Hierarchical Classification with Hierarchical Attention Networks*

Tengke Xiong
Shopify
tengke.xiong@shopify.com

Putra Manggala
Shopify
putra.manggala@shopify.com

ABSTRACT

Current state-of-the-art document classification methods based on attention mechanisms on document structures have been shown to perform well due to their ability to attend differentially to more and less important parts of the document when constructing the representation. While these models perform well on flat classification tasks, to our knowledge not much research has been done on hierarchical classification tasks.

In this paper, we propose a few variations of the Hierarchical Attention Network (HAN) that directly incorporate the pre-defined hierarchical structure of the output label space into the network structure, via the use of hierarchical output layers representing different levels of the output label's hierarchy. We present an evaluation of this model using the task of classifying Shopify app descriptions to a hierarchically-structured app category labels, and we demonstrate that by incorporating hierarchical structure of the labels into the design of the network, the model was able to outperform the baseline by a substantial margin. Additionally, we compare the classification performance of all the proposed variations.

KEYWORDS

hierarchical classification, hierarchical attention networks

1 INTRODUCTION

Many different domains contain hierarchy of information that can be stored, in a data structure sense, as hierarchies (or taxonomies, another term we will use interchangeably), which are a good way to help organize vast amounts of information. In e-commerce, the use of taxonomies to organize information are ubiquitous. One application is surrounding the task of classifying products to one or more product categories, represented as a taxonomy of product categories [2], which is fundamental to effectively search and organize the product listings in catalogs. Large e-commerce companies like Amazon, e-Bay, Pinterest, and Etsy list millions of products on their websites. Aside from classifying products into categories, another important application in e-commerce is the task of classifying apps to one or more app categories, also represented as taxonomies of app categories, for example see the Google Play Store [1] and the Shopify App Store [3]. This is fundamental for powering search capabilities for app users, so that they could find the apps they need based on certain functionality. App categories can also be used to help with downstream tasks, like app recommendation [6]. These tasks are instances of the hierarchical classification task. There are many different types of hierarchical classification tasks. In this paper, we focus on the task where the labels are tree-structured, pre-defined, and considered by the model all at once (as opposed to locally). For more details on these criteria, and different types of hierarchical classification tasks, see [15].

In the above tasks, hierarchy of information also exists within each document related to objects to be classified. Here, we use the term document to describe the textual metadata from items being classified into the categories, for example, product characteristics and descriptions of app functionalities. This hierarchy of information exist across various structural levels of granularity, such as chapters, sections, paragraphs, sentences, and words. This can be used by domain knowledge experts to understand the document in order to perform the above classification tasks. However, manual human classification is still a time-consuming and difficult task due to the number of documents, large size of taxonomy, noisy metadata, niche categories, continuously growing corpus and frequent document updates [7] [5] [8].

Document classification has long been a fundamental task in Natural Language Processing and also finds application in many industrial settings. Amidst all the various paradigms to solve document classification, machine learning techniques have found success due to their scalability (compared to manual classification) and adaptability (compared to automated rule-based approaches). In traditional machine learning approaches, documents are represented with sparse features like *n*-grams and classified using kernel methods [16] [9]. More recent approaches use neural-networks and have been shown to be quite effective [11] [12] [10]. In [17], it was shown that leveraging the hierarchical structure of the document by including two levels of attention mechanisms [4] at the word and sentence levels allows the model to pay more or less attention to individual words and sentences during representation learning. The differential utility of using attention mechanisms to model hierarchy inspired our work, as we build upon this work specifically to solve document classification tasks where the labels are hierarchical-structured.

2 THE PROPOSED ARCHITECTURE: HIERARCHICAL ATTENTION NETWORK WITH HIERARCHICAL OUTPUTS

Our architecture is largely based on the HAN model [17], which hierarchically model the structure of a document. However, it is different in following two ways:

*Produces the permission block, and copyright information

KDD'18 Deep Learning Day, August 2018, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.

