CS4248 Project Final Report

A0239298L, A0250934E, A0239754R,
A0276597E, A0234622M, A0218236H
Group 42
Mentored by Tan Yong-Jia, Naaman
{e0773896,e0949099,e0774352,e1132330,
e0726622,e0544272}@u.nus.edu

Abstract

Neural machine translation (NMT) has become the dominant approach for automated translation between languages in recent years. In this project, we attempt various key techniques taught in the module for text processing and representations, namely text tokenisation, embeddings, similarity, and machine learning. In exploring various approaches for each of these optimisations, we attempt to investigate which approaches perform better across the whole dataset, and on specific examples such as longer/shorter sentences to highlight the strengths/weaknesses of each approach.

We evaluate our model's performance on standard English-Chinese translation benchmarks, specifically BLEU-4 and BERTScore. BLEU-4 looks at the alignment of n-grams, where n ranges from 1 to 4, emphasising on both translation precision and fluency. Complementarily, BERTScore offers insights to paraphrased translations by computing similarity with contextualised token embeddings.

Our experimental analysis revealed that TER scores consistently mirrored the trends observed in BLEU-4[1] and BERTScore[2] evaluations. In contrast, ROUGE[3] and CHRF[4] metrics exhibited limitations in the context of our translations, particularly when dealing with repetitive characters. ROUGE does not penalise repetition unless it impacts recall and precision relative to the reference, while CHRF, emphasising character n-gram precision and recall, can inflate scores despite translations having limited character variety. This analysis was based on the average scores from model predictions trained on just 10,000 rows from the training set. Recognizing that ROUGE, TER, CHRF and CHRF++ have their merits, we opted to exclude them for clarity and explainability purposes.

The results of this work confirm the effectiveness of our proposed techniques for building high-performing NMT systems capable of translating between languages such as English and Chinese. We believe that, if given sufficient compute resources and training data, a state-of-the-art (SOTA) model can be implemented using our best result implementations of tokenisation, model architecture and text embedding representations to achieve good performance in the English-Chinese translation task.

1 Introduction

2 Related Work

3 Corpus Analysis

4 Experiments

To determine which methods for each subdomain produce the best translation results, we investigate the following research questions, and elaborate on the results in Section 5.

RQ-1: Between pre-trained word tokenisation algorithms, byte-pair encoding (BPE), and WordPiece models, which approach gives the best tokenisation performance of the dataset to give the best downstream translation result?

RQ-2: Would adding an alignment layer (slightly similar to an attention mechanism) to an RNN allow it to outperform LSTM and Transformer architectures?

RQ-3: Using context-based embeddings on a trained BERT embedding model, or short-context embeddings like FastText, would they improve the model's output over a learned embedding layer?

[1]

[2]

[3]

[4]

4.1 Tokenisation Experiments

For all the attempted tokenisation algorithms, we encode the split tokens to their index in the learned vocabulary, trained by tokenising the train dataset using the chosen tokenisation algorithm.

Type	Config.	Description
Word-based	1	Self-trained tokeniser based on word-based approach
BPE	2	Self-trained tokeniser based on SentencePiece BPE (for both languages)
WordPiece	3	Adapted tokeniser from pretrained models

Table 1: Tokenisation Configurations and their descriptions

We utilise the entire training set (231,266 rows) to generate the vocabulary for all these tokenisation configurations.

For configuration 1, we used spaCy's en_core_web_lg tokeniser component for English and stanza for Chinese. The vocabulary size from word-based tokenisation was 62,716 words/characters for English, and 101,686 words/characters for Chinese.

For configuration 2, given parameter size constraints for other model architectures, we limited the vocabulary size for our BPE tokenisers to 16,384 symbols per language, including Unknown, Padding, EOS and BOS tokens.

For configuration 3, we adapted popular pretrained tokenisers on our dataset. We chose to work with the models from google-bert/bert-base-uncased and google-bert/bert-base-chinese for English and Chinese respectively. The original vocabulary sizes for bert-base-chinese and bert-base-uncased was 21,128 and 30,522 respectively. We set the limit for the new vocabulary size to 32,000 for both languages to reduce resource utilisation for our models.

