



# FIT9136 Assignment 2

Semester 2 2020

Deep Mendha

Teaching Associate, Faculty of IT

Email: [deep.mendha@monash.edu](mailto:deep.mendha@monash.edu)

© 2020, Monash University

Assignment Structure by Shirin Ghaffarian Maghool

Date: 24 Aug 2020

© 2020, Monash University

# Table of Contents

1. Key Information.....	3
1.1. Learning outcomes.....	3
1.2. Do and Do NOT.....	3
1.3. Marking Criteria.....	3
1.4. Submission details.....	4
2. Getting help.....	5
2.1. English language skills.....	5
2.2. Study skills.....	5
2.3. Things are tough right now.....	5
2.4. Things in the unit don't make sense.....	5
2.5. I don't know what I need.....	5
3. Key tasks (100 marks).....	6
3.1. Digital Shakespeare: Pre-processing (30 marks).....	6
3.2. Zipf's Law: Common Statistics (40 marks).....	7
3.3. Bi-gram Model (15 Marks).....	8
3.4. Generate Statement (15 marks).....	9

# 1. Key Information

<b>Format:</b>	Individual
<b>Due date:</b>	19 <sup>th</sup> Oct '20 5:00 pm (AEST)
<b>Weight:</b>	25% of unit mark

## 1.1. Learning outcomes

1. design, construct, test and document Python programs;
2. demonstrate on advance topics of python, like classes, objects, visualization;
3. evaluate different algorithms and analyse their complexity;
4. translate problems into algorithms with appropriate implementations by investigating different strategies for the algorithm development

## 1.2. Do and Do NOT

Do	Do NOT
<ul style="list-style-type: none"><li>• Maintain academic integrity<sup>1</sup></li><li>• Get <a href="#">support</a> early from this unit and other services in the university</li><li>• Apply for special consideration for extensions<sup>2</sup></li></ul>	<ul style="list-style-type: none"><li>• Leave your assignment in draft mode</li><li>• Submit late (10% daily penalty applies)<sup>3</sup></li><li>• Submission is not accepted after 5 days of the due date, unless you have special consideration.</li></ul>

## 1.3. Marking Criteria

Your work will be marked on

Functionality	Correctly working program	<b>60%</b>
Code Architecture	Algorithms, data types, control structures and use of libraries	<b>10%</b>
Code Style	Variable names, readability, clear logic	<b>10%</b>
Documentation	Program comments, clarity and connection to code	<b>20%</b>

<sup>1</sup> <https://www.monash.edu/rlo/research-writing-assignments/referencing-and-academic-integrity/academic-integrity>

<sup>2</sup> <https://www.monash.edu/exams/changes/special-consideration>

<sup>3</sup> eg: original mark was 70/100, submitting 2 days late results in 56/100 (14 marks off). This includes weekends

## 1.4. Submission details

Submit to “Assignment 2 Submission” on moodle:

- A2\_studentID.zip<sup>4</sup>
- Make different A2\_QN\_StudentID.py file for the assignment 2.<sup>5</sup>
- Please make sure **do not** add dataset folder to the .zip file.

Containing each of the four (4) tasks covered in [Section 3. Key tasks \(100 marks\)](#)

<sup>4</sup> studentID is your student ID. E.g. if your Id was 12345678, you would submit “A2\_12345678.zip”

<sup>5</sup>StudentID is your student ID, and QN is question number. E.g. if your Id was 12345678, and you are submitting Q1, then you would submit “A2\_Q1\_12345678.py”

## **2. Getting help**

### **2.1. English language skills**

if you don't feel confident with your English.

- Talk to English Connect: <https://www.monash.edu/english-connect>

### **2.2. Study skills**

If you feel like you just don't have enough time to do everything you need to, maybe you just need a new approach

- Talk to a learning skills advisor: <https://www.monash.edu/library/skills/contacts>

### **2.3. Things are tough right now**

Everyone needs to talk to someone at some point in their life, no judgement here.

