Computing and Informatics, Vol. 39, 2020, 907–924, doi: 10.31577/cai_2020_5_907

HIERARCHICAL TEXT CLASSIFICATION USING CNNS WITH LOCAL APPROACHES

Milan Krendzelak, Frantisek Jakab

Technical University of Košice
Faculty of Electrical Engineering and Informatics
Department of Computers and Informatics
Letná 9, 040 01 Košice, Slovakia
e-mail: krendzelak.m@gmail.com, frantisek.jakab@tuke.sk

Abstract. In this paper, we discuss the application of convolutional neural networks (CNNs) for hierarchical text classification using local top-down approaches. We present experimental results implementing a local classification per node approach, a local classification per parent node approach, and a local classification per level approach. A 20Newsgroup hierarchical training dataset with more than 20 categories and three hierarchical levels was used to train the models. The experiments involved several variations of hyperparameters settings such as batch size, embedding size, and number of available examples from the training dataset, including two variation of CNN model text embedding such as static (stat) and random (rand). The results demonstrated that our proposed use of CNNs outperformed flat CNN baseline model and both the flat and hierarchical support vector machine (SVM) and logistic regression (LR) baseline models. In particular, hierarchical text classification with CNN-stat models using local per parent node and local per level approaches achieved compelling results and outperformed the former and latter state-of-the-art models. However, using CNN with local per node approach for hierarchical text classification underperformed and achieved worse results. Furthermore, we performed a detailed comparison between the proposed hierarchical local approaches with CNNs. The results indicated that the hierarchical local classification per level approach using the CNN model with static text embedding achieved the best results, surpassing the flat SVM and LR baseline models by 7 % and 13 %, surpassing the flat CNN baseline by 5 %, and surpassing the h-SVM and h-LR models by 5 % and 10 %, respectively.

Keywords: Hierarchical text classification, convolutional neural network, local top-down approach

Mathematics Subject Classification 2010: 68-U01

1 INTRODUCTION

Various methods exist for solving the hierarchical text classification (HTC) task, which is primarily based on how hierarchical relationships are utilized. Flat classification treats a flattened taxonomy as a set of unique classes that represent its hierarchy. Therefore, a binary classifier is usually trained for each class to discriminate it from the remaining classes. Although the flat classification approach is well known for its efficiency and simplicity when handling small-sized and well-balanced datasets, its performance suffers when the dimensions of the classes to be predicted not only increase but also become hierarchically interdependent. Due to the invocation of binary classifiers for all trained nodes, computation becomes time-consuming and costly [1].

The application of hierarchically organized categories into a taxonomy has become the most frequent method of organizing large quantities of data. For instance, enterprise customer care, product knowledge bases, online self-help centers, and e-learning systems. The most frequent strategy is to flatten a taxonomy so that all training datasets belong only to leaf classes. However, organizing the training datasets in this way does not mimic real-world examples very effectively. The connection between classes is lost because of the flattening process; thus, it is not possible to forecast the parent category of a new case [2].

However, in the context of HTC hierarchical relationships between parent and children elements, derived from a taxonomy of classes, should be considered either for training or predicting phases or both. In this case, the difference between different approaches to tackle the HTC task is the way the training dataset is leveraged. Currently, there are well-known hierarchical approaches known as local and global approaches. A hierarchical local approach is further divided into a local per node approach, local per parent node approach, and a local per level approach. Then the local approach involves splitting the hierarchical structure into several smaller structures for training local learning models; the global approach consumes the entire hierarchy of classes at once for training the global learning model [3].

2 RELATED WORK

There is a number of studies that focus on solving hierarchical text classification. For example, a well-known hierarchical top-down approach with level-based support vector machine models for text classification has been suggested by Sun and Lim [4]. Similarly, Sokolov et al. proposed a model for ontology term forecasting by explicitly simulating a construction hierarchy using kernel techniques for structured output [8]. Cerri et al. proposed an approach for hierarchical multi-label text classification that involves training a multi-layer perceptron for each level of the classification hierarchy [9]. The predictions generated by a neural network in a given level serve as inputs to the neural network responsible for the forecast in the following level. Their method was evaluated against several datasets with promising results.

Within the context of neural networks, Kurata et al. proposed a strategy for initializing neural networks' hidden output by considering multi-label co-occurrence. Their method treats a number of neurons in the final hidden layer as committed neurons for every pattern of tag co-occurrence [10]. In addition, there have been several important studies that proposed the inclusion of multi-label co-occurrence into loss functions, such as pairwise standing loss by Zhang and Zhou [11]. Furthermore, in more recent work, Nam et al. [12] reported that binary cross-entropy can outperform pairwise ranking reduction by minding rectified linear units (ReLUs) for nonlinearity.