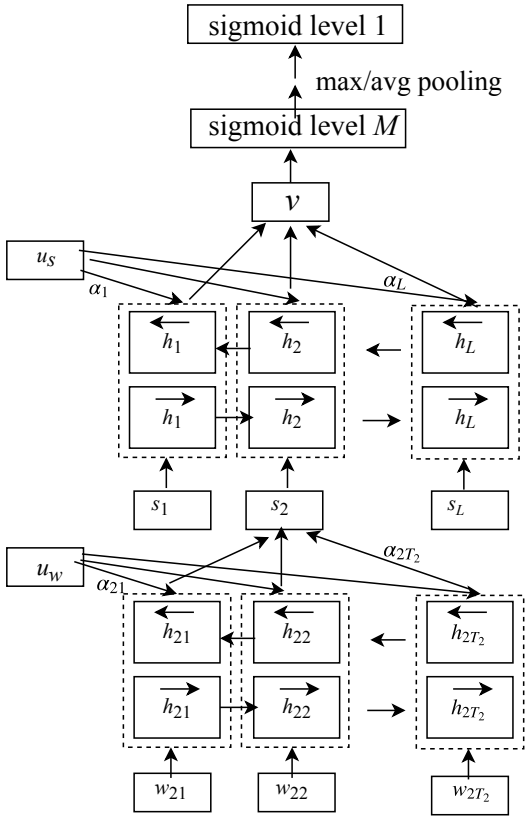

Figure 1: Hierarchical Attention Network with Hierarchical Outputs

- Our architecture has hierarchical output layers representing hierarchical taxonomy labels. Pre-defined tree structure of the taxonomy labels explicitly induces part of the network.
- Our architecture removes the one-layer Multilayer Perceptron (MLP) in the attention-building part of the networks. The attention weights are directly calculated as the similarity of context vector and the output hidden state vectors of the GRU.

The overall architecture of Hierarchical Attention Network with Hierarchical Outputs is shown in Figure 1.

Similar to in [17], we use a bidirectional GRU [4] to encode a sequence at both sentence level and word level. Given a document d with L sentences $(s_1, s_2, ..., s_L)$, each sentence s_i is a sequence of words $(w_{i1}, w_{i2}, ..., w_{iT_i})$. The document representation is obtained through two levels of sequence encoding, word and sentence.

In word sequence encoding, the outputs of the GRU network on sentence s_i are:

$$h_{it} = \overrightarrow{h_{it}} \oplus \overleftarrow{h_{it}},$$

where $\oplus$ is vector concatenation operation, and $\overrightarrow{h_{it}}$ and $\overleftarrow{h_{it}}$ are forward and backward hidden states at the t^{th} word, respectively.

In sentence sequence encoding, the outputs of the GRU network on a document d are:

$$h_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i}$$

Where $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are forward and backward hidden states at the i^{th} sentence respectively.

In RNNs with attention, the sequence representation is learned as a weighted average over the all the hidden states. The weight values are determined by the association between the hidden states and the context vector. To wit, the weight for the t^{th} ($t \in [1, T_i]$) hidden state in sentence s_i is:

$$\alpha_{it} = \frac{\exp(h_{it} u_w)}{\sum_t \exp(h_{it} u_w)}, \tag{1}$$

and the representation for the sequence s_i is:

$$s_i = \sum_t \alpha_{it} h_{it}. \tag{2}$$

The weight for the i^{th} ($i \in [1, L]$) hidden state in document d is:

$$\alpha_i = \frac{\exp(h_i u_s)}{\sum_i \exp(h_i u_s)}, \tag{3}$$

and the representation for the document d is:

$$v = \sum_i \alpha_i h_i \tag{4}$$

In our architecture, the attention weights are defined by the softmax of the dot products between context vector and the hidden state vectors, instead of projecting the hidden state vectors to a new feature space as in [17]. In our application domain and with hierarchical outputs, the experimental results demonstrate that the simplified architecture is more performant.

Our architecture has multiple output layers corresponding to different levels of taxonomy labels. The document representation vector is fully connected with the first output layer, which represents the bottom taxonomy level (level M). The sigmoid activation function is used in the output layer, as a document may have multiple labels for each taxonomy level in our overall output label hierarchy.

The first output layer is as follows:

$$P_M = sigmoid(w_c v + b_c),$$

where w_c and b_c are the weights and biases in a fully connected layer.

The next output layer is derived from the current output layer by either using maximum or average pooling. When maximum pooling is used, we have:

$$p^k_{m-1} = \max_j p^j_m,$$

while with average pooling:

$$p^k_{m-1} = \text{avg}_j p^j_m,$$

where the taxonomy node j at level m is a child of taxonomy node k at level $m-1$.

