Inngangur að máltækni - Verkefni 4

Guðmundur Óli Norland

1 Kóði

Ég nálgaðist verkefnið þannig að ég bjó til yfirklasa (*Markari*) sem geymir þjálfunar og prófunargögnin, sem og aðferðir sem allir markararnir þurfa sameiginlega. Kallað er á aðferðina *tag_test* til að marka prófunartextann. Hver undirklasi (markari) samanstendur aðallega af smið og mörkunaraðferð þar sem smiðurinn gerir nauðsynlegar ráðstafanir (þjálfun) fyrir viðkomandi markara. Mörkunaraðferðin (*tag*) sér um að marka inntaksorðið. Nánari útskýringar í athugasemdum kóðans.

verkefni4.py

```
import copy
import random
import re
import textwrap
import time

from collections import Counter
from itertools import chain

from nltk import ConditionalFreqDist, FreqDist
from nltk.corpus import brown
from nltk.metrics import accuracy


def read_and_split(filepath):
    tvenndir = []
    texti = open(filepath, "r")
    for line in texti:
        split = line.split("\t")
        if len(split) > 1:
            # orð
            o = split[0]
            # orðflokkur
            flokkur = split[1].rstrip()
            tvenndir.append((o, flokkur))
    # viljum splitta í train og test
    split_index = int(len(tvenndir) * 0.9)
    train = tvenndir[:split_index]  # 90%
    test = tvenndir[split_index:]  # 10%
    return train, test


# ofurklasi
class Markari:
    def __init__(self, train, test, backoff=None):
        self.train = copy.copy(train)
        self.test = copy.copy(test)
        self.backoff = backoff
        # geymum upphaflegu gögnin sem við breytum aldrei
        self.test_original = copy.copy(test)

    """ skilar lista með orðunum úr þjálfunargögnunum """

    def train_words(self, unique=False):
        words = [x[0] for x in self.train]
        if unique:
            words = list(set(words))
        return words
```

```
    """ skilar lista með orðunum úr prufunargögnunum """

    def test_words(self, unique=False):
        words = [x[0] for x in self.test]
        if unique:
            words = list(set(words))
        return words

    """ skilar mörkum úr þjálfunargögnunum """

    def train_tags(self, unique=False):
        tags = [x[1] for x in self.train]
        if unique:
            tags = list(set(tags))
        return tags

    """ skilar mörkum úr prufunargögnunum """

    def test_tags(self, unique=False):
        tags = [x[1] for x in self.test]
        if unique:
            tags = list(set(tags))
        return tags

    # aðferð sem við yfirskrifum í undirklösunum
    def tag(self, tup):
        raise NotImplementedError

    # markar prófunargögnin upp á nýtt eftir þeirri undirklasaaðferð sem verið er að nota
    def tag_test(self):
        for i, x in enumerate(self.test):
            new_tag = None
            try:
                new_tag = self.tag(x)
            # ef það náðist ekki að marka reynum við það með backoffunum
            except:
                backoff = self.backoff
                while backoff:
                    try:
                        new_tag = self.backoff.tag(x)
                        # tókst
                        self.test[i] = new_tag
                        break
                    except:
                        backoff = backoff.backoff
            if new_tag:
                self.test[i] = new_tag

    def evaluate_test(self):
        return accuracy(self.test_original, self.test)


class Sjalfgildismarkari(Markari):
    def __init__(self, train, test, backoff=None):
        super().__init__(train, test, backoff)
        fd = FreqDist(self.train_tags())
```

```
        # algengasta markið
        self.most_common_tag = fd.most_common()[0][0]

    # markar inntakið með algengasta markinu úr öllum þjálfunargögnunum
    def tag(self, tup):
        new_tup = (tup[0], self.most_common_tag)
        return new_tup


class Reglulegur_Markari(Markari):
    def __init__(self, train, test, backoff=None):
        super().__init__(train, test, backoff)
        # regluleg mynstur
        self.patterns = [
            (r"^\.$", "."),
            (r"^\,$", ","),
            (r"^[0-9]+.*", "ta"),
            (r"^og$|^að$|^en$|^sem$|.*ða$|.*að.*|^þe.*", "c"),  # samtenging
            (r"^af$|^me.*|^frá$|^fyrir$|^ge.*", "af"),  # atviksorð sem stýrir þágufalli
            (
                r"^er$|.*ur$|.*ir$|^he.*|.*ar$|",
                "sfg3en",
            ),  # sögn - framsöguháttur - germynd - 3. persóna - eintala - nefnifall
            (r"^um$|^við$|.*ir$|^með$", "ao$"),  # atviksorð, stýrir þolfalli
            (
                r".*ta$|.*ra$|.*ha$|.*fa$|.*na$|.*ga$|.*da$|.*ka$",
                "sng",
            ),  # sögn - nafnháttur - germynd
        ]

    # berum orðið saman við reglulegu mynstrin
    def tag(self, tup):
        word = tup[0]
        for p in self.patterns:
            if re.match(p[0], word):
                return (word, p[1])
        # ekkert fannst
        raise LookupError


class Einstaedumarkari(Markari):
    def __init__(self, train, test, backoff=None):
        super().__init__(train, test, backoff)
        # búum til skilyrta tíðnidreifingu fyrir þjálfunargögnin
        self.cfd = ConditionalFreqDist(self.train)

    # aðferð til að sækja algengasta markið fyrir tiltekið orð
    def most_common_for_word(self, word):
        return self.cfd[word].most_common()[0][0]

    # markar inntakið með algengasta markinu fyrir það orð úr þjálfunargögnunum
    def tag(self, tup):
        new_tup = (tup[0], self.most_common_for_word(tup[0]))
        return new_tup
```

```
class N_Markari(Markari):
    def __init__(self, train, test, n, backoff=None):
        super().__init__(train, test, backoff)
        if n < 2:
            raise Exception("n verður að vera stærra en 1.")
        self.n = n
        # "shiftum" listanum til hægri hverja ítrun svo við getum búið til lista af n-dum með zip aðferðinni
        listar = []
        for i in range(n):
            listar.append(train[i:])
        # geymum lista af öllum n-dunum úr textanum
        self.n_dir = list(zip(*listar))
        # til að geta sótt orð fljótt úr þjálfunargögnunum geymum við öll index hvers orðs
        self.train_index = dict()
        for i, x in enumerate(self.train):
            if x[0] in self.train_index:
                self.train_index[x[0]] += [i]
            else:
                self.train_index[x[0]] = [i]

    def tag(self, tup):
        # orðið sem við viljum marka
        word = tup[0]
        index = self.test.index(tup)
        # sækjum n nýmörkuðu mörkin á undan þessu fyrir prófunargögnin
        n_tags = [x[1] for x in self.test[index - self.n + 1 : index]]
        ## skoðum öll tilvik í þjálfunargögnunum með þessu orði og þessum mörkum á undan
        context = []
        # sækjum öll index fyrir þetta orð
        for i in self.train_index[word]:
            # orðið er í síðasta sæti, svo við mínusum n-ið
            x = self.n_dir[i - self.n + 1]
            # skoðum hvort mörkin séu á undan
            in_context = True
            for j in range(self.n - 1):
                if not x[j][1] == n_tags[j]:
                    in_context = False
            if in_context:
                context.append(x)
        # ef context er autt köstum við villu, backoff taggerinn getur þá séð um þetta
        if not context:
            raise LookupError
        # þurfum bara mörkin í samhenginu
        context_tags = [tuple([y[1] for y in x]) for x in context]
        # finnum hvað er algengasta markið fyrir orðið miðað við samhengi
        fd = FreqDist(context_tags)
        return fd.most_common()[0][0][self.n - 1]


def prenta(markari, heiti):
    print(f"{heiti}-markari:")
    print(f"Brot: {random.sample(markari.test, 5)}")
    print(f"Nákvæmni: {markari.evaluate_test()}")
    print()
```

```
def run():
    train, test = read_and_split("./MIM-GOLD-1_0/MIM-GOLD-1_0/mbl.txt")

    s = Sjalfgildismarkari(train, test)
    s.tag_test()
    prenta(s, "Sjálfgildis")

    r = Reglulegur_Markari(train, test, s)
    r.tag_test()
    prenta(r, "Reglulegur")

    m = Einstaedumarkari(train, test, s)
    m.tag_test()
    prenta(m, "Einstæðu")

    tvi = N_Markari(train, test, 2, m)
    tvi.tag_test()
    prenta(tvi, "Tvístæðu")

    thri = N_Markari(train, test, 3, tvi)
    thri.tag_test()
    prenta(thri, "Þrístæðu")


run()
```

