IN5550 – Spring 2021
1. Bag of Words Document Classification

UiO Language Technology Group

Deadline: 7 February, at 21:59 (Devilry)

Goals

1. Learn how to use the Saga cluster to train deep learning models.
2. Get familiar with the *PyTorch* library
3. Learn how to use *PyTorch* to train and evaluate neural classifiers in NLP tasks.

Introduction

This is the first obligatory assignment in the IN5550 course, Spring 2021. It is relatively small, and the goal is to introduce you to the basics of neural network approaches to classification tasks in natural language processing. For this task, you will implement a document classifier based on a simple feed-forward neural network, using bags-of-words as features. The recommended machine learning framework in this course is *PyTorch*; we provide a ready-to-use module and Python environment on the Saga cluster[1], and we strongly encourage you to train your models using Saga. If you would like to use any other libraries, (i.e. not *PyTorch*), this is allowed, provided that you implement your own classifier from scratch, and that your code is available.

Please make sure you read through the entire assignment before you start. Solutions must be submitted through Devilry[2] by 21:59 on February 7. Please upload a single PDF file with your answers. Your heavily commented code and data files should be available in a separate private repository (that you will continue to use for your other assignments) on UiO's Git platform[3]. Make your repository private, but allow full access to IN5550 teaching staff[4]. The PDF in Devilry should contain a link to your repository. If you have any questions, please raise an issue in the IN5550 Git repository[5], email *in5550-help@ifi.uio.no*, or ask on the Mattermost chat. Make sure to take advantage of the group sessions on January 26 and February 2.

[1] `https://www.uio.no/studier/emner/matnat/ifi/IN5550/v20/setup.html`
[2] `https://devilry.ifi.uio.no/`
[3] `https://github.uio.no/`
[4] `https://github.uio.no/orgs/in5550/people`
[5] `https://github.uio.no/in5550/2021/issues`

Recommended reading

1. **Neural network methods for natural language processing.** Goldberg, Y., 2017.[6]
2. **Speech and Language Processing**. Daniel Jurafsky and James Martin. 3rd edition draft of September 23, 2018. Chapter 7, 'Neural Networks and Neural Language Models' [7] (sections 7.1, 7.2, 7.3 and 7.4)
3. Linear Algebra cheat sheet by the LTG[8]
4. `https://pytorch.org/`
5. `http://research.signalmedia.co/newsir16/signal-dataset.html`

1 Implementing a neural classifier

We will be working with the *Signal Media One-Million News Articles Dataset*, which contains texts from news sites and blogs from September 2015. The dataset features several fields of metadata for each article, but in this assignment we will be interested in only one of them: the *source* of an article, i.e., from which news site it comes. This is a supervised classification task, since we have gold labels for this field. You are to implement a simple feed-forward neural network which takes bag-of-words (BoW) text representations as inputs and produces *source* predictions as an output, basically classifying the texts based on their source. The task therefore involves processing the text data to extract BoW features, formatting these appropriately for the deep learning framework you're using, and experimenting with the classifier hyperparameters to find an optimal solution.

The assignment is divided into six parts:

1. **Data processing**;
2. **Training a classifier**;
3. **Feature tuning**;
4. **Measuring time efficiency**;
5. **Model evaluations**.

You are to submit a written report (2-4 pages) to Devilry, which provides details on your experiments and addresses the questions posed in the assignment. Your code should be available in your Git repository; irrespective of what framework you decide to use, the programming language must be Python. Your code must be self-sufficient, meaning that it should be possible to run it directly on the data. If you use some additional data sources, these datasets should be included in your repository as well. If you use non-standard *Python* modules, this should be documented in your report.

[6] `https://www.morganclaypool.com/doi/10.2200/S00762ED1V01Y201703HLT037`
[7] `https://web.stanford.edu/~jurafsky/slp3/7.pdf`
[8] `https://www.uio.no/studier/emner/matnat/ifi/IN5550/v20/la.pdf`

1.1 Data processing

We use the *Signal Media One-Million News Articles* dataset. In this assignment, you'll have to work with a subset of it, containing articles from 20 news web sites represented best in the main dataset: 'MyInforms', 'Individual.com', etc. These sites are different in their dominant topics and vision; thus, we expect that the words used in their news articles would reflect these differences. Overall, the corpus contains about 75,000 articles and about 16 million word tokens.