We take the reference RNN-alignment model architecture from the following segment as the control for this experiment setup, varying only the input tokeniser and modifying the model's embedding layer size to fit the vocabulary of the various tokenisers. We also restrict its training to the first 10,000 rows of the train set, and report the results on the test set as follows:

Evaluation Metrics	**Tokenisation Config.**		
	1	**2**	**3**
BLEU-4	4.08	2.41	0.10
BERT-R	0.54	0.51	0.44
BERT-P	0.56	0.55	0.54
BERT-F	0.55	0.53	0.49

Table 2: Evaluation scores from baseline RNN with different tokenisation configurations, trained with subset of 10k rows from training set

4.2 Model Architecture Experiments

We utilised the baseline RNN with alignment mechanism from the previous section and compared it with reference LSTM architectures, and the baseline Transformer architecture. For all the model architectures, we train them on the entire training set for a maximum of 30 epochs, allowing for early stopping.

For this approach, we used the second best tokeniser (SentencePiece BPE) due to its smaller vocabulary size, shrinking the model architecture memory footprint to fit on our training device, a NVIDIA L4 GPU on Google Colab.

As the transformer model is relatively deeper and more complex, we opt to use beam search with a beam width of 8 at each step to obtain the best translation output utilising the top-8 token probabilities at each step, as opposed to blindly taking the best token at each step with the greedy method.

4.2.1 LSTM

For the LSTM model, we

4.2.2 Transformer

For the Transformer model, we use the reference architecture from the paper Attention Is All You Need, reducing the inner layer dimension to 256 and the feed forward network dimension to 1024 to fit the model in memory.

We also utilised the implementation from https://github.com/devjwsong/transformer-translator-pytorch and adapted it for use with our tokenisers.

4.2.3 RNN Implementation

we referenced the paper 'Neural Machine Translation by Jointly Learning to Align and Translate'

(Bahdanau et al., 2014) to implement a more advanced RNN variant with alignment mechanism, which allowed the model to focus on only relevant parts of the input, addressing limitations observed with long sentences in basic encoder-decoder setups.

The implemented RNN variant uses a bidirectional RNN for input sentence processing, resulting in a set of annotations h_i encompassing contextual data from both directions. These annotations are derived from the concatenated forward $\overrightarrow{h_i}$ and backward $\overleftarrow{h_j}$ hidden states of each word in the input sentence:

$$h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$$

The encoder's output and the hidden state of the decoder s_{t-1} is then fed into an attention mechanism, which computes an attention vector a_t by determining the weights of each annotation in the context of the current target word being predicted.

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$

where e_{tj} is a compatibility score between the decoder's previous hidden state and the j-th annotation from the encoder, defined by:

$$e_{tj} = a(s_{t-1}, h_j)$$

A context vector c_t is then created for each word in the target sequence as a weighted aggregate of these annotations, informed by their respective attention weights:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

The weighted context vectors are then utilised by the decoder to generate the translated words sequentially. With functions f and g representing the RNN decoder and output layer respectively, s_t denoted the dynamic state of the decoder at each stage of the translation process, and y_t denotes the generated output word at each time step, building up the translated sequence one word at a time.

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$
$$y_t = g(s_t, y_{t-1}, c_t)$$

The model achieved its lowest validation loss after 8 epochs, with a learning rate of 1e-3.

4.2.4 Results

Training the models on the entire training set, we obtain the following results:

Evaluation Metrics	RNN	LSTM	Transformer
BLEU-4	9.74	0.70	13.78
BERT-R	0.64	0.49	0.71
BERT-P	0.69	0.44	0.75
BERT-F	0.67	0.46	0.73

Table 3: Evaluation scores from selected architectures with trained BPE tokeniser (config 2.) , trained on full training set

Binning the individual scores by length and averaging them by bin, we get the following results:

Sentence	BLEU		
Length (EN)	**RNN**	**LSTM**	**Transformer**
0-20	12.30	1.18	15.27
21-30	7.60	0.07	12.36
31-40	5.84	0.06	11.76
41-50	5.35	0.06	11.46
51-60	4.50	0.07	10.78
60+	3.31	0.07	9.71

