**A link to download the third-party library if you used any in your assignment**

Part 1: Dataset (in report) <https://archive.org/details/stackexchange>

Part 2.1: Online text analyser (<https://www.online-utility.org/text/analyzer.jsp>) & Porter Stemmer Online was used (<http://9ol.es/porter_js_demo.html>).

Part 2.2 Import NLTK from python (www.nltk.org/install.html)

Part 3: Import numpy (www.numpy.org/)

**A link to download the datasets used in your assignment, one for the 500 posts and**

**the other for the 100 annotated posts.**

100 annotated post: (<https://docs.google.com/spreadsheets/d/1ggjnI1y3otsuQVQL03Lzly2cYeBU5Udj2pmcUTi_Z6A/edit?usp=sharing>)

500 posts:

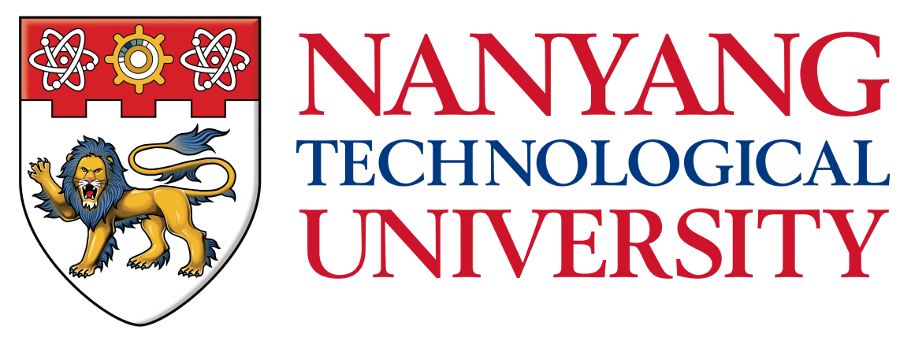
(<https://drive.google.com/file/d/0B-EdIqBldhXRdHAxTWhtQ3M3czg/view?usp=sharing>)

**An installation guide on how to setup your system, and how to use your system**

**(e.g., command lines, input format, parameters).**

**Explanations of sample output obtained from your system.**

（Similar to Report.PDF)



# CZ 4045 Natural Language Processing

# Assignment

# Online Forum Data Processing

# Supervisor: Dr Sun Ai Xin

# Done by: Yap Jian Hui Daniel

# Chua Hui Min

# Andre Culham

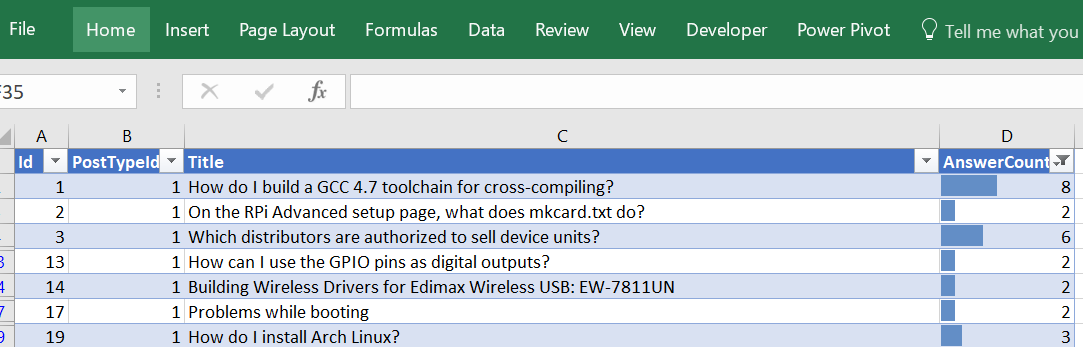
|  |  |
| --- | --- |
| 1. Dataset Collection (10 marks) | Yap Jian Hui Daniel |
| 2. Dataset Analysis and Annotation (30 marks) | Yap Jian Hui Daniel |
| 3. Tokenizer (30 marks) | Chua Hui Min |
| 4. Further Analysis (20 marks) | Chua Hui Min |
| 5. Application (10 marks) | Andre Culham |

**1. Dataset Collection (10 marks)**

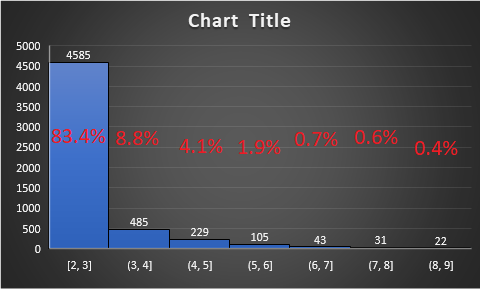
The data were collected from <https://archive.org/details/stackexchange>, as an xml file. Subsequently, post.xml was transferred to an xlsx file using Microsoft Excel 2016 **Excel Mapping Schema**.

