Додаток А

Програмний код preprocessor.py

```
import pandas as pd

import re

import matplotlib.pyplot as plt

import os

def dotsub(lines):
    lines = re.sub('\.\.', ' . . ', lines)
    lines = re.sub('(?<=[a-zA-Z_\?])\.(?=[a-zA-Z_\.])', ' . ',
       lines)
    return lines

def bracketsub(lines):
    lines = re.sub('{', '{ ', lines)
    lines = re.sub('\(', ' ( ', lines)
    lines = re.sub('\[', ' [ ', lines)
    return lines

def ltsub(lines):
    return re.sub('<[^=-]', ' < ', lines)

def gtsub(lines):
    return lines = re.sub('(?<=[^=])>', ' > ', lines)

def equalsub(lines):
```

```
27      return re.sub('=[^=>]', '=␣', lines)

29  def pointersub(lines):
        return re.sub('-[^>]', '-␣', lines)
31
    def plussub(lines):
33      lines = re.sub('\+\+', '␣++␣', lines)
        lines = re.sub('\+[^\+]', '␣+␣', lines)
35      return lines

37  def minussub(lines);
        lines = re.sub('--', '␣--␣', lines)
39      lines = re.sub('-[^-]', '␣-␣', lines)
        return lines
41
    def slashsub(lines):
43      lines = re.sub('\*', '␣*␣', lines)
        lines = re.sub(r'\\', '␣\␣', lines)
45      return lines

47  def questsub(lines):
        return re.sub('\?[^\.]', '␣?␣', lines)
49
    def spacesub(lines):
51      return re.sub('\s+', '␣', lines)

53  def punctsub(lines):
        lines = re.sub(':', '␣:␣', lines)
55      lines = re.sub(';', '␣;␣', lines)
        lines = re.sub('"', '␣"␣', lines)
57      return lines

59  def main():
        root_dir = os.getcwd()
61
        directories = ['profilereader-main', 'unitychanspringbone', '
           unityrenderstreaming', 'anotherthread-game', 'fpssample-
           game', 'unitycsreference', 'waveshooter-demo']
63



65
        directory_path = 'unitycsreference'
67      for file in os.listdir(f'{directory}/source/{directory_path}'
```

```
        ):
            filename, _ = os.path.splitext(file)

            with open(f'{directory}/source/{directory_path}/{file}')
                as f:
                lines = []
                for line in f.readlines():
                    if not line.strip().startswith('/'):
                        line = re.sub(r'.*?\/(.*)$', '', line)
                        lines.append(line)

            lines = '␣'.join(lines).replace('\n', '␣\n␣').replace(','
                , '␣').replace('@', '').replace(')', '').replace('}',
                '').replace(']', '')

            lines = dotsub(lines)
            lines = bracketsub(lines)

            lines = ltsub(lines)

            lines = gtsub(lines)

            lines = equalsub(lines)

            lines = pointersub(lines)

            lines = plussub(lines)

            lines = minussub(lines)

            lines = slashsub(lines)

            lines = questsub(lines)

            lines = spacesub(lines)

            with open(f'{directory}/corpus/unity_code/{filename}.txt'
                , 'w') as f:
                f.write(lines)

if __name__=='__main__':
    main()
```