Table 4: Average BLEU Scores by Sentence Length

Sentence	BERTSCORE — F1		
Length (EN)	**RNN**	**LSTM**	**Transformer**
0-20	0.69	0.49	0.73
21-30	0.65	0.43	0.73
31-40	0.63	0.42	0.73
41-50	0.62	0.43	0.73
51-60	0.61	0.43	0.72
60+	0.58	0.43	0.71

Table 5: Average BERTSCORE (F1) by sentence length

4.3 Embedding Experiments

In exploring the effects of various embedding representations on translation performance, we chose our Transformer model as the control as it performed the best, and swapped out its embedding layer for pre-trained embeddings, using BERT trained on our train set, as well as FastText embeddings trained on our train set.

5 Discussion

6 Conclusion

7 Report Best Practices

Your group report is **strictly** limited to eight pages[1] and summarises all of your group's understanding of your problem, models, experimental results and insights. Typically, such (empirical/experimental) scientific reports for natural language processing follows a five- to six-section format (suggested length in parentheses, for a **eight**-page limit; do not feel compelled to match these suggested lengths), as follows:

1. **Introduction** (1/2–1 page): Motivate your work, state the problem statement clearly (inputs and outputs), and summarise your key contributions. Optional to include are a concluding textual navigation paragraph, and/or running (microanalysis) example.

2. **Related Work / Background** (1/2–1 1/2 pages): Implementation and experimentation that your group did in the project should be based on others' experiences. Relate these relevant works to show that you are aware of best practices, and how your work builds upon them. Your report can call attention to gaps in the related work to motivate your work to as innovative and filling in knowledge that is lacking. This section typically features many citations to prior work and footnote references[2] to online datasets or software.

3. **Corpus Analysis & Method** (1–2 pages): Describe the different key useful approaches that yielded interesting findings here. You need not include all of your group's work if certain branches did not prove useful; those you can mention in an appendix. Describe the preprocessing, data collection, and the main methods used. You may also refer to your group's software repository in a footnote, if you host it online[3].

4. **Experiments** (1–2 pages): This section gives the main experimental settings, such as the corpus used, (macroscopic) evaluations metrics and baselines first; then proceeds to show

[1] For the main body of the report: title, abstract and main sections. Backmatter does not count towards this limit.

[2] Such as this one: `http://nlpprogress.com/`.

[3] An example would be a GitHub repository such as `https://github.com/knmnyn/cs4248-2120`. If you provide one, please ensure that you have at least a minimally-documented `README.md` file and organize your repository accordingly

the key performance evaluation experimental results, usually through figures, tables or charts. Interpret the data in these artefacts as prose explanations in the body text.

5. **Discussion** (1–3 pages): Enumerate 2–3 specific research questions and your group's answer that give more depth and analysis to the main results. These can describe performance aspects to sub-populations of input or intended users; time, memory and compute costs and scaling; micro-analysis of specific input instances. At least one question should be related to the natural language aspect (in contrast to general machine learning) of your project and corpora.

6. **Conclusion**: (1/4–1/2 page): This section is often abused as another chance to repeat the abstract or the introduction. Use this section instead to help summarize and lend insight to the reader, in light that they now have read the contents of the other sections. Limitations of your project, future directions also feature in this final section.

Aside from these sections, there will be a short 100–200 word abstract at the beginning of the containing a summary of the work accomplished. The abstract usually contains a clear task statement, highlights of experiments and key findings of the work.

Following the eight-page maximum length report body, there is unlimited space for you to include materials — an ethical statement, references and appendices (in that order).

It is recommended that you start with this format and permute it to your liking.

8 Marking Rubric

The marking of the project report follows a similar format to the *Intermediate Update*: Presentation, Content, and Miscellaneous. We will mark out of a total maximum mark of 100. Your grade and comments on the marking will be made available by Canvas Gradebook.

