Implementation and Interpretation of CNN Text Classifier

Ethan Xuanyue Yang
xuanyuey@cs.cmu.edu

1 Introduction

In this report we we present the implementation of an CNN text classifier, specifically how convolution kernels are use to extract soft k-gram features from texts. We conduct a series of experiment and show the effectiveness of our settings of certain hyper-parameters by the comparason of performance results. We then discuss the factors: batch size, activation function and regularization methods, and some issues potentially hindering the improvement of the model. Further, we analyze the parameters of trained model and obtain the top k-grams of significance to the classification decision, as an interpretation of what the model have learned.

2 Model

The model is a direct implementation of CNN text classifier as described in (Kim, 2014). We first convert the input tokenized text (word ID sequence) $\{x_t\}_{t=1}^{T}$ to an word embedding sequence $\mathbf{X} \in \mathbb{R}^{T \times D}$. Then we apply a series of convolution kernel $\{\mathbf{W}_{\text{conv}}^{(i)} \in \mathbb{R}^{k_i \times D}\}_{i=1}^{F}$ to the embedding sequence in order to extract local features. For each kernel $\mathbf{W}_{\text{conv}}^{(i)}$, the convolution slides it along the embedding sequence and calculate the matching scores with each k_i-grams, yielding a feature map

$$\mathbf{c}_i = [\cdots, a(\langle \mathbf{W}_{\text{conv}}^{(i)}, \mathbf{X}[t:t+k_i] \rangle_{\text{F}} + b_i), \cdots]^{\mathsf{T}}, \tag{1}$$

where $\langle \cdot, \cdot \rangle_{\text{F}}$ is the element-wise Frobenius inner product of two matrices, a is a non-linear activation function and we adopt a non-zero bias term b_i.

We keep the most prominent feature score by max-pooling the feature map over time:

$$f_i = \max(\mathbf{c}_i). \tag{2}$$

We thus obtain a "bag-of-soft-k-grams" (compared to hard string matching) feature vector for the text:

$$\mathbf{f} = [\cdots, f_i, \cdots]^{\mathsf{T}}. \tag{3}$$

As proved effective in reducing over-fitting, dropout prevent co-adaptations among hidden neurons (different convolution kernels in this case) by randomly omitting outputs thereof. We dropout the feature vector with a probability p (set to 0 during inference):

$$\tilde{\mathbf{f}} = \frac{1}{1-p}\mathbf{f} \odot \mathbf{m}, \tag{4}$$

where

$$m_i \sim \text{Bernoulli}(1-p). \tag{5}$$

Finally we project the feature vector to a C-dimensional logit vector, and take the softmax of which as the predicted probability distribution over the C classes:

$$\mathbf{o} = \mathbf{W}_{\text{proj}}\mathbf{f}, \tag{6}$$

$$\hat{\mathbf{p}} = \text{softmax}(\mathbf{o}), \tag{7}$$

where

$$\mathbb{P}(y|\mathbf{x}; \mathbf{\Theta}) = \hat{p}_y. \tag{8}$$

We use the cross entropy between the true and predicted probability distribution as the loss function to equivalently maximize the conditional likelihood of the data:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \prod_j \mathbb{P}(y_j|\mathbf{x}_j; \mathbf{\Theta}) \tag{9}$$

$$= \arg\min_{\mathbf{\Theta}} -\sum_j \log \mathbb{P}(y_j|\mathbf{x}_j; \mathbf{\Theta}) \tag{10}$$

$$= \arg\min_{\mathbf{\Theta}} -\sum_j \sum_y \mathbb{I}(y_j = y) \log \mathbb{P}(y_j|\mathbf{x}_j; \mathbf{\Theta}) \tag{11}$$

$$= \arg\min_{\mathbf{\Theta}} \mathcal{L}_{\text{xe}}. \tag{12}$$

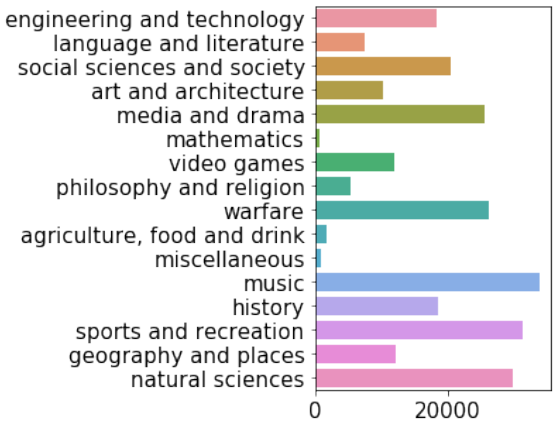

Figure 1: Counts of Each Class

To further regularize the model parameters, we add an ℓ_2 weight decay term to the loss function:

$$\mathcal{L} = \mathcal{L}_{\text{xe}} + \frac{\lambda_{\ell_2}}{2} \|\mathbf{\Theta}\|_2^2. \tag{13}$$

3 Experiment

3.1 Dataset

The dataset is obtained from the course Neural Networks for NLP (CMU 11747)[1], containing a training set of 253909 instances, a validation set of 643 instances and a unlabelled test set of 697 sentences, over $C = 16$ classes. This dataset is rather unbalanced as shown in Figure 1.