3 LOCAL CLASSIFICATION APPROACHES

The hierarchical local classification approach deals with the local cross-section of hierarchically organized classes in order to consider information about parent-child and sibling relationships during the training phase. Based on different methods of applying the cross-section to local information extraction, the local classification approach is further divided into three subcategories.

3.1 Local Classifier per Node

The local classifier per node (LCN) approach dictates that a binary classifier ψ_n is learned for each node $n \in N$ except for the root node R in the hierarchy H, as illustrated in Figure 1. The dashed squares represent binary classifiers that are assembled into top–down manner execution.

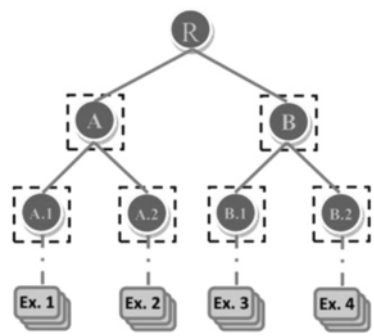

Figure 1. Local classifier per node

The training of a binary classifier at a node is performed by feeding the model with positive and negative examples. With respect to the LCN, all examples belonging to the n^{th} node and its descendants are considered positive training examples, while examples belonging to the n^{th} node siblings and their descendants are consid-

ered negative examples. The binary classifier is formulated as follows:

$$\sum_{i=1}^{N} \mathscr{L}(w_l, x(i), y(i)) + \lambda ||w_l||_2^2. \tag{1}$$

This classifier attempts to minimize the weight vectors for each label l, where $\lambda > 0$ is the penalty parameter, $\mathscr{L}$ denotes the loss function (e.g., hinge loss or logistic loss), and $||_2^2$ denotes the squared ℓ_2-norm. To predict an unknown test instance, the algorithm generally proceeds in a top-down manner, starting at the root and recursively selecting the best children until it reaches a terminal node that belongs to the set of leaf categories $\mathscr{L}$, which is the final predicted node.

The strategy described above is the one most commonly found in the literature. However, there exist additional strategies with different fine-tuning methods for annotating training data to differentiate among positive and negative examples.

3.2 Local Classifier per Parent Node

The local classifier per parent node (LCPN) approach states that a multi-class classifier is learned for each parent node $p \in N$ in the hierarchy $\mathscr{H}$, as illustrated in Figure 2. The dashed squares in the figure represent multi-class classifiers.

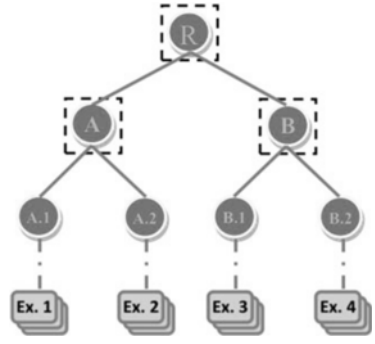

Figure 2. Local classifier per node

Like the LCN, the goal of the LCPN is to learn classifiers that can effectively discriminate between siblings. To train the classifier at each parent node N, we use the examples from its descendants, in which each of the children categories $C(N)$ of parent node N corresponds to different classes. The multi-class classifier is formulated as follows:

$$\text{minimize} \frac{1}{N} \sum_{i=1}^{N} \xi_i + \lambda \sum_{l=1}^{L} ||w_l||_2^2, \tag{2}$$

assuming that

$$w_{l_i}^T x(i) - w_l^T x(i) \geq 1 - \xi_i, \quad \forall l \in L - l_i, \forall i \in [1, 2, ..., N]$$

and

$$w_{l_i}^T x(i) - w_l^T x(i) >= 1 - \xi_i.$$

This classifier attempts to minimize the weight vectors, where $\lambda > 0$ is the penalty parameter, $\mathscr{L}$ denotes the loss function (e.g., hinge loss or logistic loss), ξ_i denotes the slack variables, and $||_2^2$ denotes the squared ℓ_2-norm.

3.3 Local Classifier per Level

In the local classifier per level (LCL) approach, a multi-class classifier is learned for every level in the hierarchy, as illustrated in Figure 3. To train the classifier at each level, examples from the nodes are used individually for each level along with its descendants. It should be noted that nodes at the same level do not overlap and correspond to distinct classes. Prediction is performed by selecting the best node at each level in the hierarchy.

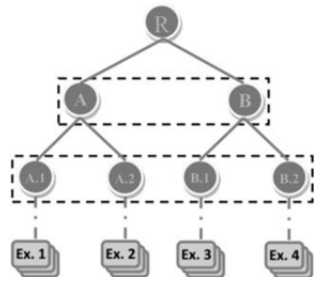

Figure 3. Local classifier per level

Because classifiers at each level make independent predictions, it is possible that this approach may result in vertical inconsistency in prediction. For this strategy to be useful, a post-processing measure is employed to solve inconsistent predictions, if necessary.

