ELEC0141 Deep Learning for Natural Language Processing Assignment (2023)

General overview

The DLNLP assignment consists of writing individual code, training and testing on data, preparing a conference paper-style individual report, and providing supplementary material through an online code repository of your choice. While you can discuss ideas with peers, all code, experiments, and report must be your own work.

Some practical tips:

- Clearly define your goals: At the start of your project, outline specific objectives and create a detailed timeline. Be realistic about what you can achieve, considering your current experience and the available time frame. Consider the following questions: What NLP task do you want to solve? What deep learning tools will you use? What datasets will you use? What are the inputs and outputs? Provide specific examples.
- Conduct a literature review: This will give you an understanding of which approaches are likely to be successful and which have already been attempted.
- Consult with the course staff: If you need guidance, reach out to the course staff for help via email or Moodle. They may be able to point you in the right direction.
- Don't be afraid of negative results: Remember that a negative outcome is not a failure in this course project. We are looking for a thorough project plan and execution.

Expectations:

Your project should demonstrate your understanding of the Deep Learning and NLP techniques covered in class. You are not expected to make a significant scientific breakthrough or outperform state-of-the-art methods. The goal is to correctly apply the modeling techniques learned in this course to an NLP task.

Your report should provide a thorough explanation of your experimental findings, including qualitative and quantitative analysis. For example, identify specific examples where your model did not perform as expected.

The performance of your methods will not be the sole factor in grading the project. Projects will be evaluated holistically, taking into account criteria such as originality, the

complexity of techniques used, thoroughness of evaluation, amount of effort put into the project, analysis quality, and report quality.

Assignment summary

- The assignment covers elements from the DLNLP lectures, lab sessions, and relevant research literature, competitions, and datasets associated with deep learning NLP systems. A list of project ideas is provided in the assignment description, and each student must choose one and provide their own solution.
- The solution should be implemented in a programming language of the student's choice, but it is worth noting that the labs are based on Python and PyTorch. Third-party components such as neural network designs, pre-trained tokenizers, or noise robustness techniques may be used, but the report and code must clearly indicate what was developed by the student and what was sourced from external sources.
- A report summarizing all steps taken to solve the task, including the NLP task, model, and design choices, and the results obtained from experiments and the accuracy of predictions on unseen data should be written. Please refer to the Report and Code Format, and Marking Criteria section for more information about the report.

The goal of the assignment

- To further develop your programming skills.
- To further develop your skills and understanding of NLP and deep learning systems and tasks.
- To acquire experience in dealing with real-world data and real-world research challenges.
- To develop good practice in model training, validation, and testing!
- To read state-of-the-art research papers on NLP and deep learning and understand the current challenges and limitations.
- To develop your writing skills by presenting your solutions and findings in the form of a conference paper.

Assignment description

You can choose any topic related to deep learning applied to NLP. In addition, your project should make use of the methods taught in this class. If you have any questions about what an acceptable product would entail, please feel free to contact the course staff.

Project types

Here is a non-exhaustive list of possible project types:
• applying an existing pre-trained model to a new application or task and exploring how to approach/solve it
• transferring an existing model, method or technique from another field to NLP;
• proposing a new model or architecture for an NLP task and demonstrate its performance on some data (don't be scared of failures)
• providing an experimental analysis of an existing model, e.g. performing a replication study of an existing paper with ablation studies or a few more experiments on other data sets;
• proposing a new training method or evaluation scheme.

Example projects: Other institutions that offer similar courses have similarly structured course projects, e.g., Stanford's CS224n[1]. You can look at their sample projects for some inspiration. Still, everybody should take into consideration their current experience and the time frame.

For your chosen challenge, you should report training errors, validation errors, hyper-parameter tuning, and comparisons with related methods in terms of accuracy and complexity. It is important to make an algorithmic design that balances complexity versus accuracy, as you will not have access to GPU hardware!

You may however explore options of free cloud credit for CPU/GPU hardware available for students via public cloud providers, or serverless versions of such services as Colab notebooks: https://colab.research.google.com/notebooks/intro.ipynb

Overall, given your limited access to compute resources, you are allowed to limit the amount of training data with a proportional reduction in the complexity of the inference task (e.g., number of classes in your sentiment analysis task, size/dimension of the input/output data, classes or inference task(s)).

