COMP6713 Natural Language Processing: 2025 Term 1 Individual Assignment

1 March 2025

Please read the complete document carefully.

This document contains the problem statement for the individual assignment that forms an assessment for COMP6713 in 2025 Term 1. If you have queries about the technical content of the assignment, please email Aditya Joshi at `aditya.joshi@unsw.edu.au`.

USE OF GENERATIVE AI TOOLS IS DISALLOWED.
UNSW PLAGIARISM POLICY FOR INDIVIDUAL ASSIGNMENTS APPLIES.
TEST CASES WILL EXPLICITLY CHECK FOR COPIED CODE.

Deadline: Friday, 21 March 2025, 9:00 PM Sydney Time

The individual assignment carries a maximum of 25 marks. It contributes to 10% of the assessment in the course.

The assignment requires you to write executable code and a report. Upon completing the assignment, you will learn to apply topics covered in weeks 1–3, including but not limited to: (a) Off-the-shelf NLP libraries; (b) Probabilistic language modeling with smoothing; and (c) Attention. Evidently, one of the learning outcomes of COMP6713 is being able to connect ideas from the past and the present of NLP.

I. Getting Started

Language models have indeed made several tasks easier than ever. You can now type "Write a poem about kangaroos in the style of William Wordsworth" in the ChatGPT tool, and it generates exactly that for you. How easy is that! I wonder what challenges a probabilistic poetry generator might face. Ah, implementing one will help me understand attention in Transformer much better!

You are provided with:

1. **Code Template**: a boilerplate code to get started. DO NOT modify the main execution block. The filename is `boilerplate.py`
2. **Training Data:** a file called `poetry_dataset.tsv` containing English language poetry dataset by Google Research.

Goal: To build a poetry generator that takes as input: (A) a start word, (B) a rhyme scheme, (C) Number of words per line, and produces a poem of exactly 4 lines based on the input arguments.

```
class ProbLangModel:
def generate_poetry(self, sentence, num_words_per_line=10, rhyme="ABAB"):
    poem = ""
    ...
    ...
    ...
    ....
    print(poem)
```

Every call to generate_poetry() must complete in 60 seconds.

Expected Output: A poem consisting of exactly four lines separated by "\n" that adhere to the input requirements given below. The poems must consist of tokens separated by space. For example, ensure that your output is not `Where are you?` but `Where are you ?`. (Note the space before the question mark).

Input Parameters: (All parameters are essential. Include default values as stated above.)

sentence: String. Must be a word seen in the training set.

num_words_per_line: Integer. The number of words per each line in the poem

rhyme_scheme: Either "ABAB" or "AABB". **ABAB** means that the first and the third line rhyme at the end, while the second and the fourth line rhyme at the end. **AABB** means that the first and the second line rhyme at the end, while the third and the fourth line rhyme at the end. To read more about rhyme schemes, see: `https://en.wikipedia.org/wiki/Rhyme_scheme`.

Example Output:

```
python solution.py -s 5 -w "and when they" -n 10 -r "ABAB"
and when they pilgrim shine spoiled wars sharpness vacant waste
prophets trunk briton leaf dwindle mountain notes adore dews unspoken
sleeping cloudy spotted chair peep call's ) thro deeps taste
worship throng feared who's mornin' repose de warlike river broken
```

```
python solution.py -s 5 -w "and when they" -n 10 -r "AABB"
and when they pilgrim shine spoiled wars sharpness vacant waste
prophets trunk briton leaf dwindle mountain notes adore dews taste
unspoken sleeping cloudy spotted chair peep call's ) thro through
fearless who's mornin' repose de warlike river willis tranquil flew
```

You are strongly recommended to use Python to write your code, and any of the libraries discussed in the class.

II. While you work on the assignment

Preprocessing

You may discard the second column in the training data – it corresponds to the sentiment. When tokenizing from the dataset, you must remove all numbers. A token or a word is considered to be either a string of alphabetical letters `hello`, or a symbol `?`. Alphabetical letters interchanged by a single quotation mark, such as `where're` or `'Neal` or `O'er` or `James'` are considered one unit of

token or word.
Hints:

1. You have completed probabilistic language modeling in the tutorial in week 3. You may use your learning from there to complete the assignment.
2. You are encouraged to use the NLTK-based pronouncing python library (version 0.2.0) at `https://github.com/aparrish/pronouncingpy`. Example usage is as follows

```
import pronouncing
pronouncing.rhymes('stand')
['and', 'band', 'banned', 'bland', ...]
```

Report
You need to write a report, in addition to the code. The report must contain three parts:

- **Part 1**: Describe the technical details of your approach (Approx. word count: 200 words).
- **Part 2**: Run your code for a few example inputs. Include screenshots of at least two outputs **AND** add a few sentences (at least 2 each) discussing each of the outputs.
- **Part 3**: The approach that can be used to solve this assignment has (albeit minor) similarities with masked attention in Transformer. Write a paragraph on what similarities exist between YOUR specific implementation and masked attention in Transformer.

Deliverables

You will upload the individual assignment on the Moodle coursepage. Please upload a zip file containing:

- one single python file titled `zID.py`, where `zID` is your zID on Moodle.
- one PDF report titled `zID.pdf`, where `zID` is your zID on Moodle.

You can modify all code in the boilerplate except for the main block execution. Do not modify the input and output parameters of the `gen_poetry` instance function.

IV: Submitting your assignment

- Upload the zip file to Moodle before the deadline, unless you have a prior approved Special Consideration.
- Submissions will not be accepted on email. Do not email any course team member.

III. After you submit your assignment

The course team will run initial checks related to the course's plagiarism and Generative AI use policies. After it passes the checks, assignments will be marked using a combination of automatic and manual evaluation.

Automatic Evaluation [10 marks]

Your code will be tested against several combinations of input arguments over multiple runs. Marks will be assigned based on average accuracy over the runs, alongside the following questions:

- Does the program run without error and produce an output within the specified time for all test runs? → 1 mark
- Does your generated poem start with the required start word? → 1 mark
- Are the generated poems exactly four lines? → 1 mark
- Does each line have the exact number of tokens per line based on the argument? → 2 mark
- Does the generated poem follow the rhyme scheme? → 5 marks

Manual Evaluation [15 marks]

Code:

- 4 marks for correct implementation and incorporation of rhyme scheme
- 2 marks for correct probabilistic language modeling under other constraints
- 2 marks for style
- 2 marks for well-commented code, all comments must be in English

Report:

- 2 marks for Part 1
- 2 marks for Part 2
- 1 mark for Part 3

IV. Additional Points

1. Automatic evaluation will be run on a computer with internet access, and will follow the details outlined in this document.
2. You must **NOT** use any other training data than the one provided.
3. Please make and state all your assumptions in your report.
4. Your code must successfully run on our automatic scripts for you to receive the corresponding marks. Therefore, please ensure that you upload all files required to run the code.
5. The automatic evaluation does not check for exact match of outputs, as is evident from the questions.
6. You are required to complete the assignment on your own without assistance from AI tools or human peers.

Hope you enjoy working on the assignment! :)