Guidance Notes for Final Projects

February 2020

1 Introduction

These notes complement the information provided in the “Final Project Practical Tips.pdf”. In particular this document contains additional links to useful resources. This document does not contain a detailed specification for the project deliverables (proposal, midterm, report and presentation) – those specifications are available in separate documents.

2 Choosing a Project Topic

Project Suitability

You can choose any topic related to **NLP** where you will be implementing **a deep learning system**. That means your project should make substantive use of deep learning as techniques for textual data in support of an empirical research question.

Project types

Here is a (non-exhaustive!) list of possible project types:

1. Applying an existing NLP model to a new task or problem
2. Implementing a complex NLP architecture
3. Proposing a new NLP model (or a new variation of an existing model)
4. Proposing a new training, optimisation, or evaluation scheme
5. Experimental and/or theoretical analysis of NLP models

Project ideas from Hertie faculty

We have collected a list of project ideas from Hertie faculty. These are a great opportunity to work on an interesting research problem. If you want to do these, get started early!

2.1 Expectations

Your project should aim to provide some kind of scientific knowledge gain, similar to typical research papers at scientific conferences. A typical case is that your project will show that your proposed method provides good results on a task you’re dealing with.

Given that you only have a few weeks to work on your project, it is not necessary that your method beats the state-of-the-art performance, or works better than previous methods. But it should at least show performance broadly expected of the kinds of methods you’re using.

In any case, your paper should try to provide reasoning explaining the behaviour of your model. You will need to provide some qualitative analysis, which will be useful for supporting or testing your explanations. This will be particularly important when your method is not working as well as expected.

Ultimately, your project will be graded holistically, taking into account many criteria: originality, performance of your methods, complexity of the techniques you used, thoroughness of your evaluation, amount of work put into the project, analysis quality, writeup quality, etc.

2.2 Finding existing research

Generally, it's much easier to define your project if there is existing published research using the same or similar task, dataset, approaches, and/or evaluation metrics. Identifying existing relevant research (and even existing code) will ultimately save you time, as it will provide a blueprint of how you might sensibly approach the project. There are many ways to find relevant research papers:

- You could browse recent publications at any of the top venues: ACL, EMNLP, TACL, NAACL, EACL, NIPS, ICLR, ICML. This list is not exhaustive!
- Try a keyword search at:
  - http://arxiv.org/
  - http://scholar.google.com
  - http://dl.acm.org/
- Look at publications from top research groups (e.g. Stanford NLP group) https://nlp.stanford.edu/pubs/
- Many top ML/NLP venues will have workshops on more applied topics that contain recent research on that task or problem. Here is an example of a workshop at ICML on the contribution AI/ML can make to climate change research: https://www.climatechange.ai/ICML2019_workshop.html

2.3 Finding datasets and tasks

There are lots of publicly-available datasets on the web. Here are some useful resources to find datasets:

- Wikipedia has a list of machine learning text datasets, tabulated with useful information such as dataset size. https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
- Kaggle has many datasets, though some of them are too small for ML. https://www.kaggle.com/datasets
- Datahub has lots of datasets, though not all of it is Machine Learning focused https://datahub.io/collections
- Microsoft Research has a collection of: https://msropendata.com
- Google Dataset Search: https://toolbox.google.com/datasetsearch
- A script to search arXiv papers for a keyword, and extract important information such as performance metrics on a task: https://huyenchip.com/2018/10/04/sotawhat.html
- A collection of links to more collections of links to datasets: http://kevinchai.net/datasets
- A collection of papers with code on many tasks: https://paperswithcode.com/sota
- Many social science datasets are deposited (and indeed this is a requirement for top journals!) on the Harvard Dataverse: https://dataverse.harvard.edu

2.4 Collecting your own data

It is possible to collect your own data for your project. However, data collection is often a time-consuming and messy process that is more difficult than it appears. Given the limited timeframe, we generally don't recommend collecting your own data. If you really do want to collect your own data, make sure to budget the data collection time into your project. Remember, your project must have a substantial NLP with deep learning component, so if you spend all your time on data collection and none on building DL models, we can't give you a good grade.