While much of today's NLP research makes heavy use of computational resources, many interesting analyses can still be done without utilizing GPUs for days on end! As such, please note that you are not required (nor encouraged) to train new, large neural networks or do extensive hyperparameter searches as this may not be feasible given the resources that we can provide.

[1] https://web.stanford.edu/class/cs224n/project.html

Fantastic datasets and where to find them

There are a number of online resources that host standard NLP datasets. We present a few here:

• NLP Progress and Papers with Code: Repositories that track state-of-the-art performance for the most common NLP tasks. They also include many of the common datasets for these tasks (https://paperswithcode.com/sota)

• Social Media and Online Platform Data: many online platforms provide APIs or readily compiled data dump files. Consider Twitter, Reddit, YouTube transcripts, Wikipedia, Reuters (e.g., https://developer.twitter.com/en/docs/twitter-api, https://dumps.wikimedia.org/)

• Kaggle: a platform for hosting competitions on tasks that also provides many datasets.

• Alphabetical list of datasets in the public domain with text data for use in Natural Language Processing: https://github.com/niderhoff/nlp-datasets

• Emailing the Authors: If you come across an interesting dataset that is not publicly available, you may consider contacting the authors. Often, authors of academic papers are more than happy to email you a copy of their dataset.

Note that just because a dataset is online and already in a single, structured file does not mean the dataset has been processed in a manner suitable for your project. Take care to analyze any online resources to first, ensure their quality and second, determine what additional steps you may need to take in order to properly format the data.

On collecting new data: While it is possible to collect your own data for your project, we do not recommend it. Data collection is often a time-consuming and messy process that is more difficult than it appears. If you choose to do so, make sure to budget the data collection time into your project and provide a thorough explanation of your processing techniques in the write-up. Note that your project must have a substantial modeling component; if you spend all your time on data collection and none applying modeling techniques, your grade will suffer.

Report and Code Format, and Marking Criteria

Report format and template

We provide latex templates in ***DLNLP_assignment_kit*** on the ELEC0141 DLNLP Moodle. The criteria for each part are detailed in the template. For beginners in latex, we recommend overleaf.com, which is a free online latex editor.

Once you finish your report, please export it into a PDF document and name it in the following format:

Report_DLNLP_23_SN12345678.pdf

The report should be 8 - 10 pages long. You can also have supplementary material (that won't be graded).

Code criteria

You should write your code in modules and organize them in the fashion included in the folder structure of the DLNLP assignment kit mentioned above. If we wish to check the results of your code, we will first run your code with a script. Therefore, please make sure your project is named properly as demonstrated above.

- The '**Datasets**' folder does not have to contain the raw datasets. You can use it to save the datasets you may have generated (e.g., if you did text pre-processing, data augmentation, etc.).
- The '**A**', '**B**', … folders should contain the code files for each subtask (depending on your assignment, you may have two or more subtasks).
- The **README** file should contain
    - a brief description of the organization of your project;
    - the role of each file;
    - the packages required to run your code (e.g. numpy, scipy, transformers, etc.).
    - The recommended format for the **README** file is markdown (**.md**). **.txt** is acceptable too.
- We should be able to run your project via '**main.py**'. The structure of '**main.py**' has been provided.
- You are NOT going to upload your code and dataset to Moodle. Please refer to the Submission section for more details.

Marking scheme

The mark will be decided based on both the **report** and the **corresponding code**. In particular, we will mark based on the following scheme:

REPORT	80%	CORRESPONDING CODE	20%
Abstract	5%	-	-
Introduction	5%	-	-
Literature survey	10%	-	-

Description of models (Use flow charts, figures, equations etc. to explain your models and justify your choices)	20%	-	-
Implementation (the details of your implementation, explain key modules in your code.)	20%	Correct implementation	10%
Experimental Results, Analysis and Conclusion	20%	The code is expected (or confirmed) to be producing reasonable results if executed	10%

It should be noted that – whereas we expect students to develop Deep Learning NLP models delivering reasonable performance on the chosen challenge – the assessment will not be based on the exact performance of the models. Instead, the assessment will predominantly concentrate on how you articulate the choice of models, how you develop/train/validate these models, and how you report/discuss/analyze the results.