The loss function at level m is a sigmoid cross entropy loss, which is as follows:

$$J_m = -\sum_d \sum_{k=1}^{C_m} \left(y^k_m \log p^k_m + (1 - y^k_m) \log (1 - p^k_m) \right), \tag{5}$$

where C_m is the size of taxonomy at level m, which is also the dimension of the output layer that represents level m of the taxonomy.

The overall training loss:

$$J = \sum_{m=1}^{M} \omega_m J_m, \tag{6}$$

where ω_m is the weight on level m. The weights ω_m are hyperparameters of this model.

3 EXPERIMENTAL RESULTS

3.1 Dataset

We evaluate the effectiveness of our architecture by using the task of classifying Shopify app descriptions to a hierarchicaly-structured app category labels [3], where an app category reflects a (one of) functionality of an app. The dataset contains 1402 app descriptions, where we randomly choose 1100 apps for training and 302 apps for testing. Each app has three levels of labels, i.e., the distance between the top level label and bottom level label is equal to two. An app may have multiple labels on the same taxonomy label. The input documents to the model are app descriptions, for an example see Figure 2.

3.2 Model Variations

We evaluate different variations of our proposed architecture, which are as follows:

- Model variation with single output layer, where we set $\omega = [0, 0, 1]$. We use maximum pooling, and directly use hidden states of the GRU for attention. This variation is denoted as HAN-MAX-SIN-HID.
- Model variation with hierarchical output layer, where we set $\omega = [0.2, 0.3, 0.5]$. We use maximum pooling, and use hidden states of the GRU for attention. This variation is denoted as HAN-MAX-HIE-HID.
- Model variation with hierarchical output layer, where we set $\omega = [0.2, 0.3, 0.5]$. We use average pooling, and use hidden states of the GRU for attention. This variation is denoted as HAN-AVG-HIE-HID.
- Model variation with hierarchical output layer, where we set $\omega = [0.2, 0.3, 0.5]$. We use maximum pooling, and use an MLP layer to project hidden states of the GRU to a new feature space for attention. This variation is denoted as HAN-MAX-HIE-MLP.

3.3 Implementation Details

We use NLTK package [1] to split and tokenize documents into sentences. For word sequence learning, we use a static pretrained Glove word vectors [14] as sequence inputs to the GRU. The pretrained word vectors are 200-dimensional[2], trained using the Wikipedia and Gigaword data with 6B tokens and 400K words. The word vectors are kept as static in the training process, as in our setting our documents are short and the word occurrences are relatively few, and so tuning the word vectors during the training process has a high risk of overfitting.

We set the GRU hidden state dimension as 50 in sentence encoding and 100 in document encoding for all network variations. Therefore, the dimensionalities of sentence and document representations are 100 and 200 respectively due to a bidirectional network structure. We use 100 dimensions for document encoding because more information is required to capture the complex taxonomy structure in a document. We use the Adam optimizer [13] to train the model with learning rate equals to 1.e-3.

[1] www.nltk.org/

[2] glove.6B, https://nlp.stanford.edu/projects/glove/

3.4 Results and analysis

The performance is measured using a metric that is useful for a downstream application of this classification task. The quality is measured by the hit-rate at k (HR@k), and set $k = 3$. An app is considered as correctly classified at a particular taxonomy level if at least one of the top-3 predictions on that level is a ground truth label. The classification results are measured on the three levels separately. The overall metric HR@3 the percentage of apps that are correctly classified. The experimental results of the four HAN model variants are shown in Table 1.

From the experimental results we can see that HAN-MAX-HIE-HID performs best on all the three taxonomy levels. The hierarchical output network structure with weighted cross entropy loss functions takes advantage of prior taxonomy knowledge during model training, resulting in a considerable performance gain when compared to the variation with a single output layer at the bottom level (HAN-MAX-SIN-HID). It is interesting to see that even though HAN-MAX-SIN-HID directly learns the bottom level (the most granular level), it is inferior to HAN-MAX-HIE-HID not only at the top and middle level, but also at the bottom level (we observed 5.3% increase of HR@3). The results also show that maximum pooling outperforms average pooling in the output layers for prediction, and for our setting, omitting the use of an MLP layer to project the GRU hidden states for attention yields better performance.

4 CONCLUSION

In this paper, we have proposed a few architecture variations for hierarchical classification tasks based on Hierarchical Attention Networks. By incorporating the taxonomy prior knowledge in model training, some variations are more performant in the task of classifying to a tree-structured output label space than the baseline variation with a single output layer.