The texts in the dataset are already tokenized, and we build the vocabulary of size 35330, out of all tokens of frequencies larger than 5 in the training set.

3.2 Settings

We list the settings of the experiment in Table 1. During training we fine-tune the pretrained word embeddings, where we use sparse and normal Adam optimizer (Kingma and Ba, 2014) for embeddings and other parameters respectively, due to the sparse gradient property of embeddings.

3.3 Results and Analysis

We report the results in terms of validation accuracy in Table 2, where "Main" is the main model based on the settings described above and others

[1]http://phontron.com/data/topicclass-v1.tar.gz

[2]https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip. For a word not in this pretrained vocabulary, we randomly initialize its embedding from standard normal distribution.

[3](He et al., 2015)

Item	Value
Embeddings	fastText[2]
Embedding Dimension D	300
Kernel Widths $\{k_i\}$	$\{3, 4, 5\} \times 100$
Initializer for Weights	He Initialization[3]
Activation Function a	$\tanh$
Dropout Probability p	0.5
ℓ_2 Weight Decay Factor λ_{ℓ_2}	5×10^{-4}
Optimizer for Embeddings	SparseAdam
Learning Rate for Embeddings	5×10^{-4}
Optimizer for Others	Adam
Learning Rate for Others	10^{-4}
Batch Size	256

Table 1: Settings

Model	Validation Accuracy (%)
Main	**85.07**
(Static Embeddings)	82.89
($-$ ℓ_2 Weight Decay)	84.91
(Batch Size = 64)	83.98
(ReLU Activation)	84.76
($-$ Dropout)	84.91

Table 2: Results

are variation. As the validation set is relatively small and does not cover all the classes, the results on which could be unstable. Thus for each group of settings, we choose the best result by running on 30 different random seeds.

We have experienced one major improvement factors — fine-tuning of embeddings. Similar to the effects discussed in (Kim, 2014) and (Zhang and Wallace, 2015), adapting the embeddings to the training data along with other parameters, instead of keeping it static, could render them more task-specific, thus generally accelerating the convergence and achieving a better validation accuracy eventually.

A larger batch size helps, along with a larger learning rate accordingly, increasing the validation accuracy to some extent, which somewhat contradicts our original belief that small batch size and learning rate prevents over-fitting. We conjecture that it gives more precise estimate of the gradient and stabilizes the training process.

Choice of activation function, specifically ReLU and $\tanh$ make a slight difference in our results. Switching to $\tanh$ help improve out validation accuracy by some 0.3% absolute value.

There is no certainty whether ReLU and tanh would be better in this case, and they take turn giving the best results in different datasets as shown in (Zhang and Wallace, 2015). Although ReLU would not suffer the gradient vanishing issue, in this model of quite shallow depth, this advantage might not stand out. On the other hand, tanh might be better because of its bounded output compared to ReLU's, thus might be of benefit to the training of the next layer.

As shown in Figure 2, the model converges and over-fits quickly in just 3-4 epochs, while the training accuracy could climb all the way up to nearly 100%, yet the validation accuracy would stuck in the range from 81%-84%, and gradually decrease. We adopt mainly dropout and ℓ_2 weight decay to ameliorate over-fitting, which is effective but slightly. One problem is that the validation set is truly small and might not be sufficiently representative of the data. In light of this, it might not worth attempting to "over-fit" the validation set.

Another issue is the unbalanced class distribution in this dataset, making it harder to train an unbiased model that generalizes well. We show in Figure 3 and 4 the accuracies of each class and the confusion matrix among classes respectively. It could be told that the model would to some extend favor major classes such as `video games` (6) and `warfare` (8). For theses classes, the model could achieve high accuracies, yet would also tend to assign instances of other classes to them.

A scrutiny of the wrongly classified results on the validation set also give a several potential problem concerning the discriminability of the data. We list below several bad cases, where the red color texts are wrong predictions, and they seem to be also a sound class attribution for the text content. We suspect that even human cannot properly classify cases as these, let alone a model.

