CS4248 Assignment 1: Regexs and Language Models

Distributed on 23 Jan 2021 (*kmy*[v210127])
Due in LumiNUS `Files` by 4 Feb 2021 11:59 PM SGT

kmy[New edits to this specification are annotated in this style.]
This assignment contributes 10 marks towards your final mark for the class, and is graded out of a rubric of 100 points.

Integrity Note. Since this assignment is similar to other assignments for Natural Language Processing courses at other institutions, there are (undoubtedly) solutions posted somewhere. Under the NUS Code of Conduct, you must follow class policy in working on this individual assignment. When in doubt of whether an action would constitute a violation of policy, please ask us on Slack on the `#general` channel or by private Direct Message to Min, **before** attempting the action. To discuss (mostly with your peers in class) the assignment, or to ask topical questions about the assignment itself, pleas use the `assignment-1` channel in Slack. Please familiarize yourself with how conversations work in Slack, so that the topical messages are organised as replies to the starter thread.

Submission. *kmy*[Changed filenames to prevent import problem] Submit your independently coded solutions (`slackbot.py`, `obj1_tokenizer.py`, `obj2_weather.py` and `obj3_ngram.py`) and outputs (`conversation.json`) for Part 1. Also include your write-ups for the discussion parts of Part 1, the whole of Part 2 and your independent work statement as a separate PDF file, named `writeup.pdf`. *kmy*[There is no fixed format for `writeup.pdf` but a suggested template Google Doc / MS Word document template is included.] Include these files all in a directory named after your correct Student ID — `A0000000X` (all letters in CAPITALS) — and place all of your submission files in it (you can simply rename the directory name in the downloaded zip). Re-zip the directory and its files, naming it as `A0000000X.zip`. Submit this file by the assigned deadline to LumiNUS files for *Assignment 1*. Do not include your name or personally identifiable information in your submission.

You must include the text of the two statements below in your submitted work and digitally sign your homework using your Student ID number (starting with `A...`; N.B., not your NUSNET email identifier). Make sure you have attached this statement to your submission either in written or typed form. Delete (and where appropriate, fill in) one of the two forms of Statement 1:

1A. Declaration of Original Work. *By entering my Student ID below, I certify that I completed my assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, I am 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 my answers as per the Pokémon Go rule.*

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

Signed, [Enter your A0000000X Student ID here]

2. References. *I give credit where credit is due. I acknowledge that I used the following websites or contacts to complete this assignment (but please note that many uses of Web search and detailed discussion are not allowed:*

- *Sample. Website 1, for following mathematical proofs.*
- *Sample. My friend, A0000001Y, whom helped me figure out the course deadlines, ... [Fill in appropriately]*

Acknowledgements. Parts of this assignment were adapted from Prof. Xanda Schofield of Harvey Mudd College and Prof. Michael Collins of Columbia University, whose permissions were explicitly secured.

1 Programming a Slackbot

A chatbot, shortened from "chat robot", is a program that imitates human conversation, often for accomplishing task-oriented goals or for socializing, chit chat. Chatbots are heavily used by various business sectors; for instance, in e-commerce. Many e-commerce services use chatbots as their front line means of providing 24/7 customer service, where requests can be first triaged, and routine tasks handled solely by the chatbot, and more complex queries can routed to a live human operator.

In this assignment, we'll be making a simple chatbot. There are three objectives of our chatbot. Please make a bot that can respond to the input based on the objectives below.