Report Presentation (25%):

- Motivation:
  - Does the report clearly outline of goals and questions addressed?
  - Is the motivation for your task clear, plausible and rational?
  - Is the problem statement well-defined using appropriate NLP terminology?
  - Does the report state the importance, usefulness, benefits of the work and the results?
- Structure
  - Does the report content flow logically?
  - Is it sufficiently well-organized to omit information that should be common knowledge to your peers?
  - Do you relegate less important information to an appropriate location (backmatter, software repository, footnote)?
- Visualization
  - Does the report use appropriate figures, plots, and tables to justify preprocessing steps, design decisions, motivating discussions and explanations?
- Presentation (more important)
  - Does the prose, references, sections and visuals all complement each other in describing the logical flow?
  - Is any corpus analysis (exploratory data analysis) done purposefully, to motivate model or experimental design?
  - Are any visuals appropriately-sized, captioned and legible? Do they serve to better explain the material than an equivalently-sized block of prose text?
  - Do you correctly follow the formatting instructions, length limitations and submission rules?

Do not just report numbers, but illustrate (with figures, tables), and explain them. Do not assume that your audience knows what your numbers mean.

Report Content (60%):

- Originality
  - What are the original elements done in the project? (It's not necessary that no group has done your task before, but your report needs to reflect your ability to think analytically and contribute novel analysis.)
  - Do you articulate how your work is novel in light of the prior work?

- Relevance
  - How strongly connected is the project to this course?
  - Do you use core concepts of NLP taught from class?
- Related Work
  - How strongly connected is the project to this course?
  - Do you use core concepts of NLP taught from class?
  - Do you present a study of related work to the task? (Formal academic references, useful web articles and posts material, and other related work should be considered in this aspect. Remember to cite explicitly.)
- Technical Justification (more important):
  - Is your technical approach suitable to try to solve your proposed problem?
  - Is your technical approach valid for your task and dataset?
  - Are there technical flaws in the execution of the approach?
  - Do you describe the data / corpora that you collected in an appropriate manner? (Self-annotated data may need evidence that the annotations are replicable; i.e., interannotator agreement)
  - Are evaluations performed with the appropriate metrics and correctly interpreted?
- Implementation (more important):
  - Did you implement multiple models (baseline, and best)?
  - Do you cleanly delineate what your group members coded as original work from public library or code repositories you used from others?
  - Did you implement or use the models correctly?
  - Did you tune them appropriately, where resources allowed?
- Model Evaluation (more important):
  - Do you address both macroscopic, dataset-wide level performance (e.g., F1 measures) as well as microscopic, individual instance level performance (careful error analysis with diagnosis)?
  - Do you demonstrate improvement in performance from your model to another, such as a baseline model? (A baseline model may be an implementation of a simpler model or version of your model, or referenced from other literature — make sure to give appropriate citations).

  Note that your performance need not be very high (e.g., 90%) if your data problem is hard. But you should show improvement over some baseline approach. This includes conscientious efforts to improve performance.

- Results Interpretation (10%): How well are the evaluation results described and interpreted.
  - Error analysis: Explain, with evidence, why the model may be performing poorly (or not as good as you wish).
  - Do you justify technically why your model is good or has improved? I.e., rationalize your approach's performance effectiveness.
  - Future improvements: Discuss how you may further improve your model.

  You do not have to implement or test all your ideas, if too infeasible. Though discussing them helps to show your grading staff that you have good and valid ideas.

Miscellaneous (15%):

- Reproducibility
  - Is the technical approach described clearly and sufficiently detailed for a peer to replicate? Is the evaluation method described clearly and detailed enough for a peer to replicate?
  - Is your source code well-organized and any ancillary materials well documented?
  - Are your results easy to replicate by running documented commands or executing a notebook?
- Limitations

- Do you state the principal limitations of your work, such as the important aspects of the problem domain, and how these factors might be mitigated?

- Backmatter
  - Do you use the backmatter and supplemental materials (website, source code repository) effectively to complement the formal report body?
  - Are your references bibliographically complete?
  - Did your group appropriately fill out the *Statement of Independent Work*?
  - Did you properly acknowledge and document how AI tools played an appropriate role in your experimentation, coding and report?