1. (`warfare` → `history`)
   `the british organized an expedition in early 1776 for operations in the rebellious southern colonies of north america .`

[1]Several classes: `mathematics` (5), `philosophy and religion` (7), `agriculture, food and drink` (9) and `miscellaneous` (10) do not occur in the validation, whose accuracies should be "NULL", rather than 0.

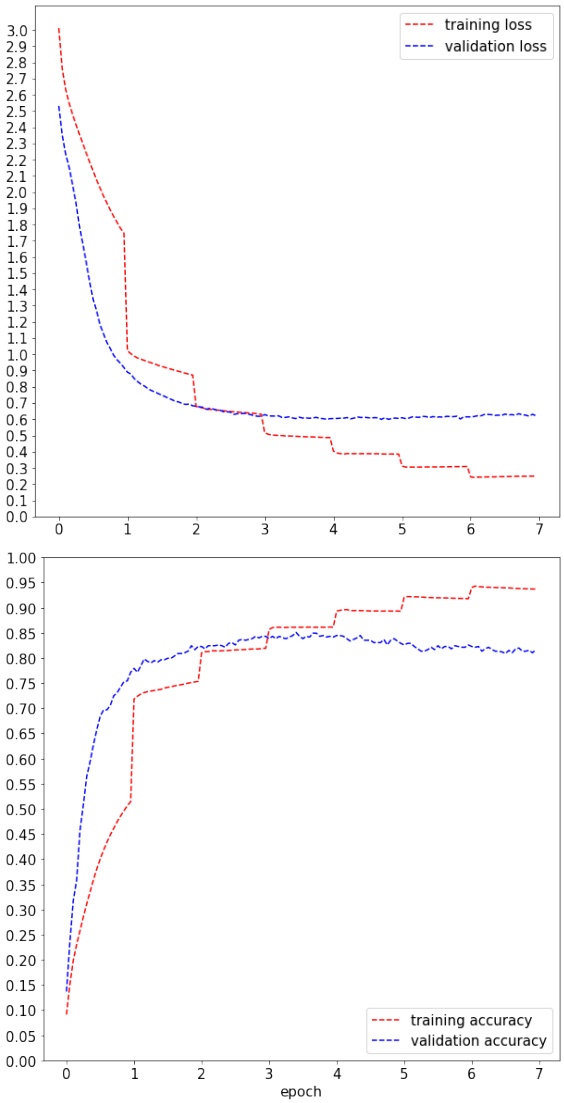

Figure 2: Losses and Accuracies During Training

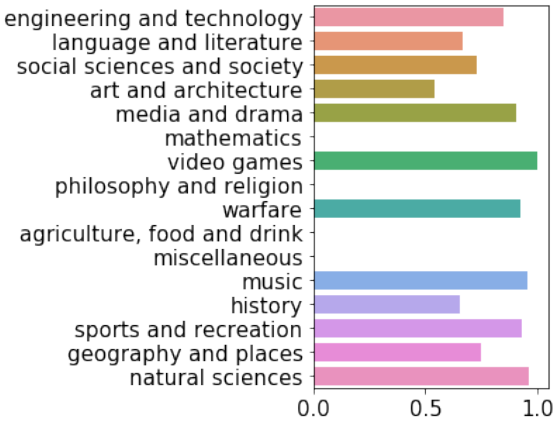

Figure 3: Accuracies of Each Class[1]

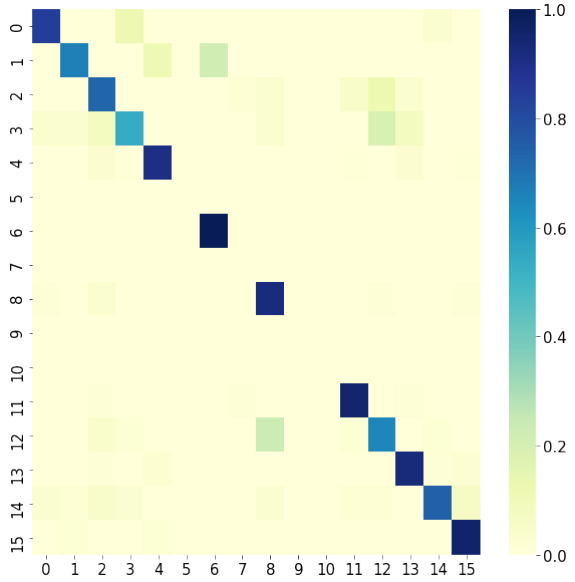

Figure 4: Comfusion Matrix

2. (sports and recreation → media and drama)
" when richmond was cast for a bit part in the 1983 movie <unk> ace , " he fell right in with the group working on the film , " said director hal <unk> .