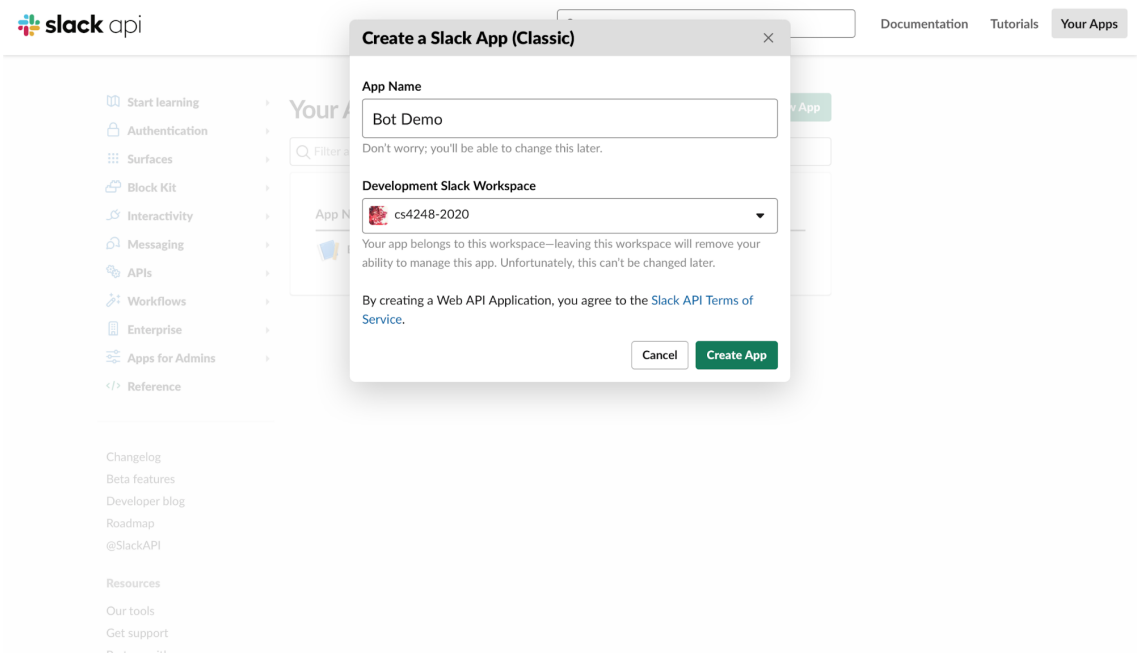
Getting Started. *These steps in this section are prerequisites to doing the assignment. We recommend you spend some time up front to accomplish these tasks first, before attempting the Objectives in Section 1.1.* To create your own chatbot, please follow these steps:

1. **Prerequisites to Getting Started**. Note: For this assignment (and for the future ones), you will need to have an appropriate python environment set up. Beyond a standard environment, this assignment depends on the `nltk` and `matplotlib` libraries that are also useful for you to have access to. We suggest that you spend some time setting up a good `python` (virtual) environment on your work computer. This may take a few hours if you have not done it before. If you have difficulties, please approach our staff in via `general` or `assignments` for help.

2. Download the skeleton code package `A0000000X.zip` from LumiNUS and install the required Python packages using the command:

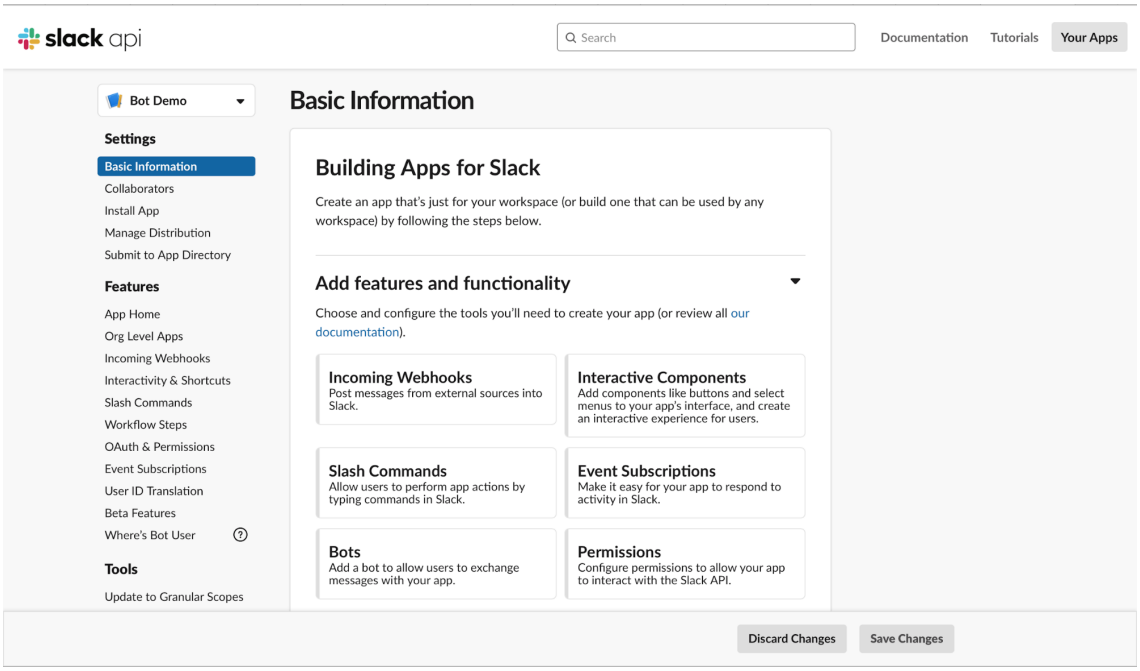
   ```
   pip3 install -r requirements.txt
   ```