REFERENCES

[1] [n. d.]. Google Play Store. https://play.google.com/store. Accessed at: 2018-07-01.

[2] [n. d.]. Google Product Taxonomy. https://www.google.com/basepages/producttype/taxonomy.en-US.txt. Version date: 2015-02-19.

[3] [n. d.]. Shopify App Store. https://apps.shopify.com/. Accessed at: 2018-07-01.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *arXiv preprint arXiv:1409.0473*.

[5] Ali Cevahir and Park Avenue South. 2016. Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant. *Proceedings of the 26th International Conference on Computational Linguistics (COLING-16)* (2016), 525–535.

[6] Heng-Tze Cheng, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, and Wei Chai. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*. 7–10. https://doi.org/10.1145/2988450.2988454 arXiv:1606.07792

[7] Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbrizio, and Ankur Datta. 2017. Web-scale language-independent cataloging of noisy product listings for e-commerce. *Proceedings of the 15th Conference of the European Chapter of the*

Powerful page builder for landing, product, and blog pages. # TOP RATED page builder for Shopify stores! Read our REVIEWS!!\r\n\r\n# POWERFUL drag and drop page editing for Landing Pages, Product Pages, Blog Pages.\r\n\r\n# TRUSTED by SHOPIFY PLUS companies. See why they rely on our software to support their stores.\r\n\r\nNo coding required! Shogun has a seamless integration with Shopify, making it easy for you to manage your whole site within Shopify's application. Works with any theme!\r\n\r\n# You keep your pages and changes even if you uninstall! Give Shogun a FREE trial, and keep all of your work.\r\n\r\n# Check out our Library of Elements\r\nOur element library includes: Box with image and video Backgrounds, Columns, Image, Parallax, Video, Heading, Text, Button, Slider, Table, Tabs, Accordion, Instagram, Social, Countdown Timer, Separator, Forms, Map, and HTML.\r\n\r\nWe also have the Shopify Product Field Elements: Title, Image, Price, Variant, Quantity, and Add to Cart Button. These elements pull in your Shopify data automatically!\r\n\r\n# Create a Custom Home Page with Shogun\r\nYou can design a completely custom home page and replace your old homepage with the click of a button.\r\n\r\n# Easy to Use\r\nCreate new beautiful custom landing page, blog page, and product page layouts with drag & drop elements. Import and edit existing Shopify pages with the click of a button.\r\n\r\n# Mobile responsive\r\nShogun\u2019s pages are designed to look great on desktop, tablet, and mobile devices. Preview across devices in the editor.\r\n\r\n# Powerful\r\nLimitless styling options allows for completely custom landing page, blog page, and product page creation. Library of drag and drop elements includes unique features like Sliders, Tabs, Accordions, Instagram Feed and much more.\r\n\r\n# Developer friendly\r\nFor agencies, developers, or those who know how to code, custom drag and drop elements can be built by writing your own HTML / Liquid, CSS, and JavaScript.\r\n\r\n# We are here to answer any of your questions and support you!

Figure 2: App description example

Table 1: Hit rate results of the four model variations

Model variation	Top level HR@3	Middle level HR@3	Bottom level HR@3
HAN-MAX-SIN-HID	90.9%	75.6%	67.8%
HAN-MAX-HIE-HID	**93.4%**	**78.5%**	**73.1%**
HAN-AVG-HIE-HID	91.3%	74.4%	70.7%
HAN-MAX-HIE-MLP	93.4%	76.0%	70.0%

Association for Computational Linguistics, EACL 1 (2017), 3–7. http://arxiv.org/abs/1703.02344

[8] 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*. 107–115. https://doi.org/10.1145/2939672.2939678

[9] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98* 1398, 2 (1998), 137 – 142. https://doi.org/10.1007/BFb0026683

[10] Rie Johnson and Tong Zhang. 2015. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *Naacl* 2011 (2015), 103–112. arXiv:1412.1058 http://arxiv.org/abs/1412.1058{%}5Cnhttp://arxiv.org/abs/1412.1058v1

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 655–665. https://doi.org/10.3115/v1/P14-1062 arXiv:1404.2188v1

[12] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classificatio. *arxiv* (2014), 23–31. https://doi.org/10.1145/1599272.1599278 arXiv:arXiv:1408.5882v2

[13] Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations 2015*.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[15] Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 31–72.

[16] S Wang and C Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* July (2012), 90–94.

[17] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of NAACL-HLT 2016*.