README for `AutoLookup`, Computer Programming HW2

B07202020
Hao-Chien Wang

October 25, 2019

The full help message can be shown by using `$ python3 main.py -h`. In case `argparse` doesn't work correctly on all platforms, the help message and the docstring of the functions are also pasted here. However, it is recommended to use the help page.

1 Usage

```
$ python main.py [optional arguments]
```

2 Program Description

Goal: Look up difficult words from a text file in an online dictionary. A local wordlist from http://www.newgeneralservicelist.org/ is used, which contains the top words used in American TV subtitles and other sources, and their frequency of being used. This wordlist is used to define the difficulty of a word. The frequency used in this program is normalized by dividing the real frequency by the max frequency. This program reads a text file containing some English words, determine its difficulty by looking up in the local wordlist. If the (normalized) frequency is below a certain value, which can be specified by users and has a default value of 4e-5, then the program will look it up in the specified online dicitonary. Currently only lexico.com and dictionary.com is provided, but the program can be easily modified to add other sources. The source has better lookup results as its webpage is a lot

more simple for scrapers, so is set to be the default value of the source. The scraper only find the first meaning found in the dictionary, but, again, it can be easily modified to find more. For more details about arguments, see the Arguments section. For details about the functions and how they works, see the docstrings in the code.
The modules used by this program are:

- openpyxl
- regular expression
- requests
- urllib
- BeautifulSoup
- argparse

You may need to `pip3/conda/apt-get/other_package_managers install` something if you are missing any one of these.

Tested platform: Ubuntu 19.04 with Python 3.7.3 on Anaconda.

3 Arguments

- `-h`, `--help`: show the help message and exit.
- `--path PATH`, `-p PATH`: Path to the text file. The default value is `./text.txt`
- `--source SOURCE`, `-s SOURCE`: Indicate the dictionary source. Currently only dictionary.com and lexico.com are provided. The latter has much better results. The default value is lexico.com
- `--threshold THRESHOLD`, `-t THRESHOLD`: A float between 0 and 1. Only more difficult words will be looked up if a lower value is specified. The default value is 4e-5.

4 Functions

- `getWordlist()`:
  Read the excel sheet and convert it into a dictinary containing the words and its $\frac{frequency}{max\_frequency}$.

- `getArticle(path)`:
  Read the text file, cut it into words, then add them into a set.
  Arguments:
  path: A string specifying the path to the text file
  return: a set

- `onlineLookup(word)`:
  Look up a word in the specified dictionary source. The word provided must be found in the local wordlist. If the lookup fails, then return a tuple of `(word, None)`.
  Arguments:
  word: the word to look up
  return: A tuple containing the word and its definition found

- `dictLookup(word, d)`:
  Local wordlist lookup. Check if the word is in the wordlist and get its frequency. Also check the stripped words if the word matches any common suffixes.
  Arguments:
  word: the word to look up
  d: The dictionary to look up
  return: If any result is found, then return a tuple of `(word, frequency)`. If the word (or any of the stripped version) is not in the list, then return None.

- `setLookup(s, dictionary)`:
  Look up each word in the set in the local wordlist. If the word is found and its frequency is below the `THRESHOLD`, then do online lookup. Stopwords are removed.
  Arguments:
  s: a set containing the words to be looked up.
  dictionary: the dictionary with the key be the words and the value be

its frequency.
return: a list containing tuples of `(difficult_words, definition)`

5 Example

Content of the text file:

Trump subsequently deleted the tweet, but – here's a shock – he seems to be loving the attention. At 6.09am EST he wrote "Who can figure out the true meaning of "covfefe" ??? Enjoy!"

Enjoy doesn't seems quite right: it's not exactly reassuring to know that the man with the nuclear codes can't execute a tweet accurately. But we're on this ride so we might as well get what entertainment we can from it.