3. Create a **classic** Slack app here (https://api.slack.com/apps?new_classic_app=1). Please use this given link. Name the app whatever you want and choose `cs4248-2020` as the destination workspace:

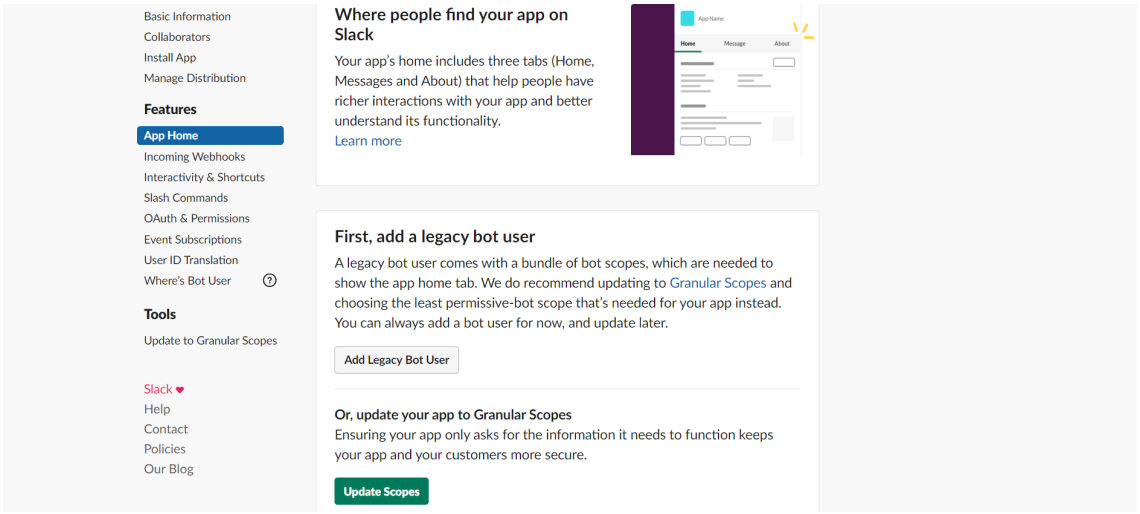


kmy[If you get the error *"Cannot install more apps into this workspace"*, please create a new workspace and select your new workspace as the destination for the Slack app.]