Late policy. Please refer to the CS4248 website for late policy, in Canvas » Pages » Grading. In general, our course's late policy is harsh to help our instructing staff mark in an efficient manner. To ensure your group does well, please turn in your report on time (.PDF version to the Canvas Assignment) by the deadline. If you envision that your group cannot meet the deadline and you wish to seek an extension, please do so well in advance of the deadline, and not after the deadline.

9 Report Formatting

This document is a LaTeX sourced document, a common typesetting system used in research communities, and commonly used for many conferences for natural language processing research. As such we are using this typesetting system and the formatting (style) files common to best practices for communicating NLP research and outcomes. This template is hosted on the third-party cloud-based LaTeX typesetting site, Overleaf, which you may decide to use if you'd like. Note that NUS has a sitewide license to this product so you can get professional features when using an NUS email address (e.g., ones ending in `u.nus.edu` for free.

To be clear, it is **not necessary** for your group to use LaTeX to typeset your final report. You may use other software and reproduce (most of)[4] the template following the styles noted below.

1. If you are using a system other than LaTeX (e.g., MS Word, Open Office, LibreOffice, Apple Pages), you will want to follow the guidelines in the rest of this section for the report format.

2. If you are using Overleaf/LaTeX, you should simply be able to use the logical formatting tags directly — which should have been configured properly by the inclusion of the `acl.sty` and `acl_natbib.bst` style files — and ignore the rest of this section. Please see the LaTeX source of this document for comments on other packages that may be useful.

Format your report to two columns to a page, A4 sized page only, in the manner these instructions are formatted. The exact dimensions for a page on A4 paper are:

- All (Left, right, top and bottom) margins: 2.5 cm

- Column width: 7.7 cm

- Column height: 24.7 cm

- Gap between columns: 0.6 cm

For reasons of uniformity, Adobe's **Times Roman** font should be used.

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word "Abstract"	12 pt	bold
section titles	12 pt	bold
document text	11 pt	
captions	11 pt	
abstract text	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 6: Final Report Font guide. Captions generally should be self-sufficient to read the table or figure independently of the prose text of the report. Use bottom and top rulelines for proper formatting.

9.1 The First Page

Center the title, author's name(s) and affiliation(s) across both columns. Do not use footnotes for

[4] In other software, it may be difficult to reproduce the margin line numbering, this is ok to omit.

affiliations. Use the two-column format only when you begin the abstract.

Title: Place the title centered at the top of the first page, in a 15-point bold font. (For a complete guide to font sizes and styles, see Table 6) Long titles should be typed on two lines without a blank line intervening. Approximately, put the title at 2.5 cm from the top of the page, followed by a blank line, then each of your groupmates' NUS student IDs, Replace the `XX` and `YY` placeholders with your two-digit Group ID (e.g., "01") and project mentor's name. Do not format title and section headings in all capitals as well except for proper names (such as "BLEU") that are conventionally in all capitals. The affiliation should an electronic mail address for at least one contact student.

Start the body of the first page 7.5 cm from the top of the page.

Abstract: Type the abstract at the beginning of the first column. The width of the abstract text should be smaller than the width of the columns for the text in the body of the paper by about 0.6 cm on each side. Center the word **Abstract** in a 12 point bold font above the body of the abstract. The abstract should be a concise summary of the general thesis and conclusions of the paper. It should be no longer than 200 words. The abstract text should be in 10 point font.

Text: Begin typing the main body of the text immediately after the abstract, observing the two-column format as shown in the present document. Do not include page numbers.

Indent when starting a new paragraph. Use 11 points for text and subsection headings, 12 points for section headings and 15 points for the title.

9.2 Sections

Headings: Type and label section and subsection headings in the style shown on the present document. Use numbered sections (Arabic numerals) in order to facilitate cross references. Number subsections with the section number and the subsection number separated by a dot, in Arabic numerals. Do not number subsubsections.

9.3 Citations

Citations are bibliographic references to scholarly works. You may include references to documentary web pages, blog posts, reports, pre-prints as well. Note that references to general webpages or software (packages) as URLs are more appropriately given as footnotes.

