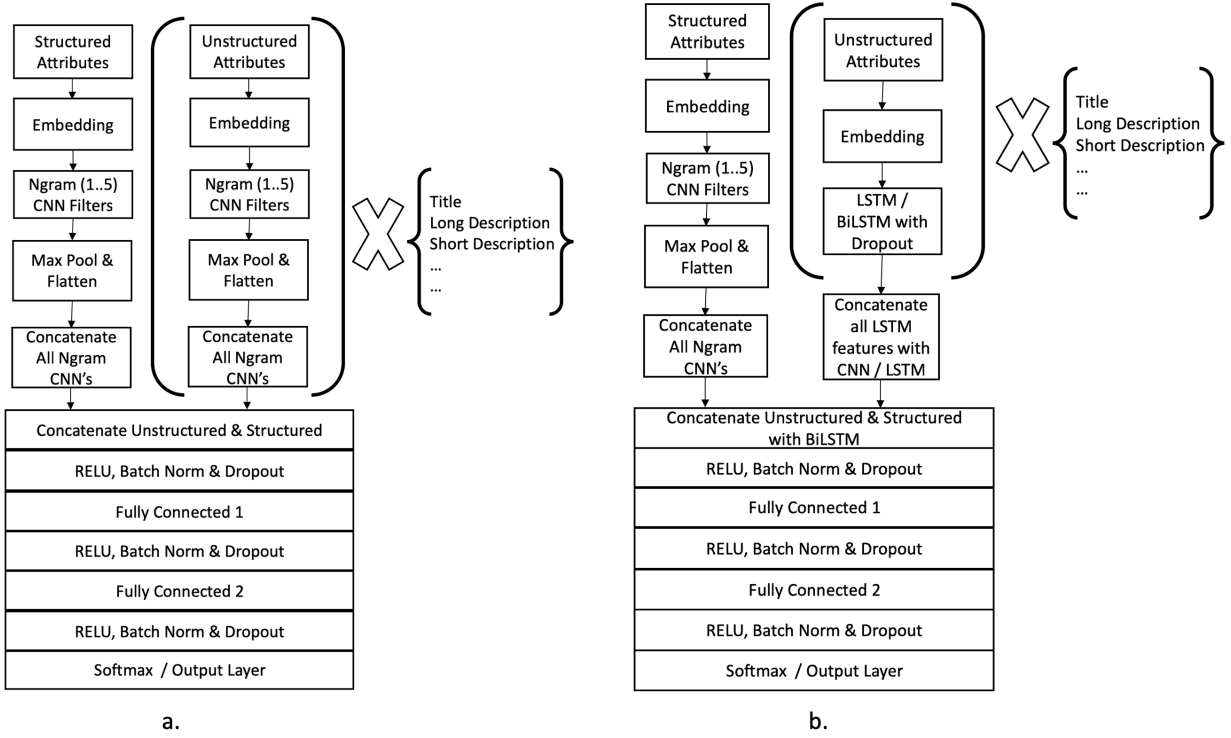
#### 3.4.5 Structured Attribute Features using Convolutional Filters.

We use the same model architecture as the Multi-CNN model mentioned above to extract features from the combined structured

attribute string. These features are concatenated with the features extracted by the Multi-LSTM or the Multi-CNN model. We use convolutional filters of multiple lengths (1,2,3,4,5) that capture features from the pairs of attribute names and values. There are 128 filters for each filter length. We originally experimented with one channel of convolutional filters for attribute names alone and another channel of convolutional filters for both attribute names and values. However, this did not improve performance over just using one channel for both attribute names and values. This is likely because the filters with smaller lengths typically capture features from the attribute names while the filters with larger lengths capture features from the attribute names and values together. Thus both types of features are being captured by the same channel itself. Figure 3 shows the updated model architectures with the block for structured attributes added.

#### 3.4.6 Advantages of Using Structured Attributes in an Unstructured Format.

There are several advantages to using structured attributes in an unstructured format along with convolutional filters as described above.



**Figure 3: Model architectures used for unstructured and structured attributes. Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.**

- Typically structured attributes are used as hand engineered features that are directly concatenated to the final fully connected layers. Our approach removes the need to hand engineer features for each individual attribute since the convolutional filters are able to capture interesting features relevant to the classification problem.
- With the current state of the art approaches for using structured attributes as explained in the previous section, there is an inherent sparsity in the feature set. Also, the number of parameters required for the proposed method increases linearly with the number of attributes and the representation format of each additional attribute.
- Since we break down the attribute names and values into natural language tokens, this approach should generalize well to new attributes and new values added to existing closed-list or open-list attributes.
- By adding the features extracted from structured attributes, we can get better representations for products in general and these representations can be used for a variety of other tasks.
- Using word embeddings to represent the tokens present in these attributes also helps capture semantic relationships between multiple attributes or between attributes and their values. Common representations can also be learned for entities identified in these attribute values which can be

shared with the other channels for unstructured attributes and potentially for other tasks.

