Inngangur að máltækni - Verkefni 2

Guðmundur Óli Norland

1 Kóði

verkefni2.py

```python
import matplotlib.pyplot as plt
import numpy as np
import random as rd

from collections import Counter
from math import log
from operator import itemgetter
from scipy.interpolate import BSpline

from nltk import (
  bigrams,
  ConditionalFreqDist,
  FreqDist,
)
from nltk.corpus import (
  brown,
  cmudict,
  genesis,
  gutenberg,
  names,
  stopwords,
  wordnet,
)
from nltk.tokenize import word_tokenize

# 8. Define a conditional frequency distribution over the Names corpus that
# allows you to see which initial letters are more frequent for males vs. females
def ver8():
  tvenndir = [
    # tvennd á forminu (kyn, fyrsti stafur)
    (kyn.split('.')[0], stafur[:1])
    for kyn in names.fileids()
    # öll nöfn undir viðkomandi kyni
    for stafur in names.words(kyn)
  ]
  # fáum tíðnidreifingu
  cfd = ConditionalFreqDist(tvenndir)
  # prentum töflu með tíðnidreifingunni
  print(cfd.tabulate())

# 12. The CMU Pronouncing Dictionary contains multiple pronunciations for certain words.
# How many distinct words does it contain?
# What fraction of words in this dictionary have more than one possible pronunciation?
def ver12():
  # öll orð
  words = cmudict.words()
  # einstök orð
  einstok = set(words)
  # tíðnidreifing
  fd = FreqDist(words)
  # til að telja fjölda orða með fleiri en einn framburð þurfum við að sía út orðin sem koma fyrir oftar en einu
  fjoldi = len([w for w in einstok if fd[w] > 1])
  # hlutfall orða með fleiri en einn framburð
```

```
  hlutfall = fjoldi / len(einstok)
  # prentum niðurstöðurnar
  print("FJöldi einstakra orða:", len(einstok))
  print("Hlutfall orða með fleiri en einn framburð:", str(round(hlutfall, 3) * 100) + "%")

# 13. What percentage of noun synsets have no hyponyms? You can get all noun synsets using wn.all_synsets('n').
def ver13():
  noun_synsets = list(wordnet.all_synsets("n"))
  noun_synsets_len = len(noun_synsets)
  # sækjum öll synset með engin hyponyms
  no_hypos = [n for n in noun_synsets if len(n.hyponyms()) == 0]
  no_hypos_len = len(list(no_hypos))
  # hlutfall með engum hypos
  hlutfall = no_hypos_len / noun_synsets_len
  print("Hlutfall samheitamengja með engum undirheitum:", str(round((hlutfall * 100), 2)) + "%")

# 15. Write a program to find all words that occur at least three times in the Brown Corpus.
def ver15():
  words = list([w.lower() for w in brown.words()])
  fd = FreqDist(words)
  thrisvar_eda_oftar = list([w for w in words if fd[w] >= 3])
  # prentum lengdina og skilum listanum
  print(len(thrisvar_eda_oftar))
  return thrisvar_eda_oftar

# 16. Write a program to generate a table of lexical diversity scores (i.e. token/type ratios),
# as we saw in 1.1. Include the full set of Brown Corpus genres (nltk.corpus.brown.categories()).
# Which genre has the lowest diversity (greatest number of tokens per type)?
# Is this what you would have expected?
def ver16():
  cats = brown.categories()
  laegst = 1000000
  laegst_nafn = ""
  # ítrum yfir alla flokkana og finnum breytileikastuðulinn fyrir þá alla
  for cat in cats:
    words = brown.words(categories=cat)
    words_uniq = set(words)
    studull = len(words) / len(words_uniq)
    # uppfærum ef núverandi lægst
    if studull < laegst:
      laegst = studull
      laegst_nafn = cat
  # húmor er með minnsta fjölbreytileikann sem kemur svo sem ekki á óvart, mikið um endurtekningar
  print("Minnsti fjölbreytileikinn:", laegst_nafn)

# 18. Write a program to print the 50 most frequent bigrams (pairs of adjacent words) of a text,
# omitting bigrams that contain stopwords.
def ver18():
  def freq_bia(texti):
    # gerum ráð fyrir að textinn sé á ensku
    sw = stopwords.words("english")
    # tilreiðum textann
    texti = word_tokenize(texti)
    # búum til bigrams
    bigrams = list(zip(texti, texti[1:]))
    # tökum út öll sem innihalda óorðin
```

```
    bigrams = list([(b[0].lower(), b[1].lower()) for b in bigrams if b[0].lower() in sw or b[1].lower() in sw])
    # finnum 50 algengustu pörin
    algengustu = Counter(bigrams).most_common(50)
    return algengustu

  print(freq_bia(gutenberg.raw('austen-emma.txt')))

# 20. Write a function word_freq() that takes a word and the name of a section of the Brown Corpus as arguments,
# and computes the frequency of the word in that section of the corpus.
def ver20():
  def word_freq(section, word):
    if section not in brown.fileids():
      raise LookupError("Þessi hluti er ekki til.")
    words = list([w.lower() for w in brown.words(section)])
    # tíðnidrefing
    fd = FreqDist(words)
    # tíðni orðsins
    tidni = fd[word]
    return tidni

  print("Fjöldi orðsins 'the' í hluta ca01:", word_freq("ca01", "the"))

# 21. Write a program to guess the number of syllables contained in a text,
# making use of the CMU Pronouncing Dictionary.
def ver21():
  d = cmudict.dict()
  # cmudict inniheldur "stress"-tölu í endan fyrir þau fónem sem innihalda sérhljóða,
  # getum nýtt okkur það til að telja þá (atkvæðin)
  def nsyl_word(word):
    n = [0]
    try:
      n = [len(list(y for y in x if y[-1].isdigit())) for x in d[word.lower()]]
    except:
      pass
    return list(n)[0]

  def nsyl_text(text):
    # tilreiðum
    words = word_tokenize(text)
    count = 0
    for word in words:
      count += nsyl_word(word)
    return count

  print("Atkvæði í setningunni 'I am a weirdo':", nsyl_text("I am a weirdo"))
  print("Atkvæði í Emmu eftir Jane Austen:", nsyl_text(gutenberg.raw('austen-emma.txt')))

#  Zipf's Law: Let f(w) be the frequency of a word w in free text.
# Suppose that all the words of a text are ranked according to their frequency,
# with the most frequent word first. Zipf's law states that the frequency of a
# word type is inversely proportional to its rank (i.e. f × r = k, for some constant k).
# For example, the 50th most common word type should occur three times as frequently as
# the 150th most common word type.
#
# Write a function to process a large text and plot word frequency against word rank using pylab.plot.
# Do you confirm Zipf's law?
```

```
# (Hint: it helps to use a logarithmic scale).What is going on at the extreme ends of the plotted line?
#
# Generate random text, e.g., using random.choice("abcdefg "), taking care to include the space character.
# You will need to import random first.
# Use the string concatenation operator to accumulate characters into a (very) long string.
# Then tokenize this string, and generate the Zipf plot as before,
# and compare the two plots. What do you make of Zipf's Law in the light of this?
def ver23():
  words = gutenberg.words('austen-emma.txt')
  # fall til að finna tíðni orðsins úr textanum
  def f(w, fd):
    return (w, fd[w])

  def tvenndir_fun(words):
    # tíðnidrefing
    fd = dict(Counter(words))

    tvenndir = []
    for w in words:
      tvenndir.append(f(w, fd))
    # tökum út endurtekningar
    tvenndir = list(set(tvenndir))
    # röðum
    tvenndir = sorted(tvenndir, key=itemgetter(1), reverse=True)
    return tvenndir

  tvenndir = tvenndir_fun(words)

  # þurfum bara indexin og tíðnigildin
  x = np.array([tvenndir.index(t) for t in tvenndir])

  y = np.array([t[1] for t in tvenndir])

  #plt.plot(x, y)
  #plt.yscale("log")
  #plt.show()

  # Við sjáum á grafinu að lögmál Zipf's stenst ansi vel. T.d. ætti fyrsta orðið að vera 1000 sinnum
  # algengara en það þúsundasta. Það stenst þar sem fyrsta orðið kemur
  # 10000 sinnum fram en það þúsundasta 10 sinnum. Annað dæmi er að tvöþúsandasta orðið kemur sirka
  # 8 sinnum fram og það fjögurþúsundasta 4 sinnum fram sem stenst einnig
  # þar sem 4000 / 2000 = 2. Zipf veit hvað hann syngur.

  # búum til mjög langan random texta (milljón stafir)
  random_text = ""
  for i in range(1000000):
    random_text += rd.choice("abcdefg ")
  # tilreiðum textann
  random_text_tokens = word_tokenize(random_text)
  tvenndir = tvenndir_fun(random_text_tokens)
  # þurfum bara indexin og tíðnigildin
  x = []
  for i in range(len(tvenndir)):
    x.append(i)
  x = np.array(x)
  y = np.array([t[1] for t in tvenndir])
```

```
  plt.plot(x, y)
  plt.xscale("log")
  plt.yscale("log")
  plt.show()

# Modify the text generation program in 2.2 further, to do the following tasks:
# Store the n most likely words in a list words then randomly choose a word from the
# list using random.choice(). (You will need to import random first.)
# Select a particular genre, such as a section of the Brown Corpus, or a genesis translation,
# one of the Gutenberg texts, or one of the Web texts.
# Train the model on this corpus and get it to generate random text.
# You may have to experiment with different start words. How intelligible is the text?
# Discuss the strengths and weaknesses of this method of generating random text.
# Now train your system using two distinct genres and experiment with generating text in the hybrid genre.
# Discuss your observations.
def ver24():
  def generate_model(cfdist, word, num=15):
    for i in range(num):
        print(word, end=' ')
        # í staðinn fyrir að taka vinsælasta orðið (max) eins og í upprunalega veljum við eitt af vinsælustu af
        items = cfdist[word].items()
        topp_nr = int(len(items) / 2)
        topp = sorted(cfdist[word].items(), key=itemgetter(1), reverse=True)[:topp_nr]
        word = rd.choice(topp)[0]
  text = genesis.words('english-kjv.txt')
  b = bigrams(text)
  cfd = ConditionalFreqDist(b)
  generate_model(cfd, 'living')

# What is the branching factor of the noun hypernym hierarchy? I.e. for every noun synset that has hyponyms -
# or children in the hypernym hierarchy - how many do they have on average?
# You can get all noun synsets using wn.all_synsets('n').
def ver26():
  all_synsets = wordnet.all_synsets('n')
  # sækjum öll hypernym fyrir öll synset
  hyper_counts = [len(syn.hypernyms()) for syn in all_synsets]
  # meðaltalið
  average = sum(hyper_counts) / len(hyper_counts)
  print("Tréþáttunin er:", average)

# The polysemy of a word is the number of senses it has. Using WordNet
# we can determine that the noun dog has 7 senses with: len(wn.synsets('dog', 'n')).
# Compute the average polysemy of nouns, verbs, adjectives and adverbs according to WordNet.
def ver27():
  words = list(wordnet.words())
  def f(ordaflokkur, stafur):
    x = [len(wordnet.synsets(w, stafur)) for w in words]
    y = sum(x) / len(x)
    print(f"{ordaflokkur} meðaltal:", y)
  ordaflokkar = [("nafnorð", "n"), ("sagnorð", "v"), ("lýsingarorð", "a"), ("atviksorð", "r")]
  for flokkur in ordaflokkar:
    f(flokkur[0], flokkur[1])

def v(tala):
  print()
```