Output	**natbib command**
(?)	`\citep`
?	`\citealp`
?	`\citet`
(?)	`\citeyearpar`

Table 7: Citation commands supported by the `acl.sty` style file. The style is based on the natbib package and supports all natbib citation commands.

Table 7 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations, like this citation to a paper by **?**. You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations (**?**). You can use the command `\citealp` (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. **?**).

9.4 Backmatter

Backmatter are other materials that follow the main report body. They include the following:

References. Your group should cite all appropriate references that you need in your report. You may place an *unlimited* number of references to work that is relevant, beyond the page limit for the main report.

The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named `custom.bib`, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliographystyle{acl_natbib}
\bibliography{custom}
```

Many papers in natural language processing come from the ACL Anthology, the digital library for NLP, which Min ran for many years. You can obtain the complete ACL Anthology as a BibTeX file from https://aclweb.org/anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliographystyle{acl_natbib}
\bibliography{anthology,custom}
```

Statement of Independent Work. Your group must include this section, to declare whether your group followed class policy. Refer to the example in this document's backmatter.

Ethical Statement. An optional, unnumbered section. Refer to the example in this document's backmatter.

Acknowledgements. An optional, unnumbered section. Refer to the example in this document's backmatter.

Appendices. You are also allowed *unlimited pages* for appendices, but be aware that your teaching staff is not obligated to read or acknowledge these sources.

Use `\appendix` before any appendix section to switch the section numbering over to letters. See Appendix A for an example.

Acknowledgements

Place any acknowledgements here. You can thank any people you contacted or sources that you used that are not bibliographic in nature.

This document has been adapted from the ACL Rolling Review Template (ACL ARR) by Min-Yen Kan. You may find the original template, which NUS has also contributed to in the past, here: https://www.overleaf.com/latex/templates/acl-rolling-review-template/jxbhdzhmcpdm. We have omitted much of the original document to cut down on verbiage.

Our final report grading rubric is based on a merger of guidelines from component courses CS5228 Knowledge Discovery and Data Mining and CS3244 Machine Learning.

Statement of Independent Work

You **must** *include the text of the two statements below in your group's submitted work. Digitally sign your submission using your Student Numbers (starting with A...; N.B., not your NUSNET email identifier). This is a required section and is not part of the main body (doesn't count towards your page limit).*

1A. Declaration of Original Work. By entering our Student IDs below, we certify that we completed our assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, we are allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify our answers as per the class policy.

We have documented our use of AI tools (if applicable) in a following table, as suggested in the NUS AI Tools policy[5]. This particular document did not use any AI Tools to proofcheck and was constructed and edited purely by manual work.

If the production of your report used AI Tools (inclusive of Generative AI), do keep detailed logs of how you used AI Tools, as your project requires the accountability of an audit trail of your interaction(s) with such tools (prompts, output).

1B. Exception to the Class Policy. We did not follow the CS4248 Class Policy in doing this assignment. This text explains why and how we believe we should be assessed for this assignment given the circumstances explained.

Signed, [Enter your Axxx Student IDs and NUSNET email addresses here]

Ethical Statement

The optional ethical statement is an unnumbered section that comes after the references. Most projects may not need to include such a statement, but we include it here, as it is important to be aware that NLP research and experimentation needs to be conducted in an ethically acceptable manner.

It describes any pertinent issues with respect to the NL technology being described in the project work. These could include dual-use, data quality discussions, compute requirements, fair pay for annotators and evaluators, among other factors.

You may read more about these issues by reading the *Guidelines for Responsible NLP Research*[6] and consulting works on the ACL Ethics Reading List[7].

A Example Appendix

Optional appendices are the last item in the report.

If your group's report is too long, working to best structure the core of the report (instead of technical details) in the main body and relegating details for replication in appropriate appendices is key.

Since you have unlimited pages for appendices, you can afford to make any plots or result tables larger in the appendices, but do ensure that key results are in the report's main page limit rather than relegated here.

B Project Frequently Asked Questions (FAQ)

This section is sourced from the CS3244 Machine Learning module's Project FAQ.