The 3 conditions were fulfilled:

* **Minimum 5000** threads (We filtered 5500 threads by using Excel’s countA function)
* All selected threads on **one language** (We focused on RPI, thus narrowing to Python)
* At least **2 posts per thread** (Sorting where ‘0’ & ‘1’ are rejected. Clearly, ‘2’ & ‘3’ are the most common range of post per thread)

****

The above is a snip of our Microsoft excel file, in source code, displaying only the necessary columns (**Id, PostId, Title, AnsweredCount**).

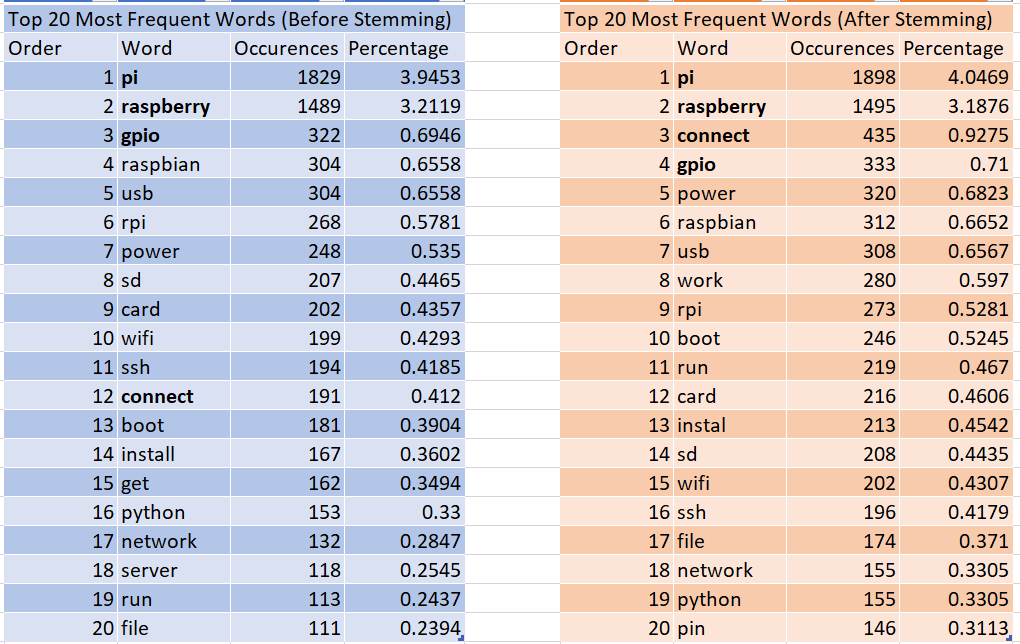


The above chart outlines the **percentage composition** of the dataset post comments, restricting range from 2 to 9, for ease in comparison. Any post number above 9 was ignored, due to exponentially decreasing percentage composition. As mentioned, there are 5500 questions, varying answers shown above per thread.

**2. Dataset Analysis and Annotation (30 marks)**

**2.1 Stemming**

Online text analyser (<https://www.online-utility.org/text/analyzer.jsp>) & Porter Stemmer Online was used (<http://9ol.es/porter_js_demo.html>). Due to the convenience of the online Porter Stemmer, stop words with no semantic meaning (like a, the, by) were easily removed with ease.

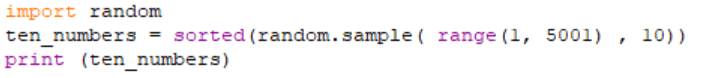
****

The above results were initially unexpected, besides the fact that raspberry and pi consistently came out top (due to the way we collected our dataset). Naturally, technical terms like **‘gpio’, ‘raspberry’ and ‘pi’** came out top.

It turns out, that **after stemming**, generally **occurrences of words increased** instead. This is likely due to occurrences of most frequent words as root words.

