CSCI-8450

2018 Spring

Yuchen Wang

1. exercise 6 b,c,f: When you test your answers use the following text:

*One day, his horse ran away. The neighbors came to express their concern: "Oh, that's too bad. How are you going to work the fields now?" The farmer replied: "Good thing, Bad thing, Who knows?"*

*In a few days, his horse came back and brought another horse with her. Now, the neighbors were glad: "Oh, how lucky! Now you can do twice as much work as before!" The farmer replied: "Good thing, Bad thing, Who knows?"*

b. [A-Z][a-z]\*

c. p[aeiou]{,2}t

f. \w+|[^\w\s]+

Test your answers using nltk.re\_show().

Code:

print(**"part b"**)  
nltk.re\_show(**'[A-Z][a-z]\*'**,SimpleText)  
print(**"part c"**)  
nltk.re\_show(**'p[aeiou]{,2}t'**,SimpleText)  
print(**"part f"**)  
nltk.re\_show(**'\w+|[^\w\s]+'**,SimpleText)

Results:

part b

[A-Z][a-z]\* : means the first letter is the capitalized, and the following letters are not capitalized ones. Capitalization.

{One} day, his horse ran away. {The} neighbors came to express their concern: "{Oh}, that's too bad. {How} are you going to work the fields now?" {The} farmer replied: "{Good} thing, {Bad} thing, {Who} knows?" {In} a few days, his horse came back and brought another horse with her. {Now}, the neighbors were glad: "{Oh}, how lucky! {Now} you can do twice as much work as before!" {The} farmer replied: "{Good} thing, {Bad} thing, {Who} knows?"

part c

p[aeiou]{,2}t : means the word, which p is the first letter, t is the last letter and other letters are vowel but no more than two letters.

One day, his horse ran away. The neighbors came to express their concern: "Oh, that's too bad. How are you going to work the fields now?" The farmer replied: "Good thing, Bad thing, Who knows?" In a few days, his horse came back and brought another horse with her. Now, the neighbors were glad: "Oh, how lucky! Now you can do twice as much work as before!" The farmer replied: "Good thing, Bad thing, Who knows?"

part f

\w+|[^\w\s]+: means they are consist of words or vocabulary.

{One} {day}{,} {his} {horse} {ran} {away}{.} {The} {neighbors} {came} {to} {express} {their} {concern}{:} {"}{Oh}{,} {that}{'}{s} {too} {bad}{.} {How} {are} {you} {going} {to} {work} {the} {fields} {now}{?"} {The} {farmer} {replied}{:} {"}{Good} {thing}{,} {Bad} {thing}{,} {Who} {knows}{?"} {In} {a} {few} {days}{,} {his} {horse} {came} {back} {and} {brought} {another} {horse} {with} {her}{.} {Now}{,} {the} {neighbors} {were} {glad}{:} {"}{Oh}{,} {how} {lucky}{!} {Now} {you} {can} {do} {twice} {as} {much} {work} {as} {before}{!"} {The} {farmer} {replied}{:} {"}{Good} {thing}{,} {Bad} {thing}{,} {Who} {knows}{?"}

1. exercise 7 a: In this question, consider SimpleText for testing your function.

Write regular expressions to match the following classes of strings:

* 1. A single determiner (assume that *a*, *an*, and *the* are the only determiners).

Code:

print(**"part a"**)  
nltk.re\_show(**r'\b(a|an|the)\b'**,SimpleText)

Result:

Exercise 7

part a

One day, his horse ran away. The neighbors came to express their concern: "Oh, that's too bad. How are you going to work {the} fields now?" The farmer replied: "Good thing, Bad thing, Who knows?" In {a} few days, his horse came back and brought another horse with her. Now, {the} neighbors were glad: "Oh, how lucky! Now you can do twice as much work as before!" The farmer replied: "Good thing, Bad thing, Who knows?"

1. exercise 21: In this question, use URL <https://www.cs.utexas.edu/~vl/notes/dijkstra.html>

What kind of differences do you see in the tokens you retrieve if you use nltk.word\_tokenize instead of re.findall(). Use nltk.WordNetLemmatizer() on your tokens before checking them against the Words Corpus. Do you find fewer or more unknown words? Why?

Write a function unknown() that takes a URL as its argument, and returns a list of unknown words that occur on that webpage. In order to do this, extract all substrings consisting of lowercase letters (using re.findall()) and remove any items from this set that occur in the Words Corpus (nltk.corpus.words). Try to categorize these words manually and discuss your findings.

Code:

**def** unknown1(raw, words\_list):  
  
 token\_list1 = re.findall(**r"\w+(?:[-']\w+)\*|'|[-.(]+|\S\w\*"**, raw)  
 words\_list1 = [w **for** w **in** token\_list1 **if** w.islower()]  
 unknown\_w = []  
 **for** w **in** words\_list1:  
 **if** w **not in** words\_list:  
 unknown\_w.append(w)  
 print(unknown\_w)  
  
**def** unknown2(raw, words\_list):  
 tokens = nltk.word\_tokenize(raw)  
 wnl = nltk.WordNetLemmatizer()  
 token\_list2 = set([wnl.lemmatize(t) **for** t **in** tokens] )  
 words\_list2 = [w **for** w **in** token\_list2 **if** w.islower()]  
 unknown\_w = []  
 **for** w **in** words\_list2:  
 **if** w **not in** words\_list:  
 unknown\_w.append(w)  
  
 print(unknown\_w)  
  
**def** exercise21():  
 html = urlopen(url).read().decode(**'utf8'**)  
 raw = BeautifulSoup(html,**"html5lib"**).get\_text()  
 words\_list = [w **for** w **in** nltk.corpus.words.words(**'en'**) **if** w.islower()]  
  
  
 print(**"With re.findall:"**)  
 unknown1(raw, words\_list)  
 print(**"With nltk.word\_tokenize():"**)  
 unknown2(raw, words\_list)

Results:

Exercise 21

With re.findall:

['universities', 'addressed', 'graduates', 'honouring', 'newest', 'lives', 'follows', 'lives', 'pressed', 'dreams', 'persons', 'pasts', 'institutions', 'peoples', 'has', 'liberated', 'centuries', 'dates', 'years', 'liberating', 'succeeded', 'tools', 'cities', 'defended', 'succeeded', '3rd', 'caused', 'dikes', 'soldiers', 'wanted', "city's", 'prizes', 'citizens', 'started', 'symbolizes', 'years', 'vigour', 'dedicated', 'exercising', 'encouraged', 'cherished', 'protected', 'referring', 'techniques', 'transmits', 'insights', 'abilities', 'techniques', 'guilds', 'insights', 'abilities', 'has', 'years', "professor's", 'insights', 'abilities', 'universities', 'started', 'techniques', 'mentioned', 'formulated', 'characteristics', 'hides', 'cheats', 'doors', '/industrial', 'cooperation', 'invents', 'so-called', 'reasons', 'results', "nation's", 'organisation', "nation's", 'laboratories', "nation's", 'things', 'academics', 'hated', 'vide', 'killed', 'barbariously', 'murdered', 'buildings', 'fortified', 'students', 'histories', 'tolerated', 'ideas', 'minds', 'minds', 'protects', 'worlds', 'ensures', '"out', 'years', 'world-at-large', 'seems', 'world-wide', 'strikes', 'ill-directed', 'heard', 'efforts', '"post', 'merits', 'lowered', 'feelings', 'hiring', 'ignoring', 'students', 'graduates', 'sciences', 'heads', 'puts', 'others', 'fashion-dependent', 'legislatures', 'bodies', 'scientists', 'things', 'industries', 'perceived', 'existing', 'universities', 'wanted', 'founded', 'decades', "institute's", 'bestowed', 'scholars', "haven't", 'things', 'frontpage', 'reported', 'strings', 'doubts', 'seems', '70s', '80s', '90s', 'world-wide', 'credo', 'seems', 'high-tech', 'industries', 'seems', 'surrounds', 'scepticism', 'well-kept', 'hype', 'slogans', 'fads', 'fashions', 'seems', 'sadenning', 'promoted', 'deprived', 'heights', 'ills', 'etc', 'operas', 'quizzes', 'welcomed', 'greatest', 'allowed', '"audio', 'aids', 'called', 'advent', 'terminals', 'debugging', 'programming', 'problems', 'advent', 'colour', 'screens', '"algorithm', 'software', 'states', 'multimedia', '/communication', 'hype', 'bits', 'arrived', '"on', '"on', "don't", 'hype', 'favour', '"home', 'remarked', 'knowledge-based', 'information-inundated', 'distractions', 'generated']

With nltk.word\_tokenize():

['knowledge-based', 'ignoring', 'favour', 'fashion-dependent', 'wanted', 'sadenning', 'arrived', 'invents', 'newest', "nation's", 'advent', 'academic/industrial', 'frontpage', 'fortified', 'vigour', 'hated', 'encouraged', 'others', 'started', 'symbolizes', 'tolerated', 'honouring', 'allowed', 'transmits', 'heard', 'deprived', 'world-wide', 'welcomed', 'etc', 'audio-visual', 'dedicated', 'ill-directed', 'protects', 'post-tenure', 'existing', 'well-kept', 'programming', 'murdered', 'cherished', 'referring', 'called', 'promoted', 'lowered', 'addressed', 'reported', 'multimedia/communication', 'hiring', 'pas', 'generated', 'vide', 'liberated', 'pressed', 'mentioned', 'caused', 'cooperation', 'defended', "professor's", 'seems', 'credo', 'scepticism', "'ll", 'organisation', 'so-called', 'colour', 'hype', 'protected', 'founded', 'perceived', 'greatest', "'m", 'formulated', "n't", 'killed', 'succeeded', 'barbariously', 'liberating', 'high-tech', 'software', 'follows', 'bestowed', 'remarked', 'debugging', 'world-at-large', 'information-inundated', 'ensures', '3rd', "'s", 'exercising']

We can see that there are some noun plurals, verbs perfect tense, third person singular form of the verb from the re.findall(). Also, if we use word\_tokenize(), there are fewer words than before, because we do not have the vocabulary like ‘70s’, ‘80s’.

1. exercise 25: In this question, use SimpleText for reporting the outcome.

*Pig Latin* is a simple transformation of English text. Each word of the text is converted as follows: move any consonant (or consonant cluster) that appears at the start of the word to the end, then append *ay*, e.g. *string* → *ingstray*, *idle* → *idleay*. http://en.wikipedia.org/wiki/Pig\_Latin

1. Write a function to convert a word to Pig Latin.
2. Write code that converts text, instead of individual words.
3. Extend it further to preserve capitalization, to keep qu together (i.e. so that quiet becomes ietquay), and to detect when y is used as a consonant (e.g. yellow) vs a vowel (e.g. style).

Code:

**def** word\_to\_pig\_latin(word):  
 *"""takes a word and converts it to pig latin"""  
  
 # matches on a cluster of consonants* pattern = re.compile(**r'^[^aeiouAEIOU]+'**)  
  
 **if** re.findall(**r'^qu'**, word):  
 *# keeps qu together a la quiet* pattern = re.compile(**r'^qu'**)  
 beginning = re.findall(pattern, word)  
 word = pattern.sub(**''**, word)  
 word += str(beginning[0]) + **'ay'  
 return** word  
  
 **elif** re.findall(**r'[^aeiouAEIOU]y[^aeiouAEIOU]'**, word):  
 *# if y has a consonant on either side it treats it like a vowel* pattern = re.compile(**r'^[^aeiouAEIOUy]+'**)  
 beginning = re.findall(pattern, word)  
 word = pattern.sub(**''**, word)  
 word += str(beginning[0]) + **'ay'  
 return** word  
  
 *# stores the beginning match* **elif** re.findall(pattern, word):  
 beginning = re.findall(pattern, word)  
  
 *# pulls out those consonants and gets rid of them* word = pattern.sub(**''**, word)  
  
 *# adds the consonants onto the end of the word* word += str(beginning[0]) + **'ay'  
 return** word  
  
**def** convert\_all(text):  
 *"""converts all words in a given text to pig latin"""* pig\_tokens = **''** *#tokenizes the text* tokens = word\_tokenize(text)  
  
 *#regex for non-alphabetical characters* pattern = re.compile(**r'[^a-zA-Z]'**)  
  
 *#converts the words to pig latin and appends them to the sentence.* **for** token **in** tokens:  
 **if not** re.findall(pattern, token):  
 word = word\_to\_pig\_latin(token)  
  
 **if** re.findall(**r'[A-Z]'**, word):  
 word = word.lower()  
 word = word.capitalize()  
 pig\_tokens += **' '** + word  
 **else**:  
 pig\_tokens += token  
  
 pig\_text = **''**.join(pig\_tokens)  
  
 **return** pig\_text  
  
**def** exercise25():  
 print(convert\_all(SimpleText))  
 print(word\_to\_pig\_latin(**'quiet'**))  
 print(word\_to\_pig\_latin(**'yellow'**))  
 print(word\_to\_pig\_latin(**'style'**))

Results:

Exercise 25

One ayday, ishay orsehay anray away. Ethay eighborsnay amecay otay express eirthay oncerncay:`` Oh, atthay's ootay adbay. Owhay are ouyay oinggay otay orkway ethay ieldsfay ownay?'' Ethay armerfay epliedray:`` Oodgay ingthay, Adbay ingthay, Owhay owsknay?'' In a ewfay aysday, ishay orsehay amecay ackbay and oughtbray another orsehay ithway erhay. Ownay, ethay eighborsnay ereway adglay:`` Oh, owhay uckylay! Ownay ouyay ancay oday icetway as uchmay orkway as eforebay!'' Ethay armerfay epliedray:`` Oodgay ingthay, Adbay ingthay, Owhay owsknay?''

ietquay

ellowyay

ylestay