Inngangur að máltækni - Verkefni 3

Guðmundur Óli Norland

1 Kóði

verkefni3.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
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, DefaultTagger, FreqDist, NgramTagger
from nltk.corpus import brown, cmudict, genesis, gutenberg, names, stopwords, wordnet
from nltk.tokenize import word_tokenize


def prosenta(tala):
    return str(round(tala * 100, 1)) + "%"


# 14. Use sorted() and set() to get a sorted list of tags used in the Brown corpus, removing duplicates.
def ver14():
    tagged_words = brown.tagged_words()
    tags_unique = sorted(set([t for (w, t) in tagged_words]))
    print("Úttaksbrot:", tags_unique[:20])
    return tags_unique


# 18. Generate some statistics for tagged data to answer the following questions:
# a) What proportion of word types are always assigned the same part-of-speech tag?
# b) How many words are ambiguous, in the sense that they appear with at least two tags?
# c) What percentage of word tokens in the Brown Corpus involve these ambiguous words?
def ver18(bin=None, gold=None):
    # byrjun á að sækja allar ólíkar tvenndir
    if bin:
        tagged_words = bin
    elif gold:
        tagged_words = gold
    else:
        tagged_words = list(set(brown.tagged_words()))
    # listi af öllum orðum úr tvenndunum
    words = [w for (w, t) in tagged_words]
    # tíðni
    fd = FreqDist(words)
    # síðum út það sem kemur bara einu sinni
    with_unique_tag = [w for w in words if fd[w] == 1]
    with_unique_tag_len = len(with_unique_tag)
    unique_words_len = len(set(words))
    # hlutfallið er því fjöldi einstakra orða með sama taggið alltaf / fjöldi einstakra orða
    print(
        "a) Hlutfall orða með sama taggið alltaf:",
        prosenta(with_unique_tag_len / unique_words_len),
    )
    # fjöldi tvíræðra orða er þá einstök orð - einstök orð með sama taggið alltaf
```

```
    print("b) Fjöldi tvíræðra orða:", unique_words_len - with_unique_tag_len)
    if bin:
        return
    # words er listi allra orða úr töggunum svo við síum tvíræðu orðin úr þeim lista
    amb_unique = list(set([w for w in words if fd[w] > 1]))
    involve_amb = [w for w in words if w in amb_unique]
    print("c) Hlutfall tvíræðra orða:", prosenta(len(involve_amb) / len(words)))


# Beygingarlýsingu íslensks nútímamáls (BÍN) á http://bin.arnastofnun.is/mimisbrunnur/.
def ver18_bin():
    # dálkur 1 geymir uppflettiorð, 3 orðflokka
    tvenndir = []
    csv = pd.read_csv("./SHsnid.csv/SHsnid.csv", sep=";")
    for row in csv.itertuples():
        tvenndir.append((row[1], row[3]))
    # keyrum a og b lið fyrir ver18
    tvenndir = list(set(tvenndir))
    ver18(tvenndir)


# Sækið markaða íslenska texta úr Gullstaðlinum á http://www.malfong.is/?pg=gull.
# Gerið aftur verkefni 18 (a, b, og c) og notið Gullstaðalinn
def ver18_gold():
    tvenndir = []
    texti = open("./MIM-GOLD-1_0/MIM-GOLD-1_0/books.txt", "r")
    for line in texti:
        split = line.split("\t")
        if len(split) > 1:
            o = split[0]
            flokkur = split[1]
            tvenndir.append((o, flokkur))
    tvenndir = list(set(tvenndir))
    ver18(None, tvenndir)