- Talk to a counsellor: <https://www.monash.edu/health/counselling/appointments>  
(friendly, approachable, confidential, free)

### **2.4. Things in the unit don't make sense**

Even if you're not quite sure what to ask about, if you're not sure you won't be alone, it's always better to ask.

- Ask in Ed: <https://lms.monash.edu/course/view.php?id=78169&section=4>
- Attend a consultation: <https://lms.monash.edu/course/view.php?id=78169#section-3>
- Email your tutor: <https://lms.monash.edu/course/view.php?id=78169&section=1>

### **2.5. I don't know what I need**

Everyone at Monash University is here to help you. If things are tough now they won't magically get better by themselves. Even if you don't exactly know, come and talk with us and we'll figure it out. We can either help you ourselves or at least point you in the right direction.

### 3. Key tasks (100 marks)

Language modelling is one the most interesting and important task in natural language processing (NLP). A language model lies at the core of machine translation systems such as google translate, summarisation systems, text completion systems such as auto-complete to name a few. In this assignment you will be writing a simple language model known as the bi-gram model. The tasks that follow will build the model step by step. Each task is attempting to access a particular aspect of programming. Read each section carefully and attempt to solve them using the techniques we have explored in class.

- **Libraries can be used:** **math, matplotlib, os, pandas, numpy.**
- Dataset is available on the assessment page, under [Assessment 2](#).

#### 3.1. Digital Shakespeare: Pre-processing (30 marks)

Our language model will be trained on Shakespeare's books. Your first task is to read in all the given books and **remove all special sentences from it**. The special sentences lies between **< and >**.

<b>Inputs</b>	<b>Files:</b> All the files in the dataset folders
<b>Output</b>	<b>Filename:</b> <i>cleaned.txt</i> Example in sample behaviour.
<b>Details</b>	Task: <ul style="list-style-type: none"><li>• Remove all special sentences (Special sentences lies between “&lt;” and “&gt;”).</li><li>• Remove all the characters that are <b>not alphanumeric</b> excepts space.</li></ul>
<b>Sample behaviour</b>	

```

< Shakespeare -- A MIDSUMMER-NIGHT'S DREAM >
< from Online Library of Liberty (http://oll.libertyfund.org) >
< Unicode .txt version by Mike Scott (http://www.lexically.net) >
< from "The Complete Works of William Shakespeare" >
< ed. with a glossary by W.J. Craig M.A. >
< (London: Oxford University Press, 1916) >
<STAGE DIR>
<Scene.--Athens, and a Wood near it.>
</STAGE DIR>

<ACT 1>

<SCENE 1>
<Athens. The Palace of Theseus.>
<STAGE DIR>
<Enter Theseus, Hippolyta, Philostrate, and Attendants.>
</STAGE DIR>
<THESEUS>      <1%>
    Now, fair Hippolyta, our nuptial hour
    Draws on apace: four happy days bring in
    Another moon; but O! methinks how slow
    This old moon wanes; she lingers my desires,
    Like to a step dame, or a dowager
    Long withering out a young man's revenue.
</THESEUS>

<HIPPOLYTA>    <1%>
    Four days will quickly steep themselves in night;
    Four nights will quickly dream away the time;
    And then the moon, like to a silver bow
    New-bent in heaven, shall behold the night
    Of our solemnities.
</HIPPOLYTA>

```

### 3.2. Zipf's Law: Common Statistics (40 marks)

Zipf's law of word distribution states that the frequency of every word in a large corpus is inversely proportional to its rank in the frequency table. Let  $f_1$  be the  $I^{th}$  largest frequency in the list that is  $f_1$  is the frequency of most common word,  $f_2$  is the frequency of second most common word and so on. Zipf's law states that  $f_1$  is approximately equal to  $\frac{a}{I}$  for some constant .