4 HIERARCHICAL PERFORMANCE EVALUATION

4.1 Flat Evaluation Metrics

As metrics, we use the standard micro-F_1 (μF_1) score and macro-F_1 (MF_1) score to evaluate the performance of various methods. To compute μF_1, we sum the category-specific true positives (TP_c), false positives (FP_c), and false negatives

(FN_c). Then, the definition shall be for different categories define micro-F_1 as follows:

$$\mu F_1 = \frac{2\ P\ R}{P + R} \tag{3}$$

where P is precision and R is recall, defined as follows:

$$P = \frac{\sum_{c \in \mathscr{L}} TP_c}{\sum_{c \in \mathscr{L}(TP_c + FP_c)}}, \tag{4}$$

$$R = \frac{\sum_{c \in \mathscr{L}} TP_c}{\sum_{c \in \mathscr{L}(TP_c + FN_c)}}. \tag{5}$$

The MF_1 score, which gives equal weight to all categories so that the average score is not skewed in favor of the larger categories, is defined as follows:

$$MF_1 = \frac{1}{|\mathscr{L}|} \sum_{cin\mathscr{L}} \frac{2P_cR_c}{P_c + R_c} \tag{6}$$

where $|\mathscr{L}|$ is the number of leaf categories, and P_c and R_c are defined as follows:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \tag{7}$$

$$R_c = \frac{TP_c}{TP_c + FN_c}. \tag{8}$$

4.2 Hierarchical Evaluation Metrics

With respect to HTC performance, hierarchical metrics should consider the hierarchical distance between the true class and predicted class. The principle is to penalize misclassification differently from flat metrics, which penalize each misclassified example equally. Generally, misclassifications that are closer to the actual class are penalized less than misclassifications which are further from it with respect to the hierarchy.

The hierarchical metrics include the hierarchical F_1 (hF_1) score, hierarchical precision P (hP), hierarchical recall R (hR), and tree-induced error (TE), which are defined as follows:

$$hF_1 = \frac{2\ hP\ hR}{hP + hR}, \tag{9}$$

$$TE = \frac{1}{N} \sum_{i=1}^{N} \delta(\hat{y}_i, y_i) \tag{10}$$

where hP and hR are defined as follows:

$$hP = \frac{\sum_{i=1}^{N} |A(\hat{y}_i) \cap A(y_i)|}{\sum_{i=1}^{N} |A(\hat{y}_i)|}, \tag{11}$$

$$hR = \frac{\sum_{i=1}^{N} |A(\hat{y}_i) \cap A(y_i)|}{\sum_{i=1}^{N} |A(y_i)|}. \tag{12}$$

Here, $A(\hat{y}_i)$ and $A(y_i)$ are the set of ancestors of the predicted and true labels, respectively, including the class itself but not the root node. $\delta(\hat{y}_i, y_i)$ represents the length of the undirected path between categories $\hat{y}_i$ and y_i in the tree.

5 TOP-DOWN HIERARCHICAL ENSEMBLE

A top-down ensemble of prediction evaluation is one of the most efficient approaches for solving the HTC task with binary classifiers, either implemented as CNN, SVN or LR models [8]. The principle of this approach is to recursively evaluate a prediction from the top of the hierarchy down to the leaf, traversing each node on the path, as illustrated in Algorithm 1. At each step, the node with the highest prediction score is selected. This process repeats recursively until the last leaf node is reached, which corresponds to a certain class and is usually located at the bottom.

Result: Repeat recursively until last leaf node is reached
initialization n := Root;
while $n \notin L$ **do**
  $n := \arg\max_{q \in C(n)} f_q(x)$;
end
return n;

Algorithm 1: Local top-down approach

Top-down methods are popular for large-scale problems due to their computational advantages when only a subset of classes in the appropriate path is considered in the prediction phase. In addition, these methods minimize problems related to prediction inconsistencies because the best child node is selected at each level of the path. Top-down methods have been successfully utilized to resolve HTC problems either for learning or training and prediction phases.

A major disadvantage of top-down technique that results in poor classification performance is error propagation, namely, the compounding of errors from misclassifications at higher levels that cannot be corrected at the next lower levels. This problem can be relieved to a certain extent by shifting the hierarchy to temper the level of deformation. It is important to note that this top-down technique is applied only for binary classifiers.

6 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) have been adopted from the field of computer vision, in which they have been shown to provide state-of-the-art results with default baseline hyperparameter settings.