1. *Q: Just to be sure, for our project work, can we code in any language (i.e. R) other than python?*
   A: Yes.

[5] https://libguides.nus.edu.sg/new2nus/acadintegrity, tab "AI Tools: Guidelines on Use in Academic Work"

[6] https://aclrollingreview.org/responsibleNLPresearch/

[7] https://github.com/acl-org/ethics-reading-list

2. *Q: Does the project difficulty matters for the grading, e.g. taking the* `easy` *dataset versus doing something tagged as* `hard`*? Would we be graded based on the novelty or "importance" of the use case/problem our group comes up with?*
   A: We aren't looking at technical complexity when grading the project. We state the notional technical difficulty of project dataset to help your team decide which type of project to take on. More difficult datasets usually involve more specific preprocessing, data normalization and usually (much) larger compute costs in manipulating large-scale data.

   - What we are looking at is the learning that comes out of engaging in the project. We want to see you twist your mind and come up with interesting approaches to the problem of choice.
   - We also do not place heavy emphasis on metrics like accuracy, log loss, precision, recall, etc.. We'd care more about questions like "why did you use XYZ Metric over ABC Metric for this problem?". Getting a +0.5 accuracy boost doesn't matter as much as why you chose to do ABC Technique that brought about that accuracy boost in the first place.
   - We care more about how you communicate your findings to us in an interesting way like your analysis, your wins, your losses (pun not intended), etc.. That way, it shows us that you gained valuable experience from this project that you can apply to future projects.
   - You are allowed to explore models not covered in class at your own discretion. We only teach you the fundamentals in hopes of making you comfortable with the math/concepts involved. Beyond that, you can look at more complex models and techniques not taught in CS4248 for your projects. But again, complexity is not the focus, communication and understanding the 2W1H (why, what, how) are.
   - There isn't any true "novelty" in these projects *per se*. They are popular benchmarks found in the real world with increasing difficulty of use. We want you to have your own unique spin to these solutions (please do not copy-paste/plagiarise someone else's code from online) and present them in a way you and we (i.e. the teaching staff) understand.

   These projects are for you in the long run, not us. Hope this helps.

3. *Q: What local compute do we have access to for our projects?*
   A: Please take note that our class' reservation for compute nodes in SoC has now taken effect (from 26 Jan until 15 Apr). If your groups find it useful, you may start using it if you have previously registered an SoC UNIX ID. The nodes you may work with are `xgpf0-6` (i.e., use the command `ssh xgpf0.comp.nus.edu.sg` within SoC's network to reach the first of seven available servers). You may use other nodes but these compute resources are exclusively for our class' use. If you use these resources, please self-regulate and use a maximum of one (1) node per group. It is unfair if one group hogs all of the resources and makes the resources unavailable to others. Please be respectful and mindful that your group is one of many in our cohort and all groups should be able to utilize some of these resources.

4. *Q. Have any advice for experimentation?*
   A: Sure. We recommend staging your experiments' time to execute to fit your working style. For example, we recommend having 3 granularities of time for your experiment execution: immediate (finishes within 1–3 minutes), coffee/tea break (finishes within 30-60 mins), overnight (as the name implies). Based on some initial runs, you should develop a good estimation for how long your pipeline takes for a certain data scale, and retrofit/sample data from your dataset to fit accordingly.

   Immediate experiments should just to check that your code works with a tiny toy dataset without faults and to assess whether an experimental setting can be escalated to the next granularity.

   Coffee / Tea Break experiments test those runs from the Immediate scale that reach your standard for trying on a larger dataset. These experiments can be set to run on a server with a medium-sized dataset that can complete independently while you are eating a meal, or taking a break to do other work or play. These validate your ideas on larger scale datasets without committing to training an entire dataset without knowing whether the results point appropriately in the proper direction – Min has seen many times that the "math works out (e.g., shape of matrices are fine) but which the computation is garbage (e.g., one off indexing errors) – so this scale mitigates this. These scale experiments need to be followed up with analysis to ensure that the results are as expected and appropriate.