The training dataset is available on the UiO Github[9]. It is provided as a gzipped tab-separated text file, with one line corresponding to one news article. The columns are:

- **source** (class)
- **text**

We also have a held-out test set of 8,350 documents, which will not be published until we receive all the submissions. Your solutions will then be evaluated on this held-out test set, so that we can rank the systems by their objective performance.

The texts are already lemmatized and POS-tagged, and stop words and punctuation signs are removed, to make NLP tasks easier. Thus, a sentence '*But the report by Secure Fisheries, a part of the One Earth Future Foundation campaign group, warned those advances could be reversed if illegal fishing is not stemmed.*' is converted to '*report_NOUN secure_NOUN fishery_NOUN part_NOUN one_NUM earth_NOUN future_NOUN foundation_NOUN campaign_VERB group_NOUN warn_VERB advance_NOUN reverse_VERB illegal_ADJ fishing_NOUN stem_VERB*'. Note that the *Universal POS tags* tag set was used for part of speech tagging[10]. The documents are shuffled, so there is no inherent order in the dataset.

We want to train a classifier which will be able to predict the source of a news text based on the words used in it. For this, we need to somehow move from documents as character sequences to documents as feature vectors that can be fed to a machine learning algorithm, such as a feed-forward neural network. The obvious way to do this is to represent all the documents as bags-of-words: that is, extract the collection vocabulary (the union of all word types in the texts) and count frequency of each word type in each document (or mark the binary value of whether the word type appears in the document or not). Documents are then represented as sparse vectors of the size equal to the size of the vocabulary. Machine learning classifiers can be trained on such data, which is what you are going to do.

We provide the dataset as is, but before training you have to split it into **train** and **development** parts. Make sure to keep the distribution of source labels as similar as possible in both subsets. You can use the `scikit-learn` library, or write your own code, using PyTorch's `torch.utils.data` as a base if necessary. Describe the process of splitting the dataset in your PDF report in detail.

[9] `https://github.uio.no/in5550/2021/blob/master/obligatories/1/signal_20_obligatory1_train.tsv.gz`

[10] `https://universaldependencies.org/u/pos/all.html`

1.2 Training a classifier

We formulate our classification task as follows: given a list of words in a document, **predict the source** of this document.

You must write the code which takes the provided dataset and:

- infers a common vocabulary for all documents in the dataset;
- extracts bags of words for each document;
- extracts a list of gold source labels for the documents (for example, `sources[i]` contains the source label for the i_{th} document);
- all this serves to prepare the data to be fed into a classifier.

Briefly describe this code in the report.

Then implement a fully-connected **feed-forward neural network** (multilayer perceptron) which trains on these bag-of-words features and evaluate it on the development dataset (see the '*Evaluation*' subsection below). For a start, you can look at the classifier code discussed at the group sessions. Describe the parameters of your network and the evaluation results in the PDF report.

1.3 Feature tuning

Try to experiment with feature tuning. Introduce at least 3 variations of your BoW features. In order to do so you may glance at the lecture slides or at the literature for inspiration. Write a short description of your experiments. Provide examples.

1.4 Time efficiency

Try at least 5 different values for the **number of hidden layers** hyperparameter in your neural network (starting from 1). For each number of layers, calculate the performance of the resulting model **and** the time it took to train this model. Draw a plot showing how performance and training time depend on the number of layers. This plot should be included in the PDF report, and the code you used to produce it must be committed to your repository.

1.5 Evaluation

You should report the following performance metrics on the development (validation) set for each of your models:

1. accuracy;
2. precision, recall and macro-F1 score.

You can calculate these scores using either *PyTorch*, *scikit-learn*, or any other mainstream machine learning library.

Describe the differences in your systems' performance in the PDF report. Give your opinion on the influence of different hyper-parameter values. Finally, choose your best-performing system and publish it in your UiO Github repository as a saved *PyTorch* model.

Recall that non-linear models sometimes can produce different results with the same hyperparameters because of different random initializations. Thus, train your best model 3 times and evaluate it 3 times, providing the average and the standard deviation.

What are the results? Are they worse or better than on the train data? Why? Report the results both in the PDF report and in a machine-readable format in your repository.

You also have to provide an `eval_on_test.py` script. It should take as an input a path to a saved model and a test dataset (presume it is in the same format as the training dataset), and return accuracy, precision, recall, and F1 scores for each class in the test set, as well as their macro averaged values. We will use this script to evaluate your classifier on the held-out test set. If the eval script fails to execute correctly without us having to modify your code, we will deduct 1 point from your submission. Please also give your models sensible names.

Good luck!