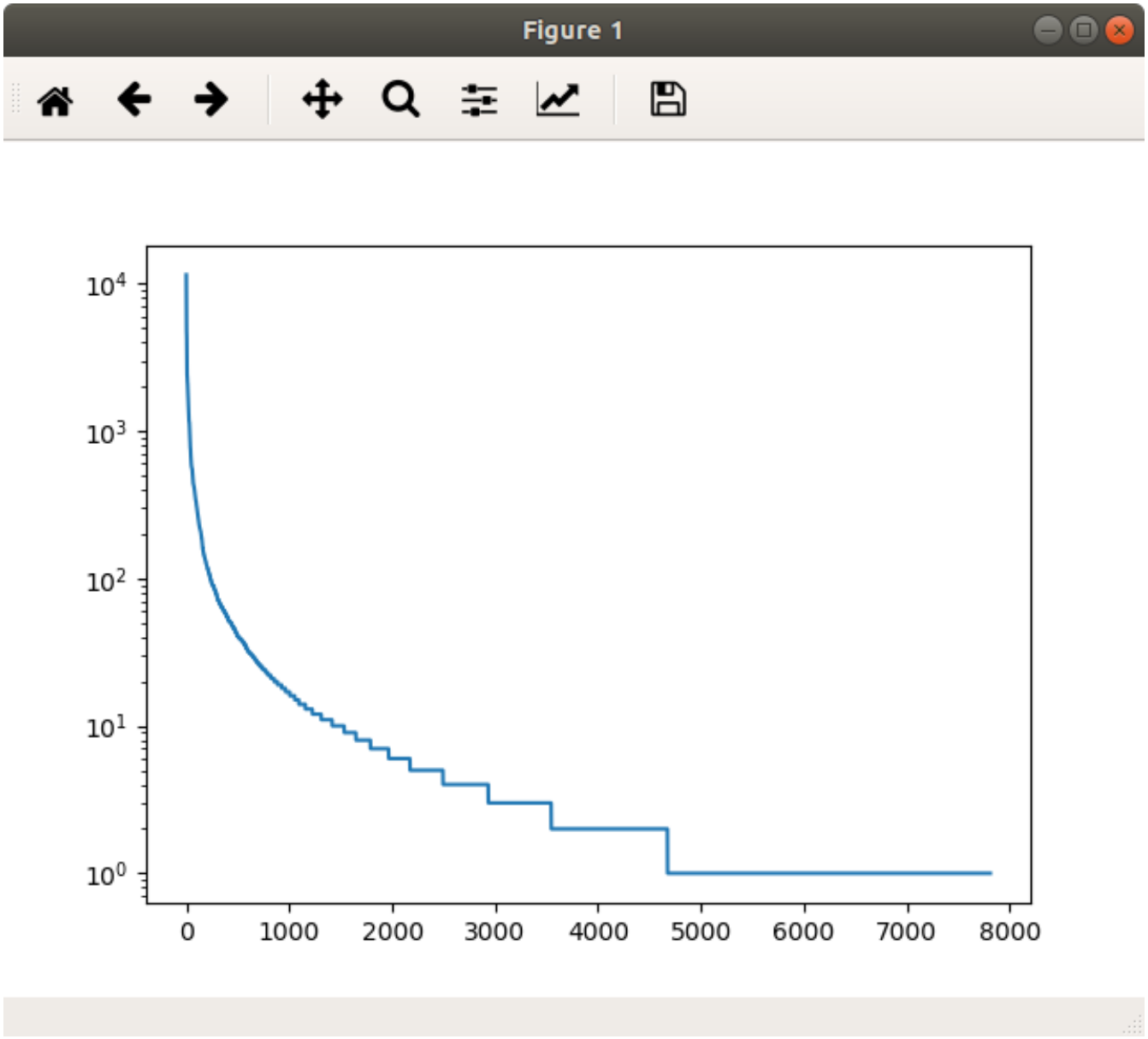
```
  print()
  print(f"Verkefni {tala}")
  print("--------------------------------------------------------------------")

verkefni = [8, 12, 13, 15, 16, 18, 20, 21, 23, 24, 26, 27]
for ver in verkefni:
  v(ver)
  locals()[f"ver{ver}"]()
```