<b>Inputs</b>	<b>File:</b> <i>cleaned.txt</i>
<b>Output</b>	There are 3 outputs: <ol style="list-style-type: none"> <li><b>File:</b> vocab.txt (Using the file that was created in the previous task.)</li> <li>Graph of frequency against first 100 words in sorted vocab.</li> <li>Graph of words that occur n time against word occurrence.</li> </ol>
<b>Details</b>	There are 3 Task:

	<ol style="list-style-type: none"> <li>1. Finding all the unique word in the cleaned file. Save all the clean words along with their frequencies, and sort (in descending order), based on word's frequency in file <i>vocab.txt</i> <ol style="list-style-type: none"> <li>1. To get unique words convert all the text into lower case first and the find unique words.</li> </ol> </li> <li>2. Plot a graph of frequencies against first 100 words from the sorted vocab file.</li> <li>3. Count the number of words that occur once, twice , thrice till 250. Plot number of words that occur n times against word occurrence.</li> </ol>
--	--

### 3.3. Bi-gram Model (15 Marks)

A bi-gram language model is a probabilistic model where a sentence probability is decomposed into product of conditionals as follow:

$$p(x_1, x_2, x_3, \dots, x_n) = \prod p(x_k \vee x_{k-1}) \text{-----(equation 1)}$$

These probabilities are approximated from the corpus using the following equation.

$$p(x_k \vee x_{k-1}) = \frac{c(x_k, x_{k-1})}{c(x_{k-1})} \text{-----(equation 2)}$$

- where **c** stands for **count** of the **words occurring together in the corpus**.

For the **first 1000 words** in the **vocab.txt** file, fill in the following table.

	word 1	word 2	word 3	.....	word 1000
word 1	c(word1  word1)	c(word1  word2)	c(word1  word3)		c(word1  word1000)
word 2					
word 3					
.....					
word 1000					

If you look at the numbers in your table you will see a **lot of zeros**. This is called **data sparsity problem** and causes a major issue by pushing probabilities to zero. A simple fix is to **smooth out the probabilities** by adding one to each count. Then our **new equation of probability** becomes

$$p(x_k \vee x_{k-1}) = \frac{c(x_k, x_{k-1}) + 1}{c(x_{k-1}) + V} \text{-----(equation 3)}$$

- where **V** is the words in **vocabulary**.

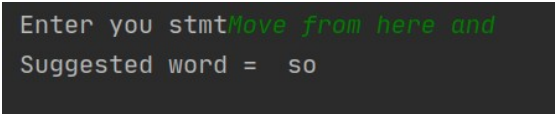
**Convert the counts into probabilities using equation 3 and save them** in a file called **model**. You can save it in a file extension of your choice.



The **data structure to store such a table is also up to you**. All your choices **must** be made within the **scope** of what you have **studied** in this **unit**. **Advance packages such as pickle, nltk, spacy and advance data structures such as heaps should not be used.**

### 3.4. Generate Statement (15 marks)

Given your model and a prompt, generate sentences of various lengths.

<b>Inputs</b>	<b>Input Sentences:</b> <ul style="list-style-type: none"> <li>• This is a</li> <li>• What is the purpose</li> <li>• Move From here and</li> </ul>
<b>Output</b>	<b>Suggested Word:</b> <ul style="list-style-type: none"> <li>• man</li> <li>• because</li> <li>• so</li> </ul>
<b>Details</b>	<p>Chose the word with the highest probability at each location based upon the model. This is called <b>greedy decoding or inference</b>.</p> <p>For example for prompt number 1, the probability can be decomposed as <math>p(\text{this})p(\text{is} \text{this})p(\text{a} \text{is})p(\text{.} \text{a})</math> where you chose word with the highest probability at <math>p(\text{.} \text{a})</math>.</p>
<b>Sample behaviour</b>	 <pre>Enter you stmtMove from here and Suggested word = so</pre>

#### Note:

- To test your model further, test dataset is available on the [Assessment page](#) on the Moodle.