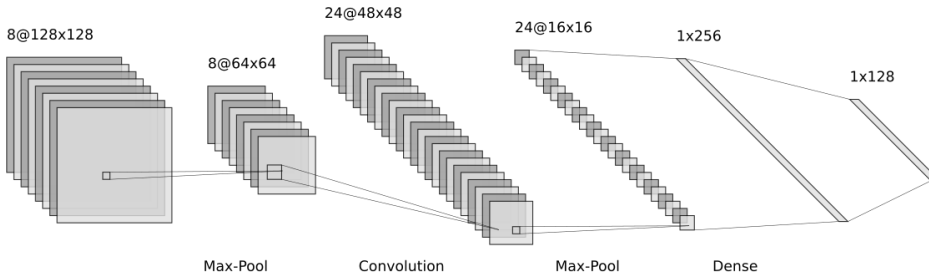

Figure 4. Convolution neural network (CNN)

Let $x_i \in R^k$ be the k-dimensional word vector relevant to the i^{th} word in a sentence. A sentence of length n (padded if required) is represented as follows:

$$x_{1:n} = x_1 \oplus x_2 \oplus \dots x_n \tag{13}$$

where $\oplus$ is the concatenation operator. Furthermore, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, \dots x_{i+j}$. Applying a filter $w \in R^h k$ in strides defines a convolution operation, which is applied to a selected window of h terms to compute a new feature. Therefore, feature c_i is produced for each window of words $x_{i:i+h-1}$ as follows:

$$c_i = f(w\ x_{i:i+h-1} + b). \tag{14}$$

Let $b \in R$ be a bias and f a non-linear function, for example, a hyperbolic tangent or relu function. Then, a filter is used to stride through each possible window in the given sentence $\{x_{1:h}, x_{2:h+1}, \dots x_{n-h+1:n}\}$ to create a feature map, such as

$$c = [c_1, c_2, c_3, \dots c_{n-h+1}] \tag{15}$$

where $c \in R^{n-h+1}$. Thereafter, a max-overtime pooling operation is applied over the set of feature maps to select a maximum value $C = \max\{c\}$ as the feature. The general principle is to be able to capture the most important set of features for all feature maps.

The above process describes how one feature is extracted from one filter. However, the model uses multiple filters, typically with varying window sizes, to extract multiple features. Therefore, these features form next to the last layer, which is directly followed by a fully connected softmax layer whose output is the probability distribution over the labels.

6.1 Convolution

A 1D convolution is an operation between a vector of weights m, where $m \in R^m$, and a vector of input sequences s, where $s \in R^s$. Vector m is the filter of the convolution. Let s be an input sentence and $s_i \in R$ be a single feature value associated with the i^{th} word in a sentence. Then, a 1D convolution produces the dot product of vector m with each n-gram in the sentence s to obtain another sequence c as follows:

$$c_j = m^T S_{j-m+1:j}. \tag{16}$$

Equation (16) describes two possible types of convolution – narrow and wide – depending on the range of index j. The narrow type of convolution requires that $s \geq m$, and yields a sequence $c \in R^{s-m+1}$ with j ranging from m to s. The wide type of convolution does not constrain m or s, and yields sequence $c \in R^{s+m-1}$, where index j ranges from 1 to $s+m-1$. Values s_i outside of the range are considered to be zero, where $i < 1$ or $i > s$.

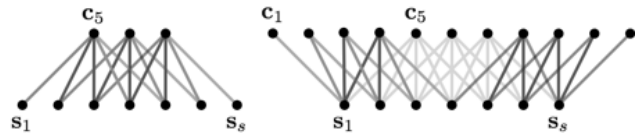

Figure 5. Narrow and wide convolutional layers

The result of a narrow convolution is a subsequence of the results of the wide convolution. The two types of one-dimensional convolutions are illustrated in Figure 5. The trained weights in filter m correspond to a linguistic feature that learns to recognize a particular class of n-grams. These n-grams have size $n \leq m$, where m is the width of the filter. Applying the weights m in a wide convolution has several advantages over applying them in a narrow convolution. A wide convolution ensures that all weights in the filter reach the entire sentence, including words at margins and paddings.

6.2 Feature Selection

One of the advantages of a CNN over other neural networks is that it is designed to automatically extract features from the given text corpora. It does so by applying convolutional layers in a specific, predefined manner as described in Section 6.1. These extracted feature maps are more efficient and less error-prone than manually constructed ones. In addition, they ensure a high level of accuracy in capturing the most important details for given examples of text data.

In general, convolution layers can be considered a feature extractor whose output is fed into a fully connected layer for the purpose of simple decision-making, such as classification or ranking. Because a CNN creates local features for each word in

a sentence, it is possible to combine or stack features to produce a global feature vector. Several aspects of a CNN's ability to create and extract text features are as follows:

- A CNN internally creates features that can be extracted and used as input for other custom-defined internal layers or external models.
- A CNN automatically creates features that do not rely on a hand-crafted process. It adapts well to the specifics of a training dataset in a supervised manner.
- A hierarchy of local features is considered during feature creation; therefore, the CNN captures the context effectively.