However, if the number of structured attributes available is small and limited, traditional approaches may outperform this method.

## 4 EXPERIMENTAL SETUP

### 4.1 Data

Over the years, internal and external crowd sourcing has enabled us to collect large quantities of labeled data for product type categorization. Currently our dataset contains approximately 25 million labels, which is split into 3 subsets (80-10-10) to generate train, validation and test sets. These products represent approximately 6000 leaf product types in our taxonomy. We adopted a bootstrapping approach to collect the external crowdsourced data where we trained a model with the existing data and sent out suggestions to the crowd, who would provide us with feedback. While this approach has helped us collect massive quantities of data, it also has an inherent label bias since the crowd does not have an intimate knowledge of the Walmart taxonomy and is likely to pick a close-enough label if the right label is not presented in the suggestions provided. Our hierarchical model is especially sensitive to this kind of label noise since it uses traditional logistic regression based classifiers. The dataset is also very unevenly distributed for each product type. An example of this uneven distribution is shown in Figure 4. This is typical in an eCommerce catalog, where some product types, like

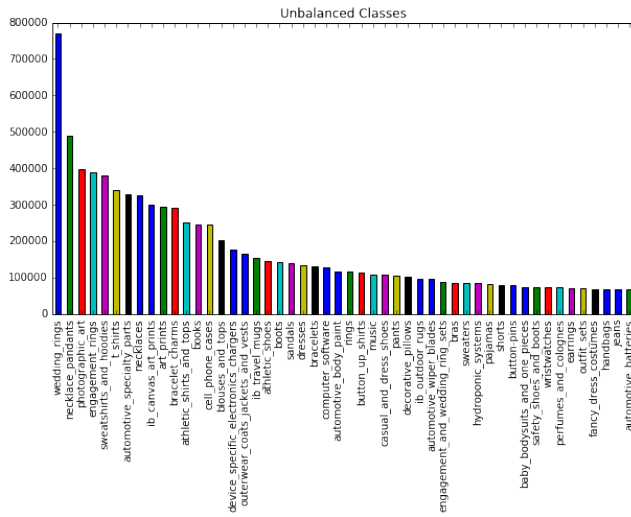


Figure 4: Unbalanced class distribution in the dataset

*T-Shirts* have several tens of millions of products while others like *Zithers* have less than ten products. We stratify low support classes by repeating samples until each class has at least 200 samples. We also ensure that we perform a stratified split into train, validation and test sets such that the sample distributions across these sets are similar.

## 4.2 Preprocessing

Standard whitespace tokenization is used to tokenize both unstructured and structured attributes. Tokenization and dictionary lookups are not precomputed but performed on the fly during training. **Preprocessing is an expensive step** however and does slow down training when done on the fly. In our case, since the product information keeps changing frequently, we typically don't store preprocessing results ahead of time before training.

## 4.3 Training Schedule

For both the Multi-LSTM and the Multi-CNN model, we adopt similar training procedures. We use **SGD with restarts to train these models** as described in Smith [10]. We start with one epoch and double the cosine annealing period with each cycle. The models with the lowest validation loss values are usually seen after around 5-6 epochs of training. Even though we also update the randomly initialized word embeddings at training time, the model converges to a loss value within 10% of the final loss value within one epoch after which it slowly improves and reaches its peak.

## 4.4 Dictionary and Embedding Details

As mentioned above, word **embeddings with 200 dimensions were used for each unstructured attribute**. The same embedding size was used for structured attributes too. The dictionary used for *product\_name* had 500K tokens, for *product\_description* had 1 million tokens and for *structured attributes* had 100K tokens.

## 4.5 Model Details and Training Time

The **hierarchical model** has approximately 1.5 billion parameters since it uses one model at each node in the taxonomy tree. The Multi-LSTM model has approximately 180 million trainable parameters while the Multi-CNN model has approximately 330 million trainable parameters. However, we have achieved similar results where both the models were compressed to around 65 million trainable parameters.

The hierarchical model is embarrassingly parallel, but uses CPUs to train and hence takes almost a day to train. The Multi-CNN model with structured attributes takes around 2.5 hours to complete one epoch while the Multi-LSTM model with structured attributes takes around 7 hours. Considering the training time, inference time and overall model performance on the test set, the Multi-CNN model wins over the other two model architectures.

