```
import nltk
nltk.download('all')
```

```
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package tagsets_json to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package tagsets_json is already up-to-date!
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_treebanks_v20 is already up-to-
[nltk_data]    |       date!
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package vader_lexicon is already up-to-date!
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Package verbnet is already up-to-date!
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Package verbnet3 is already up-to-date!
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]    |   Package wmt15_eval is already up-to-date!
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package word2vec_sample is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet2022 to /root/nltk_data...
[nltk_data]    |   Package wordnet2022 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Package ycoe is already up-to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection all
True
```

```
# importing regular expression
import re

# words
w = []

# reading text file
with open('/content/a-z-daily-words-meaning.txt', 'r', encoding="utf8") as f:
    file_name_data = f.read()
    file_name_data = file_name_data.lower()
    w = re.findall('\w+', file_name_data)

# vocabulary
main_set = set(w)


# Functions to count the frequency
# of the words in the whole text file
```

```
def counting_words(words):
    word_count = {}
    for word in words:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1
    return word_count


# Calculating the probability of each word
def prob_cal(word_count_dict):
    probs = {}
    m = sum(word_count_dict.values())
    for key in word_count_dict.keys():
        probs[key] = word_count_dict[key] / m
    return probs


pip install pattern
```

```
Requirement already satisfied: pattern in /usr/local/lib/python3.10/dist-packages (3.6)
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packages (from pattern) (1.0.0)
Requirement already satisfied: backports.csv in /usr/local/lib/python3.10/dist-packages (from pattern) (1.0.7)
Requirement already satisfied: mysqlclient in /usr/local/lib/python3.10/dist-packages (from pattern) (2.2.5)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from pattern) (4.12.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from pattern) (5.3.0)
Requirement already satisfied: feedparser in /usr/local/lib/python3.10/dist-packages (from pattern) (6.0.11)
Requirement already satisfied: pdfminer.six in /usr/local/lib/python3.10/dist-packages (from pattern) (20240706)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pattern) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pattern) (1.13.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from pattern) (3.8.1)
Requirement already satisfied: python-docx in /usr/local/lib/python3.10/dist-packages (from pattern) (1.1.2)
Requirement already satisfied: cherrypy in /usr/local/lib/python3.10/dist-packages (from pattern) (18.10.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from pattern) (2.32.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->pattern) (2.6)
Requirement already satisfied: cheroot>=8.2.1 in /usr/local/lib/python3.10/dist-packages (from cherrypy->pattern) (10.0.1)
Requirement already satisfied: portend>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from cherrypy->pattern) (3.2.0)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.10/dist-packages (from cherrypy->pattern) (10.5.0)
Requirement already satisfied: zc.lockfile in /usr/local/lib/python3.10/dist-packages (from cherrypy->pattern) (3.0.post1)
Requirement already satisfied: jaraco.collections in /usr/local/lib/python3.10/dist-packages (from cherrypy->pattern) (5.1.0)
Requirement already satisfied: sgmllib3k in /usr/local/lib/python3.10/dist-packages (from feedparser->pattern) (1.0.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk->pattern) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk->pattern) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk->pattern) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk->pattern) (4.66.6)
Requirement already satisfied: charset-normalizer>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six->pattern) (3.4.0)
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six->pattern) (43.0.3)
Requirement already satisfied: typing-extensions>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from python-docx->pattern) (4.12.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->pattern) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->pattern) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->pattern) (2024.8.30)
Requirement already satisfied: jaraco.functools in /usr/local/lib/python3.10/dist-packages (from cheroot>=8.2.1->cherrypy->pattern) (4.1
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=36.0.0->pdfminer.six->pattern)
Requirement already satisfied: tempora>=1.8 in /usr/local/lib/python3.10/dist-packages (from portend>=2.1.1->cherrypy->pattern) (5.7.0)
Requirement already satisfied: jaraco.text in /usr/local/lib/python3.10/dist-packages (from jaraco.collections->cherrypy->pattern) (4.0.