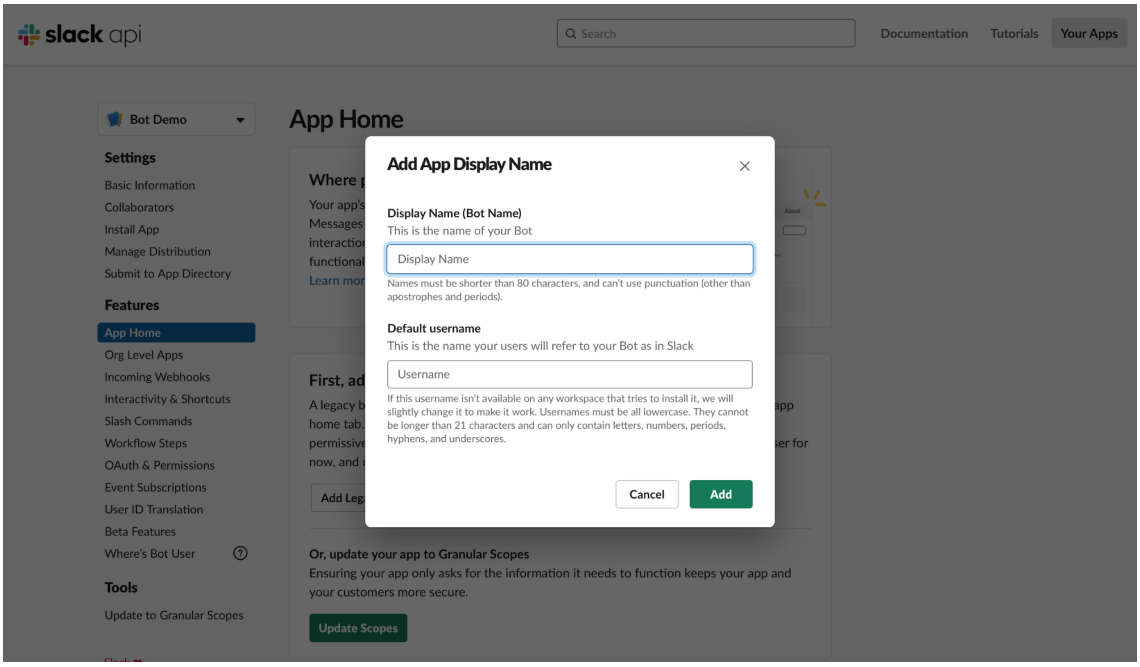
4. Under *Basic Information / Add features and functionality*, click on *Bots*.



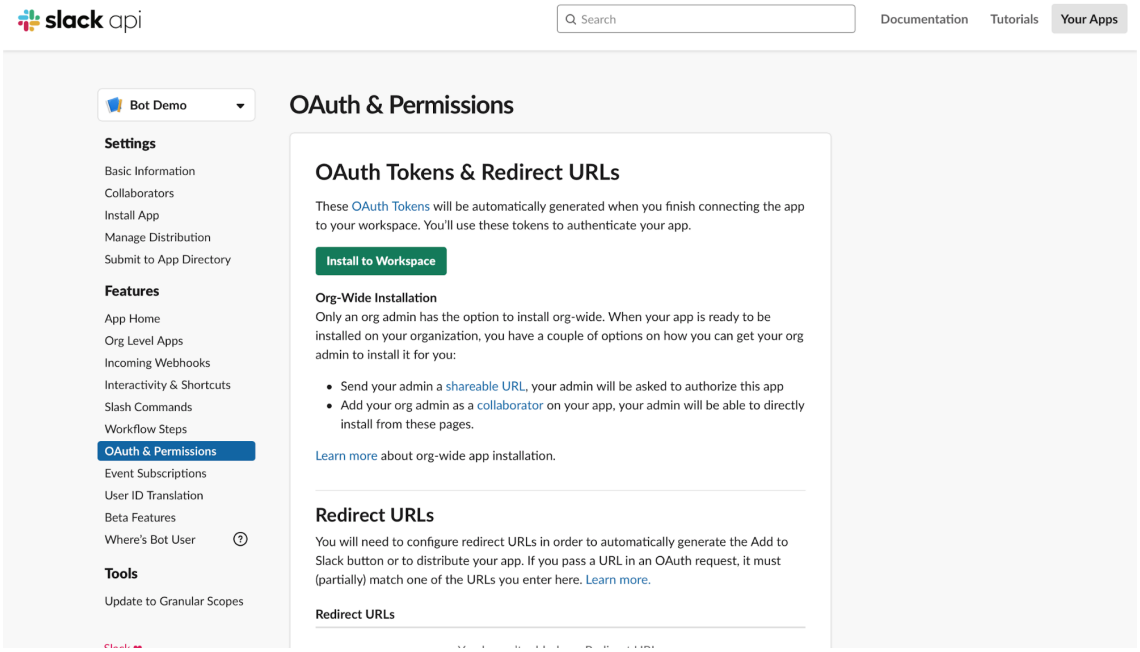
5. Click on *Add Legacy Bot User*.