# 19. The evaluate() method works out how accurately the tagger performs on this text. For example,
# if the supplied tagged text was [('the', 'DT'), ('dog', 'NN')] and the tagger produced the output
# [('the', 'NN'), ('dog', 'NN')], then the score would be 0.5.
# Let's try to figure out how the evaluation method works:
#
# a) A tagger t takes a list of words as input, and produces a list of tagged words as output.
# However, t.evaluate() is given correctly tagged text as its only parameter.
# What must it do with this input before performing the tagging?
#
# b) Once the tagger has created newly tagged text, how might the evaluate() method go about
# comparing it with the original tagged text and computing the accuracy score?
#
# c) Now examine the source code to see how the method is implemented.
# Inspect nltk.tag.api.__file__
# to discover the location of the source code, and open this file using an editor
# (be sure to use the api.py file and not the compiled api.pyc binary file).
def ver19():
    print(
        "a) Taggerinn verður að untagga textann, við sjáum þetta í frumkóðanum:",
        "'tagged_sents = self.tag_sents(untag(sent) for sent in gold)'",
```

```
    )
    print(
        "b) Það er ítrað í gegnum báða listana saman (zip) og heildarfjöldi eins",
        "taggaðra orða er deilt með fjölda orða.",
        "Semsagt t.d. 'sum(x == y for x, y in zip(rett_taggadur, test_taggadur)) / len(test_taggadur)'",
    )
    print("c) Gerði það strax fyrir a) og b), var ekki búinn að lesa c.")


# 20. Write code to search the Brown Corpus for particular words and phrases according to tags,
# to answer the following questions:
# a) Produce an alphabetically sorted list of the distinct words tagged as MD.
# b) Identify words that can be plural nouns or third person singular verbs (e.g. deals, flies).
# c) Identify three-word prepositional phrases of the form IN + DET + NN (eg. in the lab).
# d) What is the ratio of masculine to feminine pronouns?
def ver20():
    tagged_words = brown.tagged_words()

    # a)
    md = sorted(list(set([w for (w, t) in tagged_words if t == "MD"])))
    print("a) Orð sem eru töguð sem MD:", md)

    # b)
    # erum með NNS fyrir nafnorð í fleirtölu og VBZ fyrir þriðju persónu sagnir
    # tíðnidreifing
    cfd = ConditionalFreqDist(tagged_words)
    # skilyrði (orð)
    con = cfd.conditions()
    # sækjum öll orð þar sem skilyrðið inniheldur NNS og VBZ
    pluraln_or_thirdv = [c for c in con if cfd[c]["NNS"] and cfd[c]["VBZ"]]
    # tökum út endurtekningar og sorterum
    pluraln_or_thirdv = sorted(list(set(pluraln_or_thirdv)))
    print(
        "b) Nafnorð í fleirtölu + sagnir í þriðju persónu (úttaksbrot):",
        pluraln_or_thirdv[:20],
    )

    # c)
    t = list(tagged_words)
    # fáum lista af öllum þremur samliggjandi orðum út textanum
    in_dt_nn = list(zip(t, t[1:], t[2:]))
    # filterum út allar þrenndir með tögunum IN DT NN í þessari röð og geymum orðin í streng
    in_dt_nn = [
        (f"{x[0][0]} {x[1][0]} {x[2][0]}")  # orðin þrjú saman í streng
        for x in in_dt_nn
        # tögin IN DT OG NN í þessari röð
        if x[0][1] == "IN" and x[1][1] == "DT" and x[2][1] == "NN"
    ]
    print("c), úttaksbrot:", in_dt_nn[:10])

    # d)
    masculin = ["he", "him", "his", "himself"]
    feminin = ["she", "her", "hers", "herself"]
    masculin_words = [x for x in t if x[0] in masculin]
    feminin_words = [x for x in t if x[0] in feminin]
    print(
```

```
        f"Hlutfallið er: {len(masculin_words)} 3. persóna karlkyn / {len(feminin_words)} 3. persóna kvenkyn.",
        f"Semsagt {prosenta(len(masculin_words) / len(feminin_words))} fleiri karlkynsorð.",
    )


