DSGA1011 Homework 1: Bag of N-grams Classification

Rong Feng - rf1316

October 2018

1 Summary

As proscribed in the assignment description, this exercise uses bag-of-ngrams and a fully connected linear layer to classify positive and negative IMDB reviews. The final validation accuracy achieved after hyperparameter optimization was xx.x% and final text accuracy was xx.x%.

Following the assignment recommendation, the training set size was set to 20000 and validation to 5000. The test set of 25000 was only used at the end to provide the test accuracy and not touched or optimized over in any other way.

The final model used was ... ...

2 Architecture

As specified in the assignment, the model architecture is simply a ngram embedding lookup layer, followed by sum function to encode the bag-of-ngrams, and finally a fully-connected linear layer that outputs the logit of the predicted classes. The used loss was binary cross entropy loss (1)

$$l = -(y * log(p) + (1 - y) * log(1 - p)) \tag{1}$$
$$y \in \{1, 0\} = the\ class\ label$$
$$log(p) = the\ model\ output\ logit$$
$$p = the\ model\ predicted\ probability$$

3 Data Preparation

Using the github code base for the assignment in 2017 as a starting point, functionality was added to perform the required data preparation [1]. These functionalities include: Tokenization, ngram extraction, building and indexing the vocabulary, and finally loading the data into the PyTorch DataLoader pipeline. Each of these functionalities and their associated hyperparameters are described below:

Tokenization: To convert the raw text of the reviews to lists of tokens two methods were employed: "naive" and "spacy" methods. For the naive method, the sentences were tokenized by splitting them with the space delimiter. For the spacy method, the smart-tokenization implementation of spacy was called on the input sentence. For example under spacy, "don't" is tokenized into "do" and "n't", extracting the meaning out of the english abbreviation. In both methods both stop words and punctionation are moved if specified by their respective hyperparameters.

Ngram Extraction: From the list of tokens, we iterate through them to extract all ngrams of size equal or less than the specified hyperparameter.

Building Vocabulary: Using the python Counter object, we take the top n most frequently occuring ngrams and form our vocabulary of ngrams, where n is determined by the VOC_SIZE hyperparameter. For each input review, its ngrams are then mapped to the index of that ngram in the vocabulary. All encountered ngrams that do not belong to the vocabulary are mapped to a special ⟨unk⟩ placeholder. Further, a special ⟨pad⟩ placeholder is added at index 0 in the vocabulary used for input padding.

Converting to PyTorch DataLoader: Finally, a collate function was implemented in PyTorch's required format. It takes the ngram indexes of the inputs and converts a batch of inputs into a PyTorch Tensor representation. The input data is also wrapped by the DataLoader to take advantage of various convienence implementations in the pytorch DataLoader such as shuffling and batching.

4 Establishing Baseline

Starting with heuristically selected hyperparamters as a source configuration, we run basic ablation analysis over single or sets of hyperparameters to arrive at our base-line with the following hyperparameters in Table 1

Hyperparameter	Description	Baseline Value
LR	Learning Rate on used Optimizer	0.01
LR_DECAY_RATE	Gamma decay rate per epoch	0.95
NEPOCH	Number of Total Epochs	10
VOC_SIZE	Vocab Size	100k
EMBEDDING_DIM	Embedding Vector Dimension	50
NGRAM_MODE	mode of tokenizer (spacy, naive)	spacy
REMOVE_STOP	Whether to remove stop words	True
REMOVE_PUNC	Whether to remove punctuation	True
EARLY_STOP	Whether to consider early stop	True
ES_LOOKBACK	# of batches to look back for improvement	True
ES_MIN_IMPROVE	Minimum improvement required for ES	True

Table 1: Baseline Hyperparameters

5 Further Hyperparameter Tuning

Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum Lorum Ipsum

6 Filler

Lorum Ipsum Lorum Ipsum

References

[1] NYU Data Science Department. Github Repo: DS-GA-1011 Fall 2017.