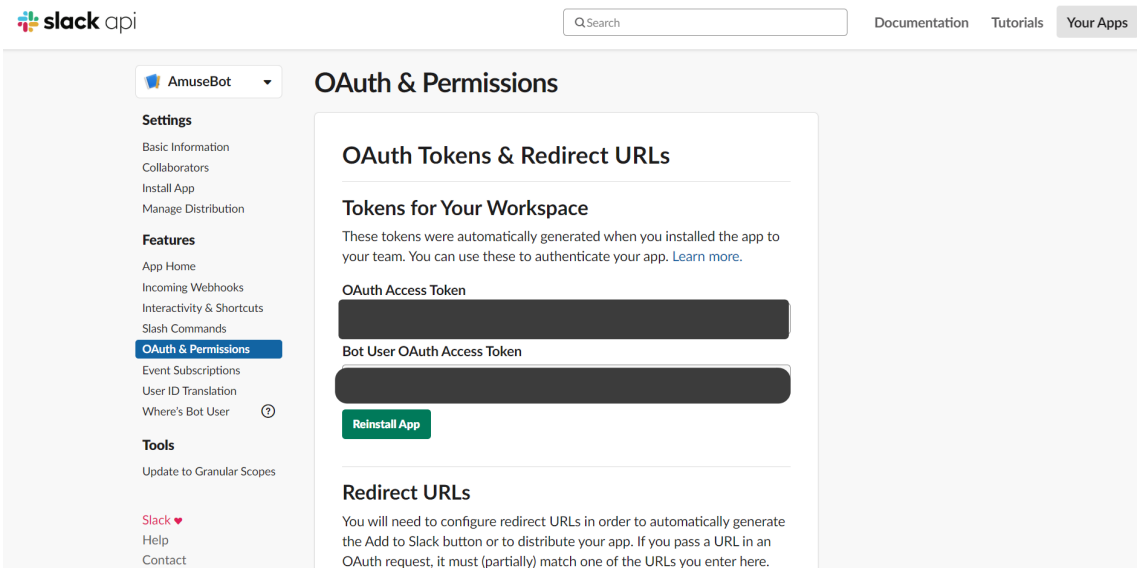
6. Name your bot. Choose a display name and username for your bot (be creative, humorous yet follow good social etiquette; do not choose names that may cause offense to others).



7. Navigate to *OAuth & Permissions* and click on *Install to Workspace*.



8. Simply copy–paste the *Bot User OAuth-Access token* as mentioned in the file. You're done, congratulations – *It'S aAaAlIvE!*.



9. **But wait, there's more...!** *Hmmph. We actually have a body without a soul. Now to the heart of the matter.*

   Complete the set of objectives by making changes to the *make_message* function and other functions from the starter code in `slackbot.py`. See the **Objectives** section below that specify the pedagogical tasks we would like you to do. Note: you will need to create appropriate routing code (using regex perhaps?) to have the user's messages be served by the appropriate respective Objective 1–3 classes.

   Let's continue on ...

10. **Running and logging your conversations with your bot.** To activate your bot on Slack, simply run the `slackbot.py` file. In Slack, search for your bot (with the display name that you had given in Step 5) using the search bar, and have a direct message (DM) conversation with it. If appropriately configured, Slack send the DMs via port communications to your running `slackbot.py` code.

    To exit the bot and save the conversation to a JSON file, simply type the **EXIT** command.

11. **Preparing your Part 1 submission.** For your final submission, re-run the bot again and complete objectives in order. Please do not enter any other commands (apart from $OBJ1, OBJ2, OBJ3$)

12. *kmy*[Updated filenames] Submit `slackbot.py`, `obj1_tokenizer.py`, `obj2_weather.py`, `obj3_ngram_lm.py`, and `conversation.json`, and write the answers to the discussion questions for your Part 1 materials as part of the `writeup.pdf` file that will also contain your answers for Part 2 and your Independent Work Statement.