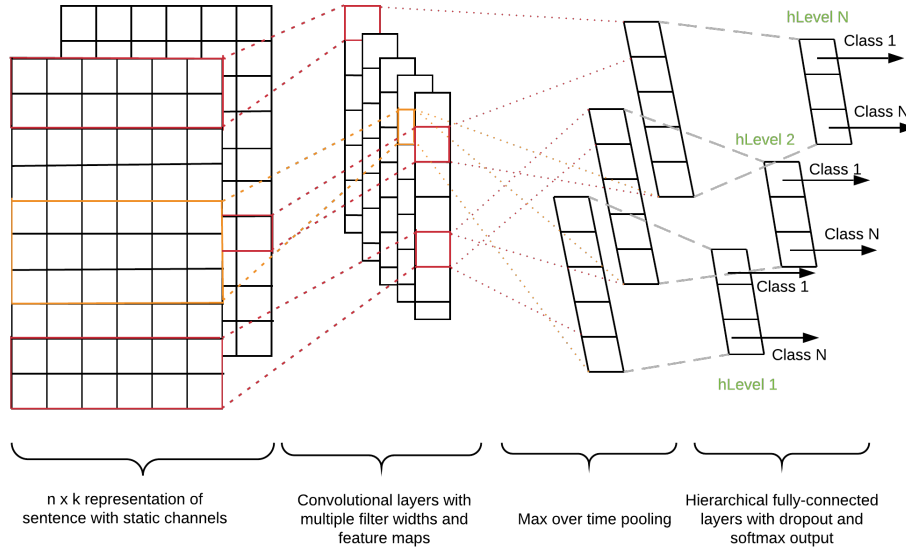

Figure 6. Conceptual diagram of experimental CNN for local per level classification

7 EXPERIMENTAL SETUP

To provide a comprehensive assessment of the use of CNNs for hierarchical text classification using local approaches, we conducted three independent experiments and observed, measured, and compared the outcomes.

It should be noted that the following constrains were taken into consideration and applied throughout the experimentation:

- The training dataset outlined in Figure 7 was used; thus, the results could be compared with each other and with previously reported applications of h-SVM and h-LR models using identical local approaches operated in a top-down fashion.

- The same variation of the CNN baseline model was used; thus, the outcome did not depend on the model specifics, but rather on the effectiveness of the local approach as a strategy.
- For all experiments in this study, we used 300-dimensional word vectors trained by Mikolov et al. on roughly 100 billion words from Google News.

7.1 Training Dataset

The 20Newsgroup dataset contained an average of 20 000 e-news items from more than 20 different categories hierarchically ordered with three hierarchical levels. The total training dataset contained 11 314 examples.

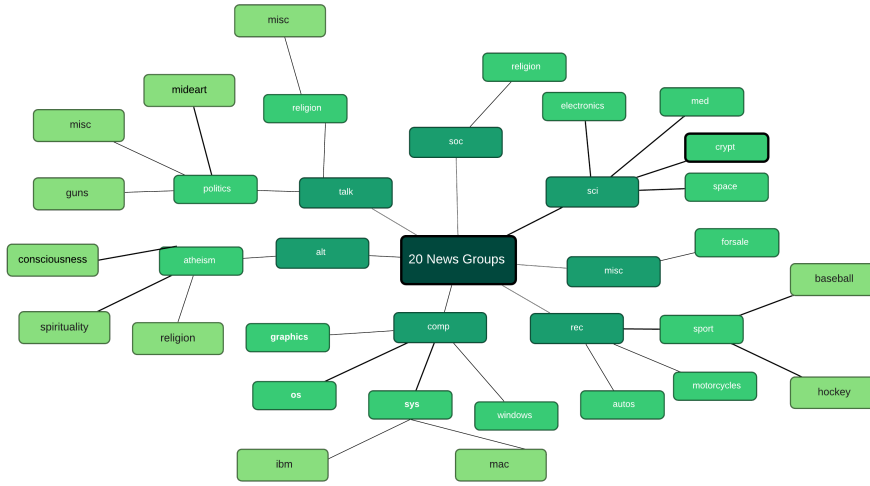

Figure 7. 20Newsgroup training dataset hierarchy

The category size corresponded to the number of available examples per category, as outlined in Table 1. The training data was preprocessed, aggregated for each hierarchy level, and counted for the number of available examples per category.

7.2 CNN Variants

One difference among the many recent studies on word-based CNNs for text classification is the choice of using pretrained or end-to-end learned word representations. In our experiments, we used two variants of the CNN baseline model that differed in the initialization of embedding.

- CNN-rand which learns the text embedding from scratch during the training.
- CNN-stat which is initialized with pretrained word2vec.

7.3 Hyperparameters and training

- Optimizer. For all experiments, we trained our model's parameters with the Adam optimizer initialized with a learning rate of 0.001.
- Batch sizes. We experimented with different batch sizes, such as 64, 128, and 192, as we wished to observe the impact of different values on model performance.
- Embedding sizes. The default embedding size was 128. However, to obtain an improved understanding of how this hyperparameter value affects performance, we experimented with additional settings, such as 256 and 300.
- Number of filters and their sizes. The presented CNN model contained three convolutional layers, each with three different filters. Each filter had a size of 3, 4, and 5.