3. (social sciences and society → music)
when the band returned to the czech republic for another concert two years later , its vocalist randy blythe was arrested , charged with causing <unk> 's death , and <unk> in custody for five weeks .

3.4 Interpretation of Model

We intentionally keep the model sufficiently simple for ease of an potential interpretation. The interpreting process resembles a traceback of the model to find:

1. for each classes, which kernels contribute most to its decision;

2. for each kernel, which k-grams match the best with it.

Through such we attempt to figure out which k-gram features the model have learned FROM SCRATCH so as to classify the text.

For each class y We first calculate the kernel scores by average over all N training instances:

$$\mathbf{s}^{(y)}_{\text{kernel}} = \frac{1}{N}\sum_{j} \mathbf{w}^{(y)}_{\text{proj}} \odot \mathbf{f}_j, \tag{14}$$

where the Hadamard product gives a proper significance socre for each of the kernel, since when deciding the output for class y, we have

$$o_y = \mathbf{w}^{(y)}_{\text{proj}} \cdot \mathbf{f} = \|\mathbf{w}^{(y)}_{\text{proj}} \odot \mathbf{f}_j\|_1. \tag{15}$$

We keep the top-m_{kernel} kernel indices:

$$\mathcal{I}^{(y)}_{\text{top-kernels}} = \text{top\_indices}(\mathbf{s}^{(y)}_{\text{kernel}}, m_{\text{kernel}}). \tag{16}$$

Then for each kernel index $i \in \mathcal{I}^{(y)}_{\text{top-kernels}}$, we retrieve the top-$m_{k_i-\text{gram}}$ among all its N max-pooled results:

$$\mathcal{G}^{(y,i)}_{\text{top}-k_i-\text{grams}} = \text{top\_k\_grams}(\{(\mathbf{x}_j[t:t+k_i], f^{(j)}_i)\}_{j=1}^{N}). \tag{17}$$

We thus could regard these k_i-grams as the features kernel i would extract for predicting class y, which we list in Table 3 with $m_{\text{kernel}} = 2$ and $m_{k-\text{gram}} = 2$ due to page limit. A brief observation is that, for classes with higher frequencies in the dataset, the k-grams we extract are more reasonable, more in alignment with the "topic" of the class. But for minority classes such as `math`, the training data contains more `<unk>`, and even non-`<unk>`'s would not get sufficiently trained.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

engineering and technology	**warfare**
freeway between winton highway connects folkston	the jna ibenik battle of taranto
with other maunsell locomotive classes orangeville , connecting highway 24	9 july 1918 mccudden raaf station amberley ,
language and literature	**agriculture, food and drink**
the codex vaticanus (the the codex vaticanus was <unk>	<unk> @-@ <unk> <unk> regions earlier @-@ <unk> <unk> clone
leaving lucasarts . holley chivers (	beecher <unk> in @-@ <unk> savoy
social sciences and society	**miscellaneous**
<unk> ramrez <unk> and <unk> as bethune @-@ <unk> university	<unk> or <unk> <unk> , <unk> @-@ <unk> subtype ,
praised osbert 's , osbert served	a novel vaccine for dengue coeliac disease was first described
art and architecture	**music**
painting velzquez is etty was prolific	studio album music indian album music
statue of horatio saint cuthbert of	the album mwng pollstar concert industry
media and drama	**history**
features sivaji ganesan , rajinikanth wordless novel with frans masereel	leader draa mihailovi to thomas beauchamp
stars sivaji ganesan being sivaji ganesan	nikolaevna <unk> ; frakkk <unk> with
mathematics	**sports and recreation**
@-@ <unk> homotopy znm problem <unk>	professional wrestling career , palomeque three players , fred wheldon
<unk> and <unk> <unk> enabled <unk> , <unk> <unk> ,	bastogne lige played at highbury
video games	**geography and places**
the game 's filmation game latter game 's filmation ii	the river moselle ruislip @-@ northwood
rhythm game developed by harmonix video game developed by harmonix	johor bahru railway johor bahru retained
philosophy and religion	**natural sciences**
diocletian , maximian , galerius <unk> <unk> <unk> <unk>]	elita weakened to tropical depression typhoon hazen , typhoon irma
@-@ rosalie cadron saint @-@ andr	tropical cyclone magda agaric genus tricholoma

Table 3: Interpreted Features for Each class