1.1 Objectives

1. **[10%]** Objective 1 — **Tokenization, Zipf's law**

   Go to Project Gutenberg and download the text of your favourite book. Move the downloaded corpus to your assignment folder. Then complete the skeleton code `tokenizer.py` to accomplish the following tasks below. *kmy*[To be clear, you should not use external libraries to do the tokenization, but instead use your knowledge and wisdom of regular expressions. You may use the stopword list from NLTK if you wish (e.g., for Objective 3 below), and you may want to use the NLTK tokenizer to see how your corpus would have been tokenized if you had access to that (as in an example of good tokenization.]

   The bot should accept the following command from the user (you), in the following format:

   ```
   OBJ1 path=STRING n_top_words=INT lowercase=YES/NO stopwords=YES/NO

   Example: OBJ1 path=./macbeth.txt n_top_words=10 lowercase=NO
                                stopwords=YES
   ```

   kmy[Argument Explanation:

   - `lowercase` (enumerated value: `YES` or `NO`): if set to `YES`, then you need to convert the entire text to lowercase.
   - `stopwords` (enumerated value: `YES` or `NO`): if set to `NO`, then you need to remove all the stopwords from the text. Otherwise, retain stopwords as-is.

   ]

   Your bot should be able to perform the following tasks. Write up any of the requested data or justifications in `writeup.pdf`:

   A. Find the most frequent unigrams (say, top $n = 10$, as set by the variable `n_top_words`) and output them Slack. (Attach a screenshot of this result in the write up.) Was this result as you expected? Reflect on this question in the file `writeup.pdf` by writing a few sentences about your findings.

   B. Plot a graph of relative frequency versus rank (use logs). Is the graph (somewhat) consistent with Zipf's law? Recall Zipf's law states that the rank of the corpus frequency of a word type (cf_i) is inversely proportional to its rank i: $cf_i \propto \frac{1}{i}$. Attach a plot in the writeup and answer the question in the writeup.

   C. Remove stopwords from the text, and repeat the above task. Are the results still consistent with Zipf's law? Again, attach a plot in your writeup along with your answer the question.)

   D. Convert the text to lowercase, and repeat the above task. Are the results still consistent with Zipf's law? Again, attach a plot in your writeup along with your answer the question.)

2. **[20%]** Objective 2 — **How's the Weather?**

Well, it's also good for a bot to have some social skills. Weather is a common subject for passing time.

Let's work on a set of regular expressions that can distinguish genuine requests for weather from non-requests. For the purpose of this assignment, requests should involve one of three cities *Cairo*, *London* or (wait for it) *Singapore* (duh!). If the user's message is a genuine request for weather information, return a the standard constant response that you can find in the class constants in the skeleton file `2_weather.py`. If it isn't, let your bot parrot the default response which is decidedly vacuous (perhaps some of your friends or lecturers also give you this response).

Hint: This exercise is a bit more freeform in nature. Use your creativity to come up with easy and hard expressions that are or aren't asking for the weather. Here we are testing your ability to create regexes that can recognize well, so the bot's responses are decidedly dumb (to help us grade). You can also use the `assignment-1` *channel to help crowdsource (from you and your peers) example natural language strings. If you do, please make sure you follow best practices and credit those who contributed in your* **Reference** *section of your independent work statement.*

kmy[For this objective, you may use a series of regular expressions instead of just a single one. The goal is to have more practice with regular expressions, so your solution should not apply an altering process to the text — e.g., normalization, case transformation — but rather seek to match the tokens as-is.]
Your bot should accept a user's command of the form:

```
OBJ2 <some natural language string>
```

Example Input:

```
OBJ2 What's the weather like in Singapore?
OBJ2 How's today's weather in London?
OBJ2 I am studying at NUS in Singapore.
OBJ2 I hear the weather in Cairo is hot.
```

Example Output:

```
Singapore: hot and humid.
London: rainy and miserable.
Hmm. That's nice.
Hmm. That's nice.
```