8 EXPERIMENTS WITH LOCAL APPROACHES

8.1 LCN Experiment

The hierarchical LCN approach consists of binary classifiers, and each classifier is built independently for each node. In this case, all categories listed in Table 1 are considered nodes. For each node, we trained one CNN classifier. In total, we had 27 CNN classifiers corresponding to 27 categories.

Category	Size	Category	Size
comp	2 936	alt-atheism	480
comp.graphics	584	soc-religion	599
comp.os	591	sci	2 373
comp.windows	593	sci.crypt	595
comp.sys	1 168	sci.electronics	591
comp.sys.ibm	590	sci.med	594
comp.sys.mac	578	sci.space	593
rec	2 389	talk	1 952
rec.autos	594	talk.religion-misc	377
rec.motorcycles	598	talk.politics	1 575
rec.sport	1 197	talk.politics.guns	546
rec.sport.baseball	597	talk.politics.mideast	564
rec.sport.hockey	600	talk.politics.misc	465
misc-forsale	585	TOTAL	11 314

Table 1. 20Newsgroup text analysis of training dataset

To perform prediction, top-down sequential evaluation was performed, and at each node, a binary decision was made regarding which classifier to execute next in the evaluation chain. The training dataset for each node was carefully manually crafted in such a way that all examples belonging to the n^{th} node and its descendants

were considered positive training examples and examples belonging to the siblings of the n^{th} node and their descendants were considered negative examples.

8.2 LCPN Experiment

The hierarchical LCPN was implemented as a multi-class classifier for each parent node in the given taxonomy. For our experiment, the nodes that were considered are listed in Table 2.

Parent Node	Size	Classes
ROOT	11 314	comp, rec, sci, talk, misc-forsale, alt-atheism, soc-religion
comp	2 936	comp.graphics, comp.os, comp.windows, comp.sys
comp.sys	1 168	comp.sys.ibm, comp.sys.mac
rec	2 389	rec.autos, rec.motocycles, rec.sport
rec.sport	1 197	rec.sport.baseball, rec.sport.hockey
sci	2 373	sci.crypt, sci.electronics, sci.med, sci.space
talk	1 952	talk.religion-misc, talk.politics
talk.politics	1 575	talk.politics.guns, talk.politics.mideast, talk.politics.misc

Table 2. Training data for LCPN

We trained eight independent multi-class CNN classifiers for each parent node, including the ROOT node. Each classifier learned from the subset of training data created in such a way that only examples belonging to the parent nodes and their descendants child nodes were selected as positive examples. We did not feed negative examples into the model because neural networks are usually trained with positive examples.

8.3 LCL Experiment

The hierarchical LCL was implemented as a multi-class classifier for every level in the given taxonomy. We thus had three levels, as listed in Table 3. Each level contained only nodes belonging to a certain hierarchical level.

We trained each of the three multi-class CNN classifiers for every level. A prediction evaluation was performed in a top-down fashion starting from the first level and iterating through the remainder of the levels. Training of the level-based classifier was performed by feeding the model with examples from the descendant nodes belonging to their level-based parents.

The conceptual diagram of CNN model with LCL approach is listed in Figure 6. It can be observed that this model implements only 3 multi-class classifiers, each classifier represents exactly one of the hierarchical levels as listed in Table 3 in such way that there is no overlap among categories and different levels.

The final dense layer in CNN baseline contains a single node for each target class in the model. However, in order to represent LCL approach using CNN, the baseline model dense layer is modified and implemented 3 fully-connected layers.

Level	Size	Category
1	11 314	alt-atheism, comp, rec, misc-forsale, soc-religion-christian, sci, talk
2	9 650	comp.graphics, comp.os, comp.sys, comp.windows, rec.autos, rec.motorcycles, rec.sport, sci.crypt, sci.electronics, sci.med, sci.space, talk.religion-misc, talk.politics
3	3 940	comp.sys.ibm, comp.sys.mac, rec.sport.baseball, rec.sport.hockey, talk.politics.guns, talk.politics.mideast, talk.politics.misc

Table 3. LCL training datset

9 EVALUATION OF EXPERIMENTS

To determine the effects of using CNN models in hierarchical text classification, we performed multiple tests to experiment with a different set of hyperparameters. We conducted three major experiments with the proposed hierarchical local approaches and compared our results with available flat LR and SVM baseline models and state-of-the-art models, such as h-LR and h-SVM. One of our goals was to demonstrate that CNNs can be successfully used for solving hierarchical text classification problems in a more effective and simplified manner than existing methods.