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from zc.lockfile->cherrypy->pattern) (75.1.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography>=36.0.0->pdfminer.six
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from tempora>=1.8->portend>=2.1.1->cherrypy->
Requirement already satisfied: jaraco.context>=4.1 in /usr/local/lib/python3.10/dist-packages (from jaraco.text->jaraco.collections->che
Requirement already satisfied: autocommand in /usr/local/lib/python3.10/dist-packages (from jaraco.text->jaraco.collections->cherrypy->p
Requirement already satisfied: backports.tarfile in /usr/local/lib/python3.10/dist-packages (from jaraco.context>=4.1->jaraco.text->jara
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil->tempora>=1.8->portend>=2.1.1->
```

```
# LemmWord: extracting and adding
# root word i.e.Lemma using pattern module
import pattern
from pattern.en import lemma, lexeme
from nltk.stem import WordNetLemmatizer


def LemmWord(word):
    return list(lexeme(wd) for wd in word.split())[0]


# Deleting letters from the words
def DeleteLetter(word):
    delete_list = []
    split_list = []
```

```
        # considering letters 0 to i then i to -1
        # Leaving the ith letter
        for i in range(len(word)):
            split_list.append((word[0:i], word[i:]))

        for a, b in split_list:
            delete_list.append(a + b[1:])
        return delete_list


    # Switching two letters in a word
    def Switch_(word):
        split_list = []
        switch_l = []

        #creating pair of the words(and breaking them)
        for i in range(len(word)):
            split_list.append((word[0:i], word[i:]))

        #Printint the first word (i.e. a)
        #then replacing the first and second character of b
        switch_l = [a + b[1] + b[0] + b[2:] for a, b in split_list if len(b) >= 2]
        return switch_l


    def Replace_(word):
        split_l = []
        replace_list = []

        # Replacing the letter one-by-one from the list of alphs
        for i in range(len(word)):
            split_l.append((word[0:i], word[i:]))
        alphs = 'abcdefghijklmnopqrstuvwxyz'
        replace_list = [a + l + (b[1:] if len(b) > 1 else '')
                        for a, b in split_l if b for l in alphs]
        return replace_list


    def insert_(word):
        split_l = []
        insert_list = []

        # Making pairs of the split words
        for i in range(len(word) + 1):
            split_l.append((word[0:i], word[i:]))

        # Storing new words in a list
        # But one new character at each location
        alphs = 'abcdefghijklmnopqrstuvwxyz'
        insert_list = [a + l + b for a, b in split_l for l in alphs]
        return insert_list


    # Collecting all the words
    # in a set(so that no word will repeat)
    def colab_1(word, allow_switches=True):
        colab_1 = set()
        colab_1.update(DeleteLetter(word))
        if allow_switches:
            colab_1.update(Switch_(word))
        colab_1.update(Replace_(word))
        colab_1.update(insert_(word))
        return colab_1

    # collecting words using by allowing switches
    def colab_2(word, allow_switches=True):
        colab_2 = set()
        edit_one = colab_1(word, allow_switches=allow_switches)
        for w in edit_one:
            if w:
                edit_two = colab_1(w, allow_switches=allow_switches)
                colab_2.update(edit_two)
        return colab_2


    # Only storing those values which are in the vocab
    def get_corrections(word, probs, vocab, n=2):
        suggested_word = []
```

```
        best_suggestion = []
        suggested_word = list(
            (word in vocab and word) or colab_1(word).intersection(vocab)
            or colab_2(word).intersection(
                vocab))

        # finding out the words with high frequencies
        best_suggestion = [[s, probs[s]] for s in list(reversed(suggested_word))]
        return best_suggestion


    # Input
    my_word = input("Enter any word:")
    # Function to calculate accuracy
    def calculate_accuracy(corrections, ground_truth):
        correct_predictions = sum(1 for word, _ in corrections if word in ground_truth)
        return correct_predictions / len(ground_truth)
    # Load ground truth data

    # Counting word function
    word_count = counting_words(main_set)
    # Calculating probability
    probs = prob_cal(word_count)
    # Get suggested corrections
    corrections = get_corrections(my_word,probs, main_set)
    # Calculate accuracy
    accuracy = calculate_accuracy(corrections, w)
    print("Top suggestions:")
    for word, prob in corrections:
        print(f"{word}: {prob}")
    print("Accuracy:", accuracy)

    # only storing correct words
    tmp_corrections = get_corrections(my_word, probs, main_set, 2)
    for i, word_prob in enumerate(tmp_corrections):
        if(i < 10):
            print(word_prob[0])
        else:
            break
```

```
Enter any word:thnn
Top suggestions:
thin: 0.00033478406427854036
than: 0.00033478406427854036
then: 0.00033478406427854036
Accuracy: 0.00012870565017804282
thin
than
then
```