Додаток Б

Програмний код corpus_analyser.py

```
import pandas as pd

from scipy import stats
import numpy as np

import os

import matplotlib.pyplot as plt


directory = os.getcwd()
folder = 'processed'

data_inside = {}

for root, dirs, files in os.walk(f'{directory}/{folder}'):
    for file in files:
        filename, extension = os.path.splitext(file)
        if extension == '.txt':
            print(file)
            vocab_counter = 0
            length_counter = 0
            vocab = []
            with open(f'{root}/{file}', 'r') as f:
                text = f.read().split(' ')
                for word in text:
                    if not word in vocab:
```

```
                    vocab.append(word)

            data_inside[file] = (len(text), len(vocab))


data = pd.DataFrame.from_dict(data_inside, orient='index').
    reset_index()
data = data.rename(columns={'index': 'name', 0: 'length', 1: '
    vocab_size'})

data = data.sort_values(by=['length'], ignore_index=True)

fig, ax = plt.subplots(dpi=300)

ax.set(xscale='log', yscale='log', xlabel=r'$L$', ylabel=r'$V$')

ax.scatter(data['length'], data['vocab_size'], edgecolor='black',
     facecolor='white')

plt.savefig(f'{directory}/figures/unity-corpuse.png')

alpha_1 = []
alpha_2 = []
gamma   = []

windows = [i for i in range(5, 300)]

for window in windows:
    step = window
    ls = data['length'].rolling(window=window, step=step).mean()
    vs = data['vocab_size'].rolling(window=window, step=step).
        mean()
    dv = data['vocab_size'].rolling(window=window, step=step).std
        ()

    slope, intercept, r, p, se = stats.linregress(np.log(ls.
        dropna()), np.log(vs.dropna()))

    alpha_1.append(slope)

    slope, intercept, r, p, se = stats.linregress(np.log(ls.
        dropna()), np.log(dv.dropna()))
```

```
64
        alpha_2.append(slope)
66
        slope, intercept, r, p, se = stats.linregress(np.log(vs.
            dropna()), np.log(dv.dropna()))
68
        gamma.append(slope)
70
window = 50
72 step = window

74 ls = data['length'].rolling(window=window, step=step).mean()
vs = data['vocab_size'].rolling(window=window, step=step).mean()
76 dv = data['vocab_size'].rolling(window=window, step=step).std()

78 r_symbol = r'$r = $'
alpha1_symbol = r'$\alpha_1 = $'
80 alpha2_symbol = r'$\alpha_2 = $'
gamma_symbol = r'$\gamma = $'
82
slope, intercept, r, p, se = stats.linregress(np.log(ls.dropna())
    , np.log(vs.dropna()))
84
print(slope, intercept, r)
86
fig, ax = plt.subplots(dpi=300)
88
ax.set(xscale='log', yscale='log', xlabel=r'$L$', ylabel=r'$V$')
90
ax.scatter(ls, vs, color='black', label='data')
92 ax.plot(ls.dropna().tolist(),  np.e**intercept * ls.dropna().
    to_numpy()**slope, '--', color='red', label='fit')

94 ax.text(x=70, y=120, s=f'{gamma_symbol}{slope:.3},\n{r_symbol}{r
    :.3}')

96 ax.legend()

98 plt.savefig(f'{directory}/figures/songs_window50_step50_VL.png')

100 slope, intercept, r, p, se = stats.linregress(np.log(ls.dropna())
    , np.log(dv.dropna()))
```

```
102 print(slope, intercept, r)

104 fig, ax = plt.subplots(dpi=300)

106 ax.set(xscale='log', yscale='log', xlabel=r'$L$', ylabel=r'$\
    Delta␣V$')

108 ax.scatter(ls, dv, color='black', label='data')
    ax.plot(ls.dropna().tolist(),  np.e**intercept * ls.dropna().
    to_numpy()**slope, '--', color='red', label='fit')
110
    ax.text(x=100, y=105, s=f'{alpha1_symbol}{slope:.3},\n␣{r_symbol
    }{r:.3}')
112
    ax.legend()
114
    plt.savefig(f'{directory}/figures/songs_window50_step50_dVL.png')
116
    slope, intercept, r, p, se = stats.linregress(np.log(vs.dropna())
    , np.log(dv.dropna()))
118
    print(slope, intercept, r)
120
    fig, ax = plt.subplots(dpi=300)
122
    ax.set(xscale='log', yscale='log', xlabel=r'$V$', ylabel=r'$\
    Delta␣V$')
124
    ax.scatter(vs, dv, color='black', label='data')
126 ax.plot(vs.dropna().tolist(),  np.e**intercept * vs.dropna().
    to_numpy()**slope, '--', color='red', label='fit')

128 ax.text(x=100, y=105, s=f'{alpha2_symbol}{slope:.3},\n␣{r_symbol
    }{r:.3}')

130 ax.legend()

132 plt.savefig(f'{directory}/figures/songs_window50_step50_dVV.png')
```

Додаток В

Програмний код static_characteristics.py

```
import os

import pandas as pd
import numpy as np
from scipy import stats

import matplotlib.pyplot as plt

from collections import Counter

directory = os.getcwd()

folder = 'processed'
filename = 'EditorGUI'

with open(f'{directory}/{folder}/{filename}.txt', 'r') as file:
    lines = file.read()


text = lines.split(' ')

unique = Counter(text)

vocab = pd.DataFrame.from_dict(unique, orient='index').
   reset_index()
vocab = vocab.rename(columns={"index": "word", 0: "freq"})
vocab = vocab.sort_values(by='freq', ascending=False,
```

```
    ignore_index=True)
vocab.index = vocab.index + 1
vocab['norm_freq'] = vocab['freq'] / vocab['freq'].sum()

start = 50
end = 5000

slope, intercept, r, p, se = stats.linregress(x=np.log(vocab.
    index[start:end]), y=np.log(vocab['freq'][start:end]))

print(slope, np.e**intercept, r)

fig, ax = plt.subplots(dpi=300)

ax.set(xscale='log', yscale='log', xlabel=r'$r$', ylabel=r'$f(r)$
    ')

ax.scatter(vocab.index, vocab['norm_freq'], edgecolor='black',
    facecolor='white', label='data')
ax.plot(vocab.index.to_numpy(), np.e**intercept * vocab.index.
    to_numpy()**slope, '--', color='red', label='Zipf')

ax.legend()

plt.savefig(f'{directory}/figures/{filename}_zipfs-law.png')

probs, bins = np.histogram(vocab['norm_freq'], bins='fd')

pdf = pd.DataFrame()
pdf['bins'] = bins[:-1]
pdf['probs'] = probs / len(vocab['norm_freq'])

pdf = pdf[pdf['probs'] != 0].reset_index(drop=True)
pdf.index = pdf.index + 1

start_rank = 5
end_rank = 250

slope, intercept, r, p, se = stats.linregress(x=np.log(pdf['bins'
    ][start_rank:end_rank]), y=np.log(pdf['probs'][start_rank:
    end_rank]))

print(slope, np.e**intercept, r)
```

```
63
   fig, ax = plt.subplots(dpi=300)
65
   ax.set(xscale='log', yscale='log', xlabel=r'$f$', ylabel=r'$p(f)$
      ')
67
   ax.scatter(pdf['bins'], pdf['probs'], edgecolor='black',
      facecolor='white')
69 ax.plot(pdf['bins'][:end_rank].to_numpy(), np.e**intercept * pdf[
      'bins'][:end_rank].to_numpy()**slope, '--', color='red', label
      ='Zipf')

71 ax.legend()

73 plt.savefig(f'{directory}/figures/{filename}_pdf.png')

75 pdf['cdf'] = np.cumsum(pdf['probs'])
   pdf['ccdf'] = 1 - pdf['cdf']
77
   slope, intercept, r, p, se = stats.linregress(x=np.log(pdf['bins'
      ][5:end_rank]), y=np.log(pdf['ccdf'][5:end_rank]))
79
   print(slope, np.e**intercept, r)
81
   fig, ax = plt.subplots(dpi=300)
83
   ax.set(xscale='log', yscale='log', xlabel=r'$f$', ylabel=r'$P(f)$
      ', ylim={1e-5, 2})
85
   ax.scatter(pdf['bins'], pdf['ccdf'], edgecolor='black', facecolor
      ='white')
87 ax.plot(pdf['bins'].to_numpy(), np.e**intercept * pdf['bins'].
      to_numpy()**slope, '--', color='red', label='Zipf')

89 ax.legend()

91 plt.savefig(f'{directory}/figures/{filename}_ccdf.png')
```

Додаток Г

Програмний код heaps.py

```
import os

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy import stats

root = os.getcwd()
corpus_path = 'corpus/mergers'
figure_path = 'figures'
textname = 'songs'

with open(f'{root}/{corpus_path}/{textname}.txt', 'r') as file:
    lines = file.read()

text = lines.split(' ')

lengths = []
vocabs = []

step = 100

for i in range(100, len(text), step):
    lengths.append(i)
    vocabs.append(len(set(text[:i])))


print(f'Length of text in words is: {len(text)}')
```

```
print(f'Vocabulary size is: {len(set(text))}')

slope, intercept, r, p, se = stats.linregress(x=np.log(lengths[
    start:]), y=np.log(vocabs[start:]))

print(f'slope is {slope:.4}')
print(f"Pearson's correlation is {r:.4}")

fig, ax = plt.subplots(dpi=300)

ax.set(xscale='log', yscale='log', xlabel=r'$L$', ylabel=r'$V$')

ax.scatter(lengths[::], vocabs[::], facecolor='white', edgecolor=
    'black', label='data')
ax.plot(lengths, np.e**intercept * lengths**slope, '--', color='
    red', label=f'Heaps')

ax.legend()

plt.show()

plt.savefig(f'{root}/{figure_path}/{textname}-heaps.png')
```

Додаток Д

Програмний код merger.py

```
import os

import shutil

home = os.getcwd()
corpus_folder = 'corpus'
texts_folder = 'unity_code'

with open(f'{home}/{corpus_folder}/mergers/{texts_folder}.txt','wb') as destination_file:
    for filename in os.listdir(f'{home}/{corpus_folder}/{texts_folder}'):
        with open(f'{home}/{corpus_folder}/{texts_folder}/{filename}','rb') as current_file:
            shutil.copyfileobj(current_file, destination_file)
```