3. **[35%]** Objective 3 — **Language Modelling**

Given a corpus (such as what you used in both previous objectives), let's further embue your Slackbot with powers to do language modeling (what we learned in Week 03 — so attempt this in after Week 03's lesson).
Your Slackbot should accept the following command from the user:

```
OBJ3 path=STRING smooth=TYPE n_gram=INT k=FLOAT text=STRING
```

Argument Explanation:

- `path` gives the filesystem path to the input file that the Slackbot should construct the language model from.
- `smooth` (string) selects what type of smoothing to use. You must handle the `add-k` smoothing method, but optionally you can support other forms of smoothing (see below).
- `n_gram` (integer) specifies the number of grams for constructing the language model. E.g., 1 would correspond to a unigram model, 2 to a bigram model.
- `k` (floating point number) specifies the amount of smoothing to apply to the model. Take note to add this float to the counts of individual words.
- `text` (string) specifies the text on which you run your language model and predict the next word(s).

Example Input:

```
OBJ3 path=./macbeth.txt smooth=add-k n_gram=2 k=1.0 text=The quick
                    brown fox jumped over the
```

kmy[After suitable tokenization and normalization – that you accomplished earlier, input your processed corpus into your language model code to have your bot perform the following tasks. Write up the requested data or justifications in `writeup.pdf`]:

- Complete the skeleton code in `ngram_lm.py` which should train a language model on the text specified by the user. Then recognise (using regexs to cater for variations in prompts by the user) input from the user to display appropriate outputs for calls to the functions `generate_word`, `generate_text`, `perplexity`) in Slack. Include sample captured conversations that use the calls in your `writeup.pdf`

- For bonus (capped at 5% for this objective, and not overflowing the 100% marks for the assignment), optionally implement any other smoothing methods (linear interpolation, back-off, discounting, etc.) and see if it improves the perplexity. Give other valid smoothing values to the `smooth` argument to do this.

2 Theory Questions

Write the answers to these questions in your `writeup.pdf` file. You can use both typewritten or embedded photo captures of handwritten work. The former is preferred for convenience of the CS4248 TA staff.

4. **[10%]** — Subtraction Regular Expressions

   Given a string in the form of `A - B = C` where A, B, and C contains arbitrary *kmy*[non-zero] number of the character `a`. Write a regular expression that accepts all valid subtractions, and reject all invalid subtractions. Hint: you may want to learn how regular expressions can capture groups for back reference.

   For instance, the regex should accept:

   ```
   aaaa - aaa = a
   aaaaaa - aa = aaaa
   ```

   and should reject:

   ```
   aaaa - aaa = aa
   aa - aaa = a
   ```

5. **[25%]** Regular Expression (**Language Modelling**)

A language model consists of a vocabulary V, and a function $p(x_1 \dots x_n)$ such that for all sentences $x_1 \dots x_n \in V^+$, $p(x_1 \dots x_n) > 0$, and in addition $\sum_{x_1 \dots x_n \in V^+} p(x_1 \dots x_n) = 1$. Here V^+ is the set of all sequences $x_1 \dots x_n$ such that $n \geq 1$, $x_i \in V$ for $i = 1 \dots (n-1)$, and $x_n =$ **STOP**.

We assume that we have a bigram language model, with

$$p(x_1, \dots x_n) = \Pi_{i=1}^{n} q(x_i | x_{i-1})$$

The parameters $q(x_i|x_{i-1})$ are estimated from a training corpus using a discounting method, with discounted counts

$$c^*(v, w) = c(v, w) - \beta$$

where $\beta = 0.5$.

We assume in this question that all words seen in any test corpus are in the vocabulary V, and each word in any test corpus is seen at least once in training.
There are 3 subparts to this question:

1. For any test corpus, the perplexity under the language model will be less than ∞. True or False? Justify.
2. For any test corpus, the perplexity under the language model will be at most $N + 1$, where N is the number of words in the vocabulary V. True or False? Justify your response.
3. Now consider a bigram language model where for every bigram (v, w) where $w \in V$ or $w =$ **STOP**,

   $$q(w|v) = \frac{1}{N+1}$$

   where N is the number of words in the vocabulary V.
   For any test corpus, the perplexity under the language model will be equal to $N + 1$. True or False? Justify your response.