Submission

- **Deadline**: Please see the ELEC0141 DLNLP Moodle page
- **Report submission**: You should only submit your report on Moodle
- **Code submission**: You must include a link to your code in an internet repository that is publicly accessible in your report, but the link is hidden (e.g., GitHub, public Dropbox or Google Drive link, or similar).

  You are also encouraged to use GitHub to save and track your project. Use your UCL github account (or create an account) to start a git repository named **DLNLP_assignment_23/**. Make sure to back-up your code on the git repository regularly and keep your repository private so it is not viewable by other students. Changes made after the assignment deadline may not be taken into account. The

code should be well documented (i.e., each class and function should be commented), and an additional README.md file containing instructions on how to compile and use your code should be created in the repository. If necessary, we reserve the right to test the code and we may ask you to provide us with your GitHub commit history evidencing how you gradually built and tested your solution.

Final advice

Data processing, even for already “clean” datasets, can be cumbersome. It is worth investing time in a thoughtful, well-constructed pipeline to assure you do not have to redo experiments. The following should all be considered:

Cleaning. You should consider if your data need to be tokenized, parsed, or annotated. The following widely used and well-documented resources may be helpful:
• NLTK (https://www.nltk.org/): a lightweight NLP toolkit written in Python;
• Moses (https://github.com/moses-smt/mosesdecoder): an SMT library that provides tokenization and cleaning scripts with coverage in many languages.
• StanfordNLP (https://stanfordnlp.github.io/stanfordnlp/): a Python library providing tokenization, tagging, parsing, and other capabilities.

Splitting. At the beginning of your project, you should split your data set into training data (most of your data), validation data, and test data. A typical train–dev–test split might be 80–10–10. Many NLP datasets come with predefined splits, and, if you want to compare against existing work on the same dataset, you should use the same splits. Explicitly, training data can be used to estimate the parameters of your model. The validation data should be used for model selection, but not parameter estimation. In terms of neural modeling, model selection may consist of choosing the best hyperparameters for an architecture or deciding whether to stop your optimizer early to avoid overfitting. The test data should only play a role at the end of your project. You should only evaluate your model once on the test data.

Baselines. NLP is largely an empirical science in that most recent progress in the field was made through experimentation. For this reason, good experimental practice is incredibly important. One aspect of good experimental practice is selecting a good baseline to benchmark your method and properly comparing against that method. Without a fair comparison to previously proposed methods, we cannot tell whether an innovation is actually an improvement over what the field currently offers. For instance, comparing against a simple baseline may reveal that your task does not actually require a complex, resource-intensive model. In general, simpler methods should be preferred. Regardless of your chosen research topic, we will expect some sort of baseline comparison in your final report. Baselines can include simpler models or models proposed by other works. You should first make an effort to understand what the baselines are. For example, suppose you’re building a multilayer LSTM-based network to do binary sentiment analysis. The simplest baseline is the guessing baseline, which would achieve 50% accuracy (assuming the dataset is 50% positive and 50% negative), or the mode baseline, which would achieve approximately the accuracy of the positive/negative split in the training set.

A more complex baseline would be a simple non-neural algorithm, such as logistic regression or a Naive Bayes classifier. Please look online before starting the grueling process of building a complex model from scratch!

There are a number of software tools and packages that can aid in your research. You are allowed (and encouraged) to make use of these tools, however, if you do so, please be explicit about it in your writeup.
These include but are not limited to:
• PyTorch https://pytorch.org/
• TensorFlow https://www.tensorflow.org/
• scikit-learn https://scikit-learn.org/stable/
• HuggingFace Transformers https://github.com/huggingface/transformers and fairseq https://fairseq.readthedocs.io/en/latest/

As is the case with collecting data from scratch, we do not recommend building complex neural models from scratch. While coding an LSTM from scratch may be an interesting exercise, it will likely take more time than you want to spend. After all, the goal of this project is to answer a research question. Making use of pre-trained models in your modeling process is fine. If you are unsure of your practices, please contact the course staff.