## 4.6 Hardware and Software Details

- P100 GPUs used to train the CNN and LSTM models
- LSTM models were trained on Keras while the CNN models were trained on PyTorch
- Hierarchical Model trained using scikit-learn and parallelized using Spark

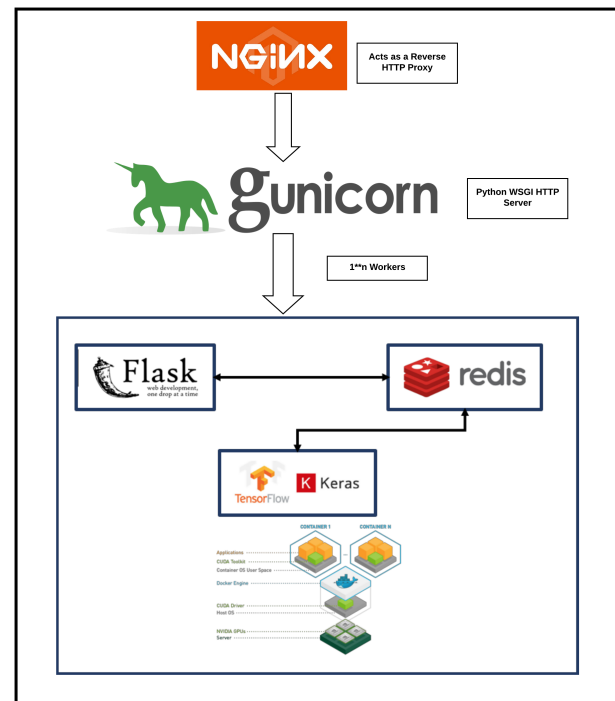


Figure 5: Deployment Architecture

## 4.7 Deployment

Our Machine Learning models are typically deployed as Micro Services to which other clients can send requests individually over HTTP. We have carefully designed our system with ideas borrowed

**Table 2: Model evaluation results**

Model architecture	Top-1 acc.	Top-2 acc.	Top-3 acc.
Hierarchical	70%	-	-
Multi-LSTM	89.9%	94.79%	96.42%
Multi-LSTM + Word Embedding Avg.	91.4%	95.95%	97.18%
Multi-LSTM + Struct. Attr.	92.28%	96.29%	97.38%
Multi-CNN	89.45%	94.93%	96.5%
Multi-CNN + Word Embedding Avg.	89.97%	95.31%	96.77%
Multi-CNN + Struct. Attr. w/o Separator	91.08%	95.87%	97.18%
Multi-CNN + Struct. Attr.	92.15%	96.36%	97.51%

from Rosebrock [8] to maximize overall throughput and minimize latency as shown in Figure 5. Since GPUs are suited for batch work loads, we microbatch multiple requests from different clients by adding a Redis buffering layer. These queued requests are sent as a minibatch to utilize the GPU effectively. A configurable poll interval (currently 0.3s) and a batch size of 1024 were chosen based on latency vs throughput trade-offs. In our load tests, we observed a throughput of 750 rps on a single P100 GPU with 6 CPU Cores.

**Table 3: Top-5 product types that benefit from using structured attributes with support  $\geq 100$  products**

Product Type	$\Delta$ f1 Score
Tank Tops	0.365
Chemistry Experiment Kits	0.317
Office Boxes	0.257
Outdoor Flags & Banners	0.249
Shape Sorting Toys	0.241

## 5 RESULTS

We have observed that attaching the structured attributes block (word embeddings followed by convolutional filter activations) to either model architecture improves overall accuracy by approximately 2.7% on the evaluation set as shown in Table 2. Considering that we have 6000 classes to classify against, this is a very significant increase in accuracy. This also shows that regardless of the model architecture, using this block for structured attributes improves overall accuracy. The model architectures we use, Multi-LSTM or Multi-CNN also lend themselves to variable sized inputs and hence can handle arbitrarily large unstructured attribute values. Upon performing an analysis of the top product types (with a support of over 100 products) in the evaluation set that benefit from using structured attributes, we found that **Tank Tops**, had the highest lift in f1 score. This is due to the presence of multiple attributes like **sleeve\_style** and **clothing\_top\_style** whose values indicate that the product is a **Tank Top**. Similarly, another product type that sees a significant lift in f1 score is **Office Boxes**. This is again because of the presence of the attribute **office\_box\_type** whose presence itself indicates that the product is an **Office Box**. From the above examples, we see that there are two useful features extracted from the structured attributes.

- The presence or absence of a particular attribute
- The value of a particular attribute

The convolutional filters are able to extract both these features efficiently from the structured text. Table 3 shows the top 5 product types ordered by the lift in f1 score using this approach.

