Inngangur að máltækni - Verkefni 8

Guðmundur Óli Norland

Hjálparaðferðir

Hér koma aðferðir sem ég skrifaði til að hjálpa mér við lausn verkefnanna:

Aðferð sem les ordmyndir.txt skránna og skilar lista með þeim

orðmyndir

```
def ordmyndir(filepath):
    with open(filepath, "r") as ordmyndaskra:
        ordmyndir = [line.rstrip() for line in ordmyndaskra]
    return ordmyndir
```

Aðferð sem tilreiðir streng og skilar lista af tokens

Tilreiða

```
def tilreida(strengur):
    tokens = word_tokenize(strengur)
    return tokens
```

Mikilvægasta aðferðin, býr til html streng með hjálp BeautifulSoup pakkans. Ítrað er í gegnum tokenin og tékkað hvort það sé næsta villan í röðinni (curr_villu_index breytan). Ef tokenið er villa er það litað rautt (klasinn villa). Valkvæmt er að senda inn lista af leiðréttingum, þá ef það finnst villa er skoðað hvort það séu til leiðréttingar fyrir villuna. Ef það eru leiðréttingar er ein leiðrétting valin af handahófi, allar eru hins vegar settar í tooltip.

HTML

```
def html(tokens, villur, leidrettingar=None):
    base = "<!DOCTYPE html><html><head></head><body></body></html>"
    soup = bs4(base, features="html.parser")
    body = soup.find("body")
    # Fyrirsögn
    texti_tag = soup.new_tag("h1")
    if leidrettingar:
        texti_tag.string = (
            "Tilreiddur texti (villur merktar með rauðu, leiðréttingar með grænu):"
        )
    else:
        texti_tag.string = "Tilreiddur texti (villur merktar með rauðu):"
    body.append(texti_tag)
    # ítrum í gegnum tokenin og merkjum villur með rauðu
    curr_villu_index = 0
    for t in tokens:
        t_strengur = t
        if not t == villur[curr_villu_index]:
            tag = soup.new_tag("span", attrs={"class": "texti"})
        else:
            # skoðum fyrst hvort það sé til leiðrétting fyrir villuna
            if leidrettingar and (
                not leidrettingar[curr_villu_index] == villur[curr_villu_index]
            ):
                tag = soup.new_tag(
                    "span",
                    attrs={
                        "class": "texti leidretting",
```

```
                        "title": ", ".join(leidrettingar[curr_villu_index]),
                    },
                )
                # veljum eina leiðréttingu af handahófi
                t_strengur = random.choice(leidrettingar[curr_villu_index])
            else:
                tag = soup.new_tag("span", attrs={"class": "texti villa"})
            curr_villu_index += 1
        tag.string = t_strengur
        body.append(tag)
    # css stílar
    head = soup.find("head")
    css = soup.new_tag("style", type="text/css")
    css.string = "body { margin: 2rem; }"
    css.string += ".texti { font-size: 1.5rem; font-weight: bold; }"
    css.string += ".villa { color: red; }"
    css.string += ".leidretting { color: green; }"
    head.append(css)
    # skilum fallegum html texta
    return soup.prettify()
```

Aðferð sem tekur við streng á HTML sniði og skrifar út HTML skrá.

HTML skrá

```
def html_string_to_file(filepath, strengur):
    with open(filepath, "w+") as html_skra:
        for line in strengur:
            html_skra.write(line)
```

Aðferð sem tekur við tveimur strengjum á HTML sniði og skeitir þeim saman (bætir tögunum undir body taginu í seinni strengnum við þann fyrsta).

Skeita HTML

```
def skeita_html(html1, html2):
    soup1 = bs4(html1, features="html.parser")
    soup2 = bs4(html2, features="html.parser")
    body = soup1.find("body")
    children = soup2.find("body").findChildren()
    for child in children:
        body.append(child)
    return soup1.prettify()
```

Aðferð sem síar niður orðmyndalistann til að einfalda breytingarfjarlægðarleitina. Segjum t.d. að við værum með orð sem er 4 stafir og með fjarlægðina 2. Þá vitum við að niðurstöðurnar okkar eru orð á bilinu 2-6 stafir, getum nýtt okkur það til að sía. Aðferðin skilar dictionary þar sem lyklarnir eru á forminu lengd:fjarlægð.

Orðmyndir filtered

```
def ordmyndir_filtered(
    ordmyndir,
    lengdir=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    distances=[1, 2],
):
    filtered = dict()
```

```
    for lengd in lengdir:
        for distance in distances:
            # fyrir hverja lengd + distance geymum við allar mögulegar orðmyndir sem edit_distance getur notað
            index_strengur = f"{lengd}:{distance}"
            filtered[index_strengur] = [
                o
                for o in ordmyndir
                if len(o) >= lengd - distance and len(o) <= lengd + distance
            ]
    return filtered
```

Aðferð sem finnur svipaða strengi innan ákveðinnar breytingarfjarlægðar með hjálp nltk.edit_distance aðferðarinnar. Gerir ráð fyrir að orðmyndalistinn hafi verið síaður með hjálp aðferðarinnar hér fyrir ofan.

HTML skrá

```
def edit_distance_listi(token, ordmyndir_filtered_dict, distance):
    token_lengd = len(token)
    index_strengur = f"{token_lengd}:{distance}"
    listi = list(
        set(
            [
                t
                for t in ordmyndir_filtered_dict[index_strengur]
                if edit_distance(token, t) == distance
            ]
        )
    )
    print(listi)
    return listi
```

Verkefnið

Aðferð sem notar hjálparföllin til að leysa verkefnið. Tokens eru útbúin útfrá texta, svo eru villur og leiðréttingar sóttar með beytingu hjálparfallana. Ég ákvað að velja texta sem er ekkert svo vel skrifaður svo ég fór bara á fésbókina og fann eitthvað sullumbull.

HTML skrá

```
def main():
    ordmyndir_listi = ordmyndir("./ordmyndalisti.txt")
    ordmyndir_filtered_dict = ordmyndir_filtered(ordmyndir_listi)
    with open("./kattadót.txt", "r") as skra:
        texti = skra.read()
    tokens = tilreida(texti)
    villur = [token for token in tokens if token not in ordmyndir_listi]
    # html án leiðréttinga
    html_strengur = html(tokens, villur)
    print(html_strengur)

    # búum til leiðréttingalista sem er jafnstór og villulistinn
    leidrettingar = []
    if villur:
        for villa in villur:
            einn = edit_distance_listi(villa, ordmyndir_filtered_dict, 1)
            # ef það eru til breytingafjarlægðir að lengd 1
            if einn:
                leidretting = einn
            else:
                tveir = edit_distance_listi(villa, ordmyndir_filtered_dict, 2)
                # ef það eru til breytingarfjarlægðir að lengd 2
                if tveir:
                    leidretting = tveir
                else:
                    # ekki til breytingarfjarlægðir að lengd 1 eða 2
                    leidretting = villa
            leidrettingar.append(leidretting)

    fjoldi_villna = len(villur)
    fjoldi_leidrettinga = len(
        [x for i, x in enumerate(leidrettingar) if villur[i] != x]
    )
    print("Fjöldi villna:", fjoldi_villna)
    print("Fjöldi leiðréttinga:", fjoldi_leidrettinga)

    # html með leiðréttingum
    html_strengur_leidrettur = html(tokens, villur, leidrettingar)
    html_strengur_allt = skeita_html(html_strengur, html_strengur_leidrettur)
    html_string_to_file("./máltækni-verkefni-8.html", html_strengur_allt)
```

Úr úttakinu fékk ég:

Fjöldi villna: 124 Fjöldi leiðréttinga: 116

HTML úttakið má finna í HTML skránni í möppu verkefnisins.