3 Project Advice

3.1 Define your goals

At the very beginning of your project, it's important to clearly define your goals in your mind and make sure everyone in your team understands them. In particular:

- Clearly define the task. What's the input and what's the output? Can you give an example? If the task can't be framed as input and output, what exactly are you trying to achieve?
- What dataset(s) will you use? Is that dataset already organised into the input and output sections described above? If not, what's your plan to obtain the data in the format that you need?
- What is your evaluation metric (or metrics)? This needs to be a well- defined, numerical metric (e.g. F1 score), not a vague idea (e.g. 'summary quality'). See below for more detail on how to evaluate your methods.
- What does success look like for your project? For your chosen evaluation metrics, what numbers represent expected performance, based on previous research? If you're doing an analysis or theoretical project, define your hypothesis and figure out how your experiments will confirm or negate your hypothesis.

3.2 Data hygiene

At the beginning of your project, split your data set into training data (most of your data), development data (also known as validation data) and test data. A typical train/dev/test split might be 90/5/5 percent (assigned randomly). 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 split as used in that work. Here is how you should use these data splits in your project:

1. Training data: Use this (and only this data!) to optimise the parameters of your NLP model.
2. Development data: This has two main uses. The first is to compare the performance of your different models (or versions of the same model) by computing the evaluation metric on the development data. This enables you to choose the best hyperparameters and/or architectural choices that should be evaluated on the test data. The second important usage of development data is to decide when to stop training your model. Two simple and common methods for deciding when to stop training are:
    (a) Every epoch (or every *N* training iterations, where *N* is predefined), record performance of the current model on the development set and store the current model as a checkpoint. If development performance is worse than on the last previous iteration (alternatively, if it fails to beat best performance *M* times in a row, where *M* is predefined), stop training and keep the best checkpoint.

    (b) Train for *E* epochs (where *E* is some predefined number) and, after each epoch, record performance of the current model on the development set and store the current model as a checkpoint. Once the *E* epochs are finished, stop training and keep the best checkpoint.
3. Test data: At the end of your project, evaluate your best trained model(s) on the test data to compute your final performance metric. To be scientifically honest, you should *only* use the training and development data to select which models to evaluate on the test set.

The reason we use data splits is to avoid *overfitting*. If you simply selected the model that does best on your training set, then you wouldn't know how well your model would perform on new samples of data – you'd be overfitting to the training set. In ML, powerful models are particularly prone to overfitting to their training data, so this is especially important.

Similarly, if you look at the test set before you've chosen your final architecture and hyperparameters, that might impact your decisions and lead your project to overfit to the test data. Thus, in the interest of science, it is extremely important that you don't touch the test set until the very end of your project. This will ensure that the quantitative performance that you report will be an honest unbiased estimate of how your method will do on new samples of data.

It's even possible to overfit to the development set. If we train many different model variants, and only keep the hyperparameters and architectures that perform best on the development set, then we may be overfitting to the development set. To fix this, you can even use two separate development sets (one of them called the tuning set, the other one the development set). The tuning set is used for optimising hyperparameters, the development set for measuring overall progress. If you optimise your hyperparameters a lot and do many iterations, you may even want to create multiple distinct development sets (dev, dev2, ...) to avoid overfitting.

3.4 Build strong baselines

A baseline is a simpler method to compare your more complex ML system against. Baselines are important so that we can understand the performance of our systems in context.

For example, suppose you're building a deep neural network to do binary classification. The simplest baseline is the guessing baseline, which would achieve 50% accuracy (assuming the dataset is 50% positive and 50% negative). A more complex baseline would be a simple non-neural Machine Learning algorithm, such as a logistic regression or Naive Bayes classifier. Lastly, you should compare against simpler versions of your full model, such as a model containing only the most basic theoretically relevant features.