## 6 CONCLUSIONS

Several insights, observations and conclusions were derived from this large-scale experiment some of which are mentioned below:

- **Training size:** For such extreme classification problems, we observe that having a large dataset with a good variety of samples to train significantly improves overall accuracy. Stratifying the dataset to improve representation for low support classes has also been shown to improve performance.
- **Choice of model:** In the presence of such large amounts of data, Deep Learning models significantly outperform traditional Machine Learning models. In this case, both Multi-LSTMs and Multi-CNNs perform equally well. However, the Multi-CNNs have the advantage of being more parallelizable and hence will be faster both at training and inference time.
- **Taxonomy structure:** Having the right taxonomy structure significantly improves the performance of classification models. Logical and mutually exclusive divisions of product types into categories helps improve classification performance significantly, especially using hierarchical models. Having more specific product types than very general ones is also recommended to aid better classification.
- **Word embeddings:** Using separate embedding spaces for each unstructured attribute seemed to perform better than using a common embedding space for all attributes. This is likely because the semantic relationships between words present in each attribute may vary.
- **Word embedding size:** Most of our experiments were performed with 200 dimensional word embeddings. However, embeddings as small as 50 dimensions also yielded similar results. Also, trimming the word dictionaries for each attribute seemed to act as an implicit regularizer and helped performance in some cases.
- **Using a separator:** While concatenating the structured attribute names and values together, it is beneficial to use a separator token between two attributes. This helps the model not to relate the values of one attribute with a different attribute name.



## REFERENCES

- [1] Erik D. Wiener, Andreas S. Weigend, and Jan O. Pederson. 1999. Exploiting Hierarchy in Text Categorization. *Information Retrieval* 1, 3 (October 1999), 193–216. <https://doi.org/10.1023/A:1009983522080>
- [2] Hsiang-fu Yu, Chia-hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih-jen Lin. 2012. *Product title classification versus text classification*. Technical Report.
- [3] Cheng Guo and Felix Berkhahn. 2016. Entity Embeddings of Categorical Variables. *arXiv e-prints*, Article arXiv:1604.06737 (April 2016), arXiv:1604.06737 pages. [arXiv:cs.LG/1604.06737](https://arxiv.org/abs/1604.06737)
- [4] Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product Classification in E-Commerce using Distributional Semantics. *CoRR abs/1606.06083* (2016). [arXiv:1606.06083](https://arxiv.org/abs/1606.06083) <http://arxiv.org/abs/1606.06083>
- [5] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. Large-Scale Item Categorization in e-Commerce Using Multiple Recurrent Neural Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 107–115. <https://doi.org/10.1145/2939672.2939678>
- [6] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1746–1751. <https://doi.org/10.3115/v1/D14-1181>
- [7] Zornitsa Kozareva. 2015. Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce. In *HLT-NAACL*.
- [8] Adrian Rosebrock. 2018. Deep learning in production with Keras, Redis, Flask, and Apache. <https://www.pyimagesearch.com/2018/02/05/deep-learning-production-keras-redis-flask-apache/>
- [9] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. *ACM International Conference Proceeding Series*, 595–604. <https://doi.org/10.1145/2396761.2396838>
- [10] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), 464–472.
- [11] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. *arXiv e-prints*, Article arXiv:1511.08198 (Nov. 2015), arXiv:1511.08198 pages. [arXiv:cs.CL/1511.08198](https://arxiv.org/abs/1511.08198)
- [12] Yandi Xia, Aaron Levine, Pradipto Das, Giuseppe Di Fabbrizio, Keiji Shinzato, and Ankur Datta. 2017. Large-Scale Categorization of Japanese Product Titles Using Neural Attention Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, 663–668. <http://aclweb.org/anthology/E17-2105>
- [13] Maggie Yundi Li, Stanley Kok, and Liling Tan. 2018. Don't Classify, Translate: Multi-Level E-Commerce Product Categorization Via Machine Translation. *arXiv e-prints*, Article arXiv:1812.05774 (Dec. 2018), arXiv:1812.05774 pages. [arXiv:cs.CL/1812.05774](https://arxiv.org/abs/1812.05774)
- [14] Tom Zahavy, Alessandro Magnani, Abhinandan Krishnan, and Shie Mannor. 2016. Is a picture worth a thousand words? A Deep Multi-Modal Fusion Architecture for Product Classification in e-commerce. *CoRR abs/1611.09534* (2016). [arXiv:1611.09534](https://arxiv.org/abs/1611.09534) <http://arxiv.org/abs/1611.09534>