2 Úttak

```
Sjálfgildis-markari:
Brot: [('meðferðarinnar', 'aa'), ('einhæfur', 'aa'), ('daga', 'aa'), ('að', 'aa'), ('flytja', 'aa')]
Nákvæmni: 0.07755709231264073

Reglulegur-markari:
Brot: [(',', ','), ('beinstyrk', 'sfg3en'), ('fram', 'sfg3en'), ('hamingjuna', 'sfg3en'), ('rósmaríninu', 'sfg3e
Nákvæmni: 0.19447571566420072

Einstæðu-markari:
Brot: [('er', 'sfg3en'), ('Ensku', 'aa'), ('stað', 'nkeo'), ('markmið', 'nhen'), ('að', 'cn')]
Nákvæmni: 0.6771067867481505

Tvístæðu-markari:
Brot: ['aþ', ('væru', 'svg3fþ'), 'nvfo', ('aukið', 'lheosf'), ('bækur', 'nvfn')]
Nákvæmni: 0.7240270183338694

Þrístæðu-markari:
Brot: [('undirleiks', 'nkee'), ('netfyrirtækja', 'nhfe'), ('litlar', 'lvfosf'), ('eða', 'c'), ('1861-1939', 'ta'
Nákvæmni: 0.9523158571888067
```