In linguistic terms, covfefe-gate (I know) raises several interesting questions. It's not yet clear what kind of life awaits the coinage in the wider language: perhaps it will remain tied to this moment, a famous one-off. It might, as McCulloch suggests, give rise to a productive suffix. Or it could become a meaningful word in its own right, with a sense akin to "snafu". What is clear is that it's not going to simply vanish. Trump is just too big(ly) for that.

All of which gets me thinking about the influence of significant (let's not say "great") individuals on language. Iberian Spanish makes use of the sound "th" in many places where Latin American speakers just have "s" – for example, the middle consonant in the word gracias. There's a story that this came about because the King of Spain had a lisp, which all his courtiers had to copy to avoid embarrassing him. Unfortunately, it's a myth: "th" occurs only in specific positions, whereas in others "s" survives – not the situation you'd expect if imitation of a complete inability to pronounce "s" was the source.

There are, however, lots of real examples of invented words that survive and flourish – often to the point where it seems bizarre to think they were ever one person's idea. Paul Dickson collected many in his book Authorisms: they include John Milton's roughly 630 contributions to the English language, such as love-lorn, satanic, didactic, liturgical and pandemonium.

Shakespeare's plays apparently introduced the words bedazzle, fashionable, vulnerable and outbreak. There's an obvious point to be made here, particularly with Shakespeare: plays have to be comprehensible to audiences, so it's possible some of the words were current at the time they were performed, and The Bard merely gave us their first recorded uses. We're on firmer ground with more recent examples: chortle was first used in Lewis Carroll's Through the Looking Glass. Cyberspace didn't exist before William Gibson's 1982 short story Burning Chrome.

In a slightly different vein, mistakes made by so many people that they become the standard form are very numerous. One is "to curry favour" – of which the Oxford Dictionaries has a wonderful explanation involving petting a mythical horse. But I haven't been able to unearth any common words which trace their origins to a mistake by a single person. If you know different, tell me in the thread below.

And what now for covfefe? Like it or not, Donald Trump has created a word. Whether it will spread its wings beyond self-reference is a different question. One way of looking at the development of language is by analogy with natural selection. If a word is "fit" enough it will multiply and colonise all sort of environments. I reckon covfefe, though the spawn of an alpha male, is somewhat hobbled by its narrowness of appeal, and won't amount to much. Then again, a year ago, I would have said the same thing about Trump.

Result:
Run `$ python main.py` or `$ python3 main.py` or `$ python3 main.py -p ./text.txt` (time required depends on the internet speed. Typically takes around 15 seconds when tested in my dorm with the text above.)

```
comprehensible : Able to be understood; intelligible.
liturgical : Relating to liturgy or public worship.
pandemonium : Wild and noisy disorder or confusion; uproar.
satanic : Of or characteristic of Satan.
didactic : Intended to teach, particularly in having moral instruction as an ulterior motive.
lisp : A speech defect in which s is pronounced like th in thick and z is pronounced like th in this.
```

coinage : Coins collectively.
alpha : The first letter of the Greek alphabet (A,), transliterated as ‘a’.
unearth : Find (something) in the ground by digging.
imitation : The action of using someone or something as a model.
cyberspace : The notional environment in which communication over computer networks occurs.
mythical : Occurring in or characteristic of myths or folk tales.
bard : A poet, traditionally one reciting epics and associated with a particular oral tradition.
consonant : A basic speech sound in which the breath is at least partly obstructed and which can be combined with a vowel to form a syllable.
narrowness : Small width in relation to length.
tweet : The chirp of a small or young bird.
chrome : Chromium plate as a decorative or protective finish on motor-vehicle fittings and other objects.
akin : Of similar character.
suffix : A morpheme added at the end of a word to form a derivative (e.g. -ation, -fy, -ing, -itis).

Run `$ python3 main.py -t 5e-6`

narrowness : Small width in relation to length.
liturgical : Relating to liturgy or public worship.
coinage : Coins collectively.
suffix : A morpheme added at the end of a word to form a derivative (e.g. -ation, -fy, -ing, -itis).
lisp : A speech defect in which s is pronounced like th in thick and z is pronounced like th in this.