You can also implement ablation experiments. An ablation experiment removes some part of the full model and measures the performance – this is useful to quantify how much different parts of the model help performance. Ablation experiments are an excellent way analyse your model.

Building strong baselines is very important. Too often, researchers and practitioners fall into the trap of making baselines that are too weak or failing to define any baselines at all. In this case, we cannot tell whether our complex NLP systems are adding any value at all.

Sometimes, strong baselines perform much better than you expected, and this is important to know.

3.5 Training and debugging NLP models

Unfortunately, NLP models can be difficult to debug. However, here are some tips:

- To debug a model, train on a small toy dataset (e.g., small fraction of training data, or a hand-created toy dataset) to sanity-check and diagnose bugs. For example, if your model is unable to overfit (e.g. achieve near-zero training loss) on this toy dataset, then you probably have a bug in your implementation.
- Hyper-parameters often impact results significantly. Use performance on the development set to tune these parameters. Though you probably won't have time for a very exhaustive hyperparameter search, try to identify the most sensitive/important hyperparameters and tune those.
- Due to their power, NLP models overfit easily. Regularisation and stopping criteria based on the development set (e.g. early stopping) are extremely important for ensuring your model will do well on new unseen data samples.
- During training, randomize the order of training samples, or at least remove obvious ordering. SGD relies on the assumption that the samples come in random order.

3.6 Evaluation

In your project, carrying out meaningful evaluation is as important as designing and building your NLP models. Meaningful evaluation means that you should carefully compare the performance of your methods using appropriate evaluation metrics.

Choosing evaluation metrics

You must have at least one evaluation metric (which should be a numerical metric that can be automatically computed) to measure the performance of your methods. If there is existing published work on the same dataset and/or task/problem, you should use the same metric(s) as that work (though you can evaluate on additional metrics if you think it's useful).

Human evaluation

Human evaluation is often necessary in research areas that lack good, comprehensive automatic evaluation metrics. If you want to use human judgment as an evaluation metric, you are welcome to do so (though you may find it difficult to find the time and/or funding to collect many human evaluations). Collecting a small number of human judgments could be a valuable addition to your project, but you must have at least one automatic evaluation metric – even if it is an imperfect metric.

What to compare

You should use your evaluation metric(s) to (a) compare your model against previous work, (b) compare your model against your baselines, and (c) compare different versions of your model. When comparing against previous work, make sure to get the details right. If you calculate your evaluation metric differently to previous work, the numbers are not comparable!

Qualitative evaluation

So far, we discussed quantitative evaluation – numerical performance measures. Qualitative evaluation, or analysis, seeks to understand your system (how it works, when it succeeds and when it fails) by measuring or inspecting key characteristics or outputs of your model. You will be expected to include some qualitative evaluation in your final report. Here are some types of qualitative evaluation:

- A simple kind of qualitative evaluation is to include some examples (e.g. input and model output) in your report. However, don't just provide random examples without comment – find interesting examples that support your paper's overall arguments, and comment on them.[1]
- Error analysis is another important type of qualitative evaluation. Try to identify categories of errors.
- Break down the performance metric by some criteria. For example, if you think a model is especially bad at classifying speeches by non-democratic leaders, show that by plotting the F1 score as a function of a democracy index like Polity.
- Compare the performance of two systems beyond the single evaluation metric number. For example, what examples does your model get right that the baseline gets wrong, and vice versa? Can these examples be characterised by some quality? If so, substantiate that claim by measuring or plotting the quality.

If your method is successful, qualitative evaluation is important to understand the reason behind the numbers and identify areas for improvement. If your method is unsuccessful, qualitative evaluation is even more important to understand what went wrong.

[1]It can be useful to provide a true random sample of model outputs. This gives readers a true, non-cherry-picked qualitative overview of your model's output. If you wish to do this, make it clear that the selection is random, and comment on it.