**2.2 POS Tagging**

The 10 sentences that are randomly chosen are displayed below, using random.sample(population, k) i.e. **random sample(5000, 10)**



The 10 index are shown below:

https://lh4.googleusercontent.com/XVnXmpqqlxLBNtAB8oJtDQw5f4tWR0FEYQYwb_qXqPJy6LP-eD-m98EP1prwnB5371UUhcq1JmvtxVHzGYVw4QWcusNurbAGx5rjbXQa8VCYuAV44SIhVYAcvReWH22B3FqynHkZ

One of the 10 sentences **before POS tagging**, is shown below:

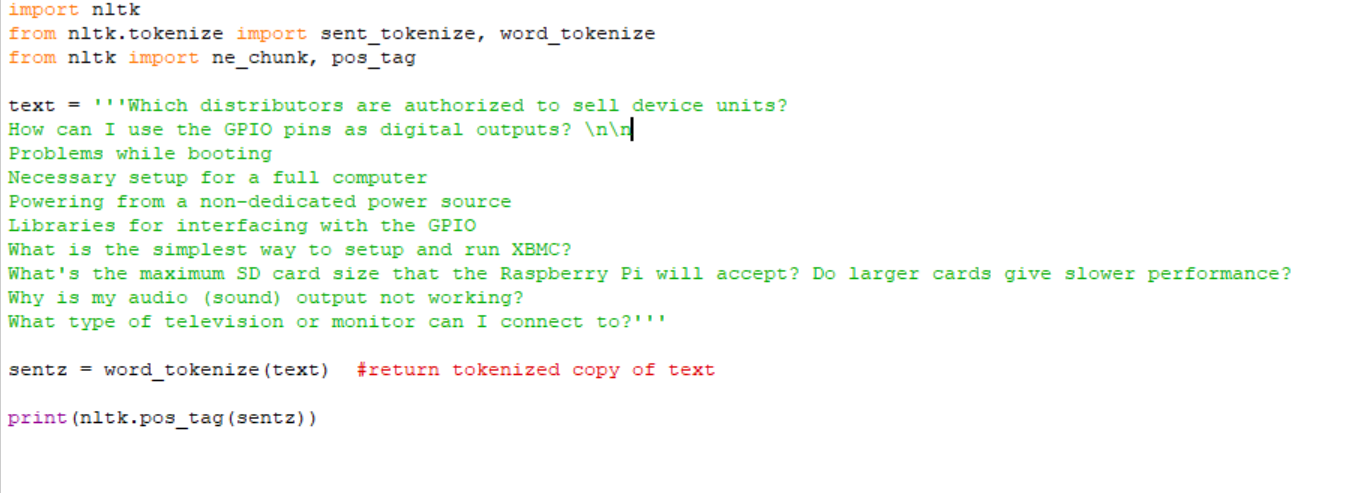
https://lh5.googleusercontent.com/FworatZZn3auAtV9AzRFf3c_4CbSFVcT1UQrSC-vsk-xjSNayC81Bzj2oPcPu2-ipxQkvaXa5-TlihWdyi01oQKc7l66aY1q-gkX0l5q7kNb4jpMfO4sc5fRyUsWFepXyFSkAeCh

POS Tagging is done by applying **word\_tokenize()** to the list of words, to return the tokenize copy of text

https://lh5.googleusercontent.com/HIOW9ElqLxYRbniiVKyn6OZaMaVMfXHAI1hvsOw7ypZNVja6trt3HK_SWjkwaqcZM8831NdKNO0GtMot-g8JMjieUa8FrMChRth5zk-zISoM7Krs3Zkc6I5oio9CRcIqnJm-QbGr

**For POS Tagging for only 1 sentence**, the results are very accurate:

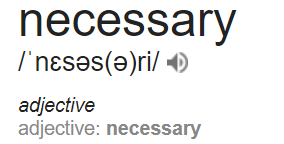
https://lh4.googleusercontent.com/4C7bIUz4jX7C8bwTUOhO1LCNmIr9SlSbnIhP0kd00e5ko7gKuteJ59Q_GMERuPfL50Zn9zaHeptCSlor-OMOekR2_FQQqGTGNff5stPQD3dCiG95L_OJtc06N2WaRxvpbik2MJh_

****

However with more testing, errors starts to surface (Source Code **‘Section 3.2 POS TAGGING Text 1’**). With the 10 sentences, 2 errors surfaced. For one, ‘Necessary’ was supposed to be an **adjective**. As shown on the left in POS Penn tag.