# 24. How serious is the sparse data problem?
# Investigate the performance of n-gram taggers as n increases from 1 to 6.
# Tabulate the accuracy score. Estimate the training data required for these taggers,
# assuming a vocabulary size of 105 and a tagset size of 102.
def ver24():
    train = brown.tagged_sents()[:5000]
    test = brown.tagged_sents()[10000:10102]
    n_list = [1, 2, 3, 4, 5, 6]
    accuracy_list = []
    for n in n_list:
        t = NgramTagger(n, train)
        accuracy_list.append(t.evaluate(test))
    # prentum 'töflu'
    print("Prófum að nota 5000 orða training-sett og 102 orða test-sett")
    print("n   accuracy")
    for n, acc in zip(n_list, accuracy_list):
        print(f"{n}   {acc}")
    print(
        "Hversu stórt training-settið þarf að vera veltur á hvað manni finnst",
        "ásættanlegt í hverju tilviki fyrir sig, mér finnst 0.8 ásættanlegt skor\n",
        "svo ég myndi segja 5000 orða traning-sett eða meira.",
    )


def v(tala):
    print()
    print()
    print(f"Verkefni {tala}")
    print("-----------------------------------------------------------------")


verkefni = ["14", "18", "18_bin", "18_gold", "19", "20", "24"]
for ver in verkefni:
    v(ver)
    locals()[f"ver{ver}"]()
```

2 Úttak

```
Verkefni 14
----------------------------------------------------------------
Úttaksbrot: ["'", "''", '(', '(-HL', ')', ')-HL', '*', '*-HL', '*-NC', '*-TL', ',', ',-HL', ',-NC', ',-TL', '--'


Verkefni 18
----------------------------------------------------------------
a) Hlutfall orða með sama taggið alltaf: 84.4%
b) Fjöldi tvíræðra orða: 8729
19611 66939
c) Hlutfall tvíræðra orða: 29.3%


Verkefni 18_bin
----------------------------------------------------------------
a) Hlutfall orða með sama taggið alltaf: 99.2%
b) Fjöldi tvíræðra orða: 2097


Verkefni 18_gold
----------------------------------------------------------------
a) Hlutfall orða með sama taggið alltaf: 86.4%
b) Fjöldi tvíræðra orða: 4683
11595 41268
c) Hlutfall tvíræðra orða: 28.1%


Verkefni 19
----------------------------------------------------------------
a) Taggerinn verður að untagga textann, við sjáum þetta í frumkóðanum: 'tagged_sents = self.tag_sents(untag(sent
b) Það er ítrað í gegnum báða listana saman (zip) og heildarfjöldi eins taggaðra orða er deilt með fjölda orða.
c) Gerði það strax fyrir a) og b), var ekki búinn að lesa c.


Verkefni 20
----------------------------------------------------------------
a) Orð sem eru töguð sem MD: ['Can', 'Could', 'May', 'Might', 'Must', 'Ought', 'Shall', 'Should', 'Will', 'Would
b) Nafnorð í fleirtölu + sagnir í þriðju persónu (úttaksbrot): ['Aids', 'Designs', 'Increases', 'Makes', 'Report
c), úttaksbrot: ['of this city', 'of this money', 'in each county', 'in this county', 'on this question', 'in th
Hlutfallið er: 16207 3. persóna karlkyn / 4975 3. persóna kvenkyn. Semsagt 325.8% fleiri karlkynsorð.


Verkefni 24
----------------------------------------------------------------
acc
acc
acc
acc
acc
acc
Prófum að nota 5000 orða training-sett og 102 orða test-sett
n   accuracy
1   0.7897631133671743
2   0.10744500846023688
```

3 0.06302876480541456
4 0.0583756345177665
5 0.05710659898477157
6 0.05710659898477157
Hversu stórt training-settið þarf að vera veltur á hvað manni finnst ásættanlegt í hverju tilviki fyrir sig, mér
svo ég myndi segja 5000 orða traning-sett eða meira.