   Overnight (or longer) experimentation runs your training or testing at scale, for production or for final presentations or reports. Try not to do this scale of experimentation without having a good reason to believe it will succeed (i.e., don't run a large-scale experiment to try something out; you should have done that at the Coffee / Tea Break scale instead).

5. *Q: Is model performance and using and getting state-of-the-art performance an important output of our project?*
   A: Generally, no. Good projects explain and teach, rather than just show good results. It is better to use simpler models where you can show that your understanding of the model, features, corpus and evaluation metrics interact to lead to the performance levels you observe. Merely swapping in a newer more advanced model and getting better results doesn't merit this understanding. Good and replicable results are nice-to-have in terms of grading criteria, but not must-haves.

6. *Q: Just want to check if our group's understanding of an ablation study is correct. If we have around 20 features that we engineered and a baseline model of previously proposed features, how do we efficiently conduct an ablation study? We are thinking of grouping the 20 features we have into a few groups to turn them on/off for the study. Is this the correct way? Should our angle start out from all features included to removing parts of it, or start from the baseline model and add features, since it's an "ablation" study.*
   A: Yes, that would be appropriate. You can turn off a feature group, or some pre-/post- processing and see what the effects are. Before you do that, your team should hypothesize what you think would happen. This can help you hone your sense of understanding. A scientist does experimentation with a hypothesis in mind. Generally you have a final model and you turn off certain features to study their (negative) impact on your final model. This allows you to argue for the necessity of all feature groups in your model.

7. *Q: I have some questions regarding the ablation studies we need to do for the project. Based on my understanding, the aim of ablation studies is to understand how our existing system (i.e. the feature engineering techniques+the models) work. And I remember you saying that it is about having hypothesis and finding ways to test them. So my first question is, do we have to have some form of "removal" to conduct ablation analysis? Or is it ok to analyse without "removal"?*
   A: Yes that's correct. No you need not (always) do a removal (ablation) for analysis. It is just a common form. Both additive and ablative (removal) studies of feature classes are common. Ablation studies often form the basis for arguing that the model cannot have any of its elements removed without damaging a performance metric.

8. *Q: If the aim is to understand how our existing system works, how do we control the factors of the experiment? For example, if feature engineering technique A works well with a model, and we wanna know what is in A that makes this succeed, so we modify technique A into its variant C (with one feature removed), and then here's the question: do we feed the input processed using C into the model trained using A and see the change of results, or do we train a new model of the same structure on this input processed using C and then compare the results with inputs processed by A feeding into model trained using A?*
   A: Yes, sometimes we refer to C as "C: Model A–<some feature>". The second method, train a new model. We compare the performance macroscopically (whole dataset, e.g., accuracy, F_1) against the system trained with the output from A.

9. *Q: It was mentioned at the start of the sem that there was no real novelty in our projects, but there was novelty of project in the project presentation rubrics. Could you please clarify what novelty in the marking rubric truly stands for?.*
   A: Indeed with many of our curated datasets, there has been much (informally) published past work. You should find, read and cite any past work in your prerecorded presentation and add these links to the supplemental materials you prepare, so that the instruction staff can check accordingly. You should validate (by replication) performance figures from other papers or posts. You do original work by going beyond what others have reported. There are many ways you can go beyond the past work in your analyses and subsequent iterative questioning and answering of your project work. Don't concentrate just on performance metrics but look for ways to connect what you've learned in lecture with your project.

   Examples include:

   - How do changes in your model architecture affect performance? Not just at the macro performance metrics but for individual (micro) and groups (meso) problem instances?
   - How does changing some input instances minimally change the results for better or worse?
   - Which features or model paradigm designs contribute the most towards performance and, more crucially, why?
   - Why do certain models do better at certain instances and not for others?

C Version History

Note: This list is in reverse chronological order. Timestamps follow the *yymmdd* format.

1. 240229 — Updated to as for audit trail regarding AI Tools Use (around Lines 480–485 in the PDF).
2. 230312 — Updated for Canvas and AI Tools policy.
3. 220324 — Updated report length and data analysis sections.
4. 220320 — Initial distribution.