However, it is displayed as NNP on the python NLTK, referring to **‘Proper noun, singular**’, supposedly due to its pairing with ‘setup’.

In addition, abbreviations like ‘Pi” in Raspberry Pi, “GPIO” and “XBNC” (a programming software), had varying accuracy. Although only **XBNC** was wrongly, interpreted as “**RB**” meaning “**Adverb**”. Whereas “Pi” and “GPIO” was accurately interpreted as Noun Phrases.



**2.3 Token Definition and Annotation**

We decided to choose split string of words as a different token, due to the varying usage of left parenthesis as start of function (e.g. String( ) ), and declaration (e.g int indexOf(int ch) ). As such, to minimize such confusion we used **white space as demarcation** of different tokens.

I.e String                 -> 1 Token

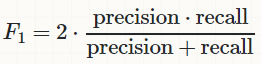
java.lang.String  -> 1 Token

String ( )             -> 3 Tokens

String()               -> 1 Token

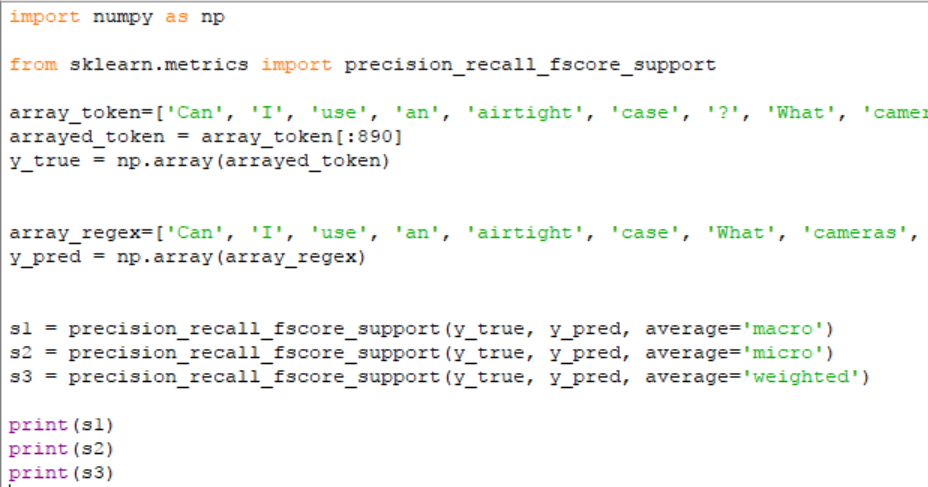
**3. Tokenizer (30 marks)**

Whitespace tokenizer is chosen to tokenize the dataset with regular expression. Scikit learn module is being used to train this tokenizer. To evaluate the effectiveness of tokenizer, a 5-fold cross-validation is applied.



Then the performance of the model can be evaluated by using Precision, Recall, and F1. Report the Precision, Recall, and F1 of your model, and analysis the errors (e.g., case studies on false positives and false negatives).

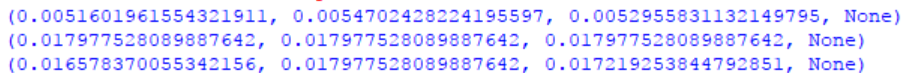
The full tokenized dataset is shown in source code (*nltk section 3.4 5500 tokens.py*). We carry out our tokenization **using regex from NTLK (numpy module).** Sklearn.metrics follows a strict matching form, with **y\_true** refering to our own **annotated tokenization** as mentioned in Section 3.2.3. Whereas, y\_predict is the tested set of tokens from **Regular Expression.**

****

The raw results for precision, recall, and F-score are displayed respectively below. The 1st row (**macro**) calculates metrics for each label, and find their **unweighted mean**. This does not take label imbalance into account.

The 2nd row (**micro**) calculate metrics globally by counting the **total true positives, false negatives and false positives**.

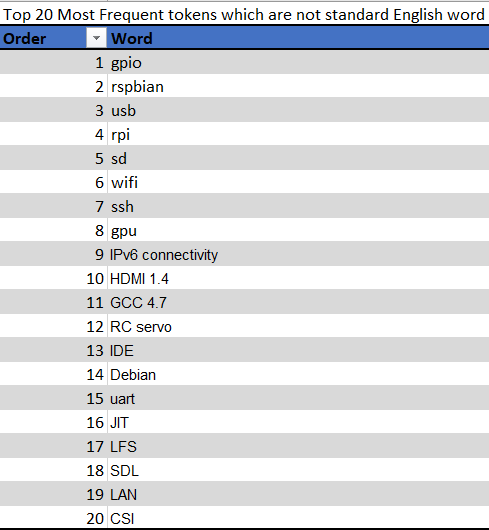
Lastly, the 3rd row (**weighted**) calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label). This alters ‘macro’ to account for **label imbalance**; it can result in an F-score that is not between precision and recall. The excel file and code can be found in Section 3.3 folder.