2 Úttak

```
Verkefni 8
------------------------------------------------------------
         A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
female 443 246 469 308 251 144 213 124  83 293 276 332 484 158  66 121   9 247 309 198  14 105  54   5  18  31
  male 213 173 166 146 119  87 156 163  45 144  70 113 200  77  52 101  15 200 238 188  22  50 151   7  16  31
None


Verkefni 12
------------------------------------------------------------
FJöldi einstakra orða: 123455
Hlutfall orða með fleiri en einn framburð: 7.5%


Verkefni 13
------------------------------------------------------------
Hlutfall samheitamengja með engum undirheitum: 79.67%


Verkefni 15
------------------------------------------------------------
1124802


Verkefni 16
------------------------------------------------------------
Minnsti fjölbreytileikinn: humor


Verkefni 18
------------------------------------------------------------
[((',', 'and'), 1882), ((';', 'and'), 867), (('to', 'be'), 605), (('.', 'i'), 570), ((',', 'i'), 569), (('of', '


Verkefni 20
------------------------------------------------------------
Fjöldi orðsins 'the' í hluta ca01: 155


Verkefni 21
------------------------------------------------------------
Atkvæði í setningunni 'I am a weirdo': 5
Atkvæði í Emmu eftir Jane Austen: 218933


Verkefni 23
------------------------------------------------------------


Verkefni 24
------------------------------------------------------------
living creature that came into Egypt unto me at even me not be blessed of

Verkefni 26
```

```
----------------------------------------------------------------
Tréþáttunin er: 0.9237045606770992


Verkefni 27
----------------------------------------------------------------
nafnorð meðaltal: 1.0089677270443838
sagnorð meðaltal: 0.2903819260586806
lýsingarorð meðaltal: 0.20851153381396548
atviksorð meðaltal: 0.03812471997067329
```

Mynd 1: Verkefni 23 - 1



Mynd 2: Verkefni 23 - 2