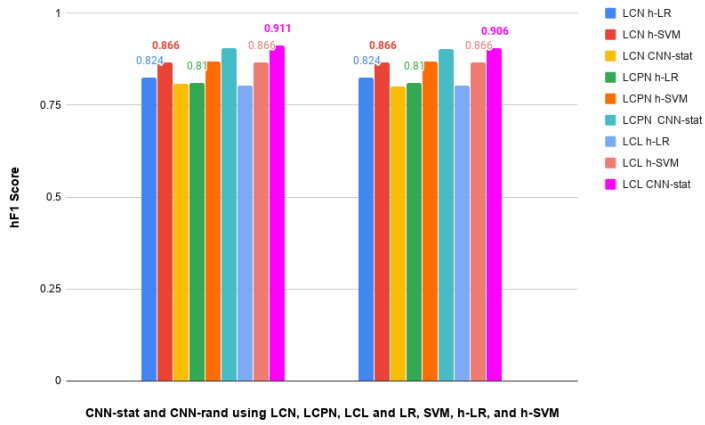

Figure 8. Benchmark of experimental CNN-rand and CNN-stat using LCN, LCPN and LCL with h-LR and h-SVM using LCN, LCPN and LCL

Of the proposed methods, we determined that the LCN hierarchical approach was the most complex. We observed that different variations of examples with different dataset sizes had a direct impact on the accuracy of the binary classifier. It thus requires additional effort during the pre-processing phase to prepare positive

and negative training examples arbitrarily selected from the dataset. Moreover, top-down prediction requires the evaluation of the final prediction to be proceeded in a node-chained manner, emulating the path of a hierarchical taxonomy. There is no mechanism available in the baseline model to mitigate the error propagated from the previous node prediction.

Our empirical observations of the LCPN approach indicated that it was less complex than the previous LCN approach mentioned above. The LCPN requires the construction of only eight multi-class classifiers whose predictions are chained in a top-down fashion to perform hierarchical prediction. Training a multi-class neural network is quite different than training a binary classifier. We observed that for this approach, less effort was required during the pre-processing of the training examples required to train a model with hierarchically categorical classes. This was due to the fact that a CNN is capable of consuming raw training data and does not require positive and negative samples.

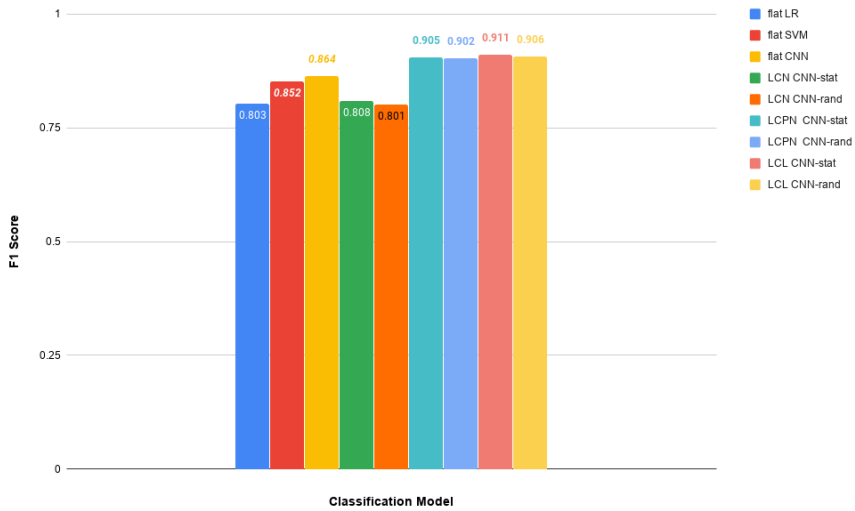

Figure 9. Benchmark of experimental CNN-stat and CNN-rand using LCN, LCPN and LCL with flat LR, SVM, and CNN baseline models

We found that the best results were achieved by using the LCL approach implemented by CNN-stat model with a hF_1 score of 0.911 trained with a dataset of size 10 500. For the CNN-rand model using LCL approach, the best results were achieved with a hF_1 score of 0.906. However, we observed that the latter model, CNN-rand, required more training data to tune its performance and was unable to outperform CNN-stat.

10 CONCLUSION

In this study, we proposed a novel application of a CNN for solving the hierarchical text classification problem using hierarchical local classification approaches. We demonstrated that hierarchical local approaches with CNN models achieved results superior to those of the flat LR and SVM baseline, which results were reported by Song et al. [14]. Moreover, additionally, experimental results achieved by proposed use of CNNs surpassed flat CNN baseline model by 5 %, which results were reported by Prakhya et al. [15].

The results confirmed that the CNN-stat LCL approach achieved the best results among the tested local approaches, furthermore, outperforming flat SVM baseline model by 7 % and flat LR baseline model by 13 %. In regards to h-SVM and h-LR models, these were outperformed by CNN-stat LCL approach by 5 % and h-LR by 10 %, as can be observed in Table 9. Moreover, the CNN-stat LCL approach required only 3 multi-class CNN classifiers, compared to 27 binary classifiers required by LCN approach and 8 multi-class classifiers required by LCL approach.

Additionally, we observed that the CNN-stat model, which has the text embedding layer initialized with pre-trained word2vec, outperformed the CNN-rand model most of the time, except only observed once. This is due primarily to the fact that pre-trained text embedding has a more precise and comprehensive representation of words than a randomly initialized embedding layer learned during the training phase.

REFERENCES

[1] Zimek, A.—Buchwald, F.—Frank, E.—Kramer, S.: A Study of Hierarchical and Flat Classification of Proteins. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 7, 2010, No. 3, pp. 563–571, doi: 10.1109/tcbb.2008.104.

[2] Babbar, R.—Partalas, I.—Gaussier, E.—Amini, M. R.: On Flat Versus Hierarchical Classification in Large-Scale Taxonomies. In: Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 26 (NIPS 2013), Vol. 2, 2013, pp. 1824–1832.

[3] Krendzelak, M.—Jakab, F.: Approach for Hierarchical Global All-In Classification with Application of Convolutional Neural Networks. 2018 16$^{\text{th}}$ International Conference on Emerging eLearning Technologies and Applications (ICETA 2018), 2018, pp. 317–322, doi: 10.1109/iceta.2018.8572074.

[4] Sun, A.—Lim, E.—Ng, W.: Performance Measurement Framework for Hierarchical Text Classification. Journal of the American Society for Information Science and Technology, Vol. 54, 2003, No. 11, pp. 1014–1028, doi: 10.1002/asi.10298.

[5] Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, Vol. 7, 2006, No. 1, pp. 1–30.

[6] Dumais, S.—Chen, H.: Hierarchical Classification of Web Content. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00), 2000, pp. 256–263, doi: 10.1145/345508.345593.

[7] Wang, K.—Zhou, S.—He, Y.: Hierarchical Classification of Real-Life Documents. Proceedings of the 2001 SIAM International Conference on Data Mining, 2001, doi: 10.1137/1.9781611972719.22.

[8] Sokolov, A.—Funk, C.—Graim, K.—Verspoor, K.—Ben-Hur, A.: Combining Heterogeneous Data Sources for Accurate Functional Annotation of Proteins. BMC Bioinformatics, Vol. 14, 2013, No. 3, Art. No. S10, doi: 10.1186/1471-2105-14-s3-s10.

[9] Cerri, R.—Barros, R. C.—de Carvalho, A. C. P. L. F.: Hierarchical Multi-Label Classification Using Local Neural Networks. Journal of Computer and System Sciences, Vol. 80, 2014, No. 1, pp. 39–56, doi: 10.1016/j.jcss.2013.03.007.

[10] Kurata, G.—Xiang, B.—Zhou, B.: Improved Neural Network-Based Multi-Label Classification with Better Initialization Leveraging Label Cooccurrence. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 521–526, doi: 10.18653/v1/n16-1063.

[11] Zhang, M. L.—Zhou, Z. H.: Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, 2006, No. 10, pp. 1338–1351, doi: 10.1109/tkde.2006.162.

[12] Nam, J.—Kim, J.—Mencia, E. L.—Gurevych, I.—Fürnkranz, J.: Large-Scale Multi-Label Text Classification – Revisiting Neural Networks. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (Eds.): Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2014). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8725, 2014, pp. 437–452, doi: 10.1007/978-3-662-44851-9_28.

[13] Kim, Y.: Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751, doi: 10.3115/v1/d14-1181.

[14] Song, Y.—Roth, D.: On Dataless Hierarchical Text Classification. AAAI Publications, Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec, Canada, 2014, pp. 1579–1585.

[15] Prakhya, S.—Venkataram, V.—Kalita, J.: Open Set Text Classification Using Convolutional Neural Networks. International Conference on Natural Language Processing, USA, 2017.

Milan Krendzelak is Tech Lead of Web Solutions Engineers team at Google Inc. that delivers innovative solutions to help drive revenue and make Google's global sales field more efficient. As Web Solutions Engineer, his responsibilities include prototyping proofs of concept, developing and supporting tools, and enhancing core products to meet the needs of his sales field. He earned a master of computer science degree in 2004 at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. Currently, he is continuing his Ph.D. research in Hierarchical Text Classification at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. He has published more than six scientific publications.

Frantisek Jakab is Director of the University Science Park TECHNICOM and Head of the Computer Networks Laboratory (`www.cnl.sk`) that he created during his career at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. He graduated from the Faculty of Computer Science and Electrical Engineering at the St. Petersburg Institute of the Electrical Engineering in the field of System Engineering (Russian Federation). Main areas of his research activities: computer networks, which is a new form of multimedia-based communication (video conferences, IP streaming). He is renowned author of more than 200 scientific publications and textbooks.