****

**4 Further Analysis (20 marks)**

**4.1 Irregular Tokens**

Top-20 most frequent tokens which are not standard English words (including their morphological forms)



**4.2 POS Tagging**

|  |  |  |
| --- | --- | --- |
| Original Sentence | POS Tagging results | Remark |
| How do I build a GCC 4.7 toolchain for cross-compiling? | ('How', 'WRB'), ('do', 'VBP'), ('I', 'PRP'), ('build', 'VB'), ('a', 'DT'), ('GCC', 'NNP'), ('4.7', 'CD'), ('toolchain', 'NN'), ('for', 'IN'), ('cross-compiling', 'NN'), ('?', '.') | GCC has been correctly identified Proper noun |
| Building Wireless Drivers for Edimax Wireless USB: EW-7811UN | ('Building', 'NNP'), ('Wireless', 'NNP'), ('Drivers', 'NNP'), ('for', 'IN'), ('Edimax', 'NNP'), ('Wireless', 'NNP'), ('USB', 'NNP'), (':', ':'), ('EW-7811UN', 'NN') | EW-7811UN is identified as a noun instead Proper noun |
| Is there a Linux From Scratch (LFS) ARM equivalent | ('Is', 'VBZ'), ('there', 'EX'), ('a', 'DT'), ('Linux', 'NN'), ('From', 'IN'), ('Scratch', 'NNP'), ('(', '('), ('LFS', 'NNP'), (')', ')'), ('ARM', 'NNP'), ('equivalent', 'NN') | LFS and ARM are correct identified as Proper noun |
| Is it possible to use the GPIO to program a PIC? | ('Is', 'VBZ'), ('it', 'PRP'), ('possible', 'JJ'), ('to', 'TO'), ('use', 'VB'), ('the', 'DT'), ('GPIO', 'NNP'), ('to', 'TO'), ('program', 'NN'), ('a', 'DT'), ('PIC', 'NNP'), ('?', '.') | GPIO and PIC are correctly identified as Proper noun |
| Can a simple cable convert HDMI output to VGA? | ('Can', 'MD'), ('a', 'DT'), ('simple', 'JJ'), ('cable', 'NN'), ('convert', 'NN'), ('HDMI', 'NNP'), ('output', 'NN'), ('to', 'TO'), ('VGA', 'VB'), ('?', '.') | VGA has been misidentified as a Verb instead of noun |
| Can I attach a SATA controller? | ('Can', 'MD'), ('I', 'PRP'), ('attach', 'VB'), ('a', 'DT'), ('SATA', 'NNP'), ('controller', 'NN'), ('?', '.') | SATA is correctly identified as a Proper noun |
| Can I use PATA cables for GPIO? | ('Can', 'MD'), ('I', 'PRP'), ('use', 'VB'), ('PATA', 'NNP'), ('cables', 'NNS'), ('for', 'IN'), ('GPIO', 'NNP'), ('?', '.') | PATA and GPIO have been correctly identified as Proper noun |
| Can I use OpenCV? | ('Can', 'MD'), ('I', 'PRP'), ('use', 'VB'), ('OpenCV', 'NNP'), ('?', '.') | OpenCV has been correctly identified as Proper noun |
| ArchLinux-SSH-First time boot | ('ArchLinux-SSH-First', 'JJ'), ('time', 'NN'), ('boot', 'NN') | ArchLinux-SSH-First is also misidentified as there is no whitespace in between as delimiter |
| I2C library for Mono/C# | ('I2C', 'NNP'), ('library', 'NN'), ('for', 'IN'), ('Mono/C', 'NNP'), ('#', '#') | C and # is misidentified as two words as # can be identified with a tag itself |

**5. Application (10 marks)**

The python application was set up to read dataset in csv form to use the python re module to compile a list of negation expression.  A code snippet is as follows:

negexp.png

Then the application will match the compiled expression for each row (each row contains 1 sentence), incrementing the sentence count and printing the sentence on success. A code snippet is as follows:

