

Додаток А

Програмний код preprocessor.py

```
1 import pandas as pd

3 import re

5 import matplotlib.pyplot as plt

7 import os

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

13

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

19

20 def ltsub(lines):
21    return re.sub('<[^\=]', '._<', lines)

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

25

26 def equalsub(lines):
```

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

29 def pointersub(lines):
    return re.sub('[^>]', '-', lines)

31
32 def plussub(lines):
33     lines = re.sub('\+\+', '+', lines)
34     lines = re.sub('\+[\^\\+]', '+', lines)
35     return lines

37 def minussub(lines):
    lines = re.sub('--', '-_', lines)
39     lines = re.sub('[^~]', '~_', lines)
    return lines

41
42 def slashsub(lines):
43     lines = re.sub('\*', '*_', lines)
44     lines = re.sub(r'\\', '\\_', lines)
45     return lines

47 def questsub(lines):
    return re.sub('\?[^\.\.]', '?_', lines)

49
50 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
62     directories = ['profilereader-main', 'unitychanspringbone',
63                   'unityrenderstreaming', 'anotherthread-game', 'fpssample-
64                   game', 'unitycsreference', 'waveshooter-demo']

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

```
):  
    filename, _ = os.path.splitext(file)  
69  
    with open(f'{directory}/source/{directory_path}/{file}')  
        as f:  
71        lines = []  
        for line in f.readlines():  
73            if not line.strip().startswith('/'):  
                line = re.sub(r'.*?\/(.*)$', '', line)  
75                lines.append(line)  
  
77        lines = '\n'.join(lines).replace('\n', '\n\n').replace(',',  
            , '\n').replace('@', '').replace(')', '').replace('}',  
            '').replace(']', '')  
  
79        lines = dotsub(lines)  
        lines = bracketsub(lines)  
  
81        lines = ltsub(lines)  
  
83        lines = gtsub(lines)  
  
85        lines = equalsub(lines)  
  
87        lines = pointersub(lines)  
  
89        lines = plussub(lines)  
  
91        lines = minussub(lines)  
  
93        lines = slashsub(lines)  
  
95        lines = questsub(lines)  
  
97        lines = spacesub(lines)  
  
99        with open(f'{directory}/corpus/unity_code/{filename}.txt',  
            , 'w') as f:  
101            f.write(lines)  
  
103 if __name__ == '__main__':  
    main()
```

Додаток Б

Програмний код `corpus_analyser.py`

```
import pandas as pd
2
from scipy import stats
4 import numpy as np

6 import os

8 import matplotlib.pyplot as plt

10
directory = os.getcwd()
12 folder = 'processed'

14 data_inside = {}

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

```
28         vocab.append(word)

30

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

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

36

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

40

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

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

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

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

50

52 alpha_1 = []
54 alpha_2 = []
56 gamma   = []

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

60

62 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
67     slope, intercept, r, p, se = stats.linregress(np.log(vs.
68         dropna()), np.log(dv.dropna()))
69
70     gamma.append(slope)
71
72     window = 50
73     step = window
74
75     ls = data['length'].rolling(window=window, step=step).mean()
76     vs = data['vocab_size'].rolling(window=window, step=step).mean()
77     dv = data['vocab_size'].rolling(window=window, step=step).std()
78
79     r_symbol = r'$r_{\square}$'
80     alpha1_symbol = r'$\alpha_{1\square}$'
81     alpha2_symbol = r'$\alpha_{2\square}$'
82     gamma_symbol = r'$\gamma_{\square}$'
83
84     slope, intercept, r, p, se = stats.linregress(np.log(ls.dropna())
85         , np.log(vs.dropna()))
86
87     print(slope, intercept, r)
88
89     fig, ax = plt.subplots(dpi=300)
90
91     ax.set(xscale='log', yscale='log', xlabel=r'$L$', ylabel=r'$V$')
92
93     ax.scatter(ls, vs, color='black', label='data')
94     ax.plot(ls.dropna().tolist(), np.e**intercept * ls.dropna().
95         to_numpy()**slope, '--', color='red', label='fit')
96
97     ax.text(x=70, y=120, s=f'{gamma_symbol}{slope:.3},\n_{r_symbol}{r
98         :.3}')
99
100    ax.legend()
101
102    plt.savefig(f'{directory}/figures/songs_window50_step50_VL.png')
103
104    slope, intercept, r, p, se = stats.linregress(np.log(ls.dropna())
105        , 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`

```
1 import os

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

7 import matplotlib.pyplot as plt

9 from collections import Counter

11 directory = os.getcwd()

13 folder = 'processed'
  filename = 'EditorGUI'

15

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

19

21 text = lines.split('\n')

23 unique = Counter(text)

25 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)
27 vocab.index = vocab.index + 1
    vocab['norm_freq'] = vocab['freq'] / vocab['freq'].sum()
29
    start = 50
31 end = 5000

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

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

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

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

41 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')
43
    ax.legend()
45
    plt.savefig(f'{directory}/figures/{filename}_zipfs-law.png')
47
    probs, bins = np.histogram(vocab['norm_freq'], bins='fd')
49
    pdf = pd.DataFrame()
51 pdf['bins'] = bins[:-1]
    pdf['probs'] = probs / len(vocab['norm_freq'])
53
    pdf = pdf[pdf['probs'] != 0].reset_index(drop=True)
55 pdf.index = pdf.index + 1

57 start_rank = 5
    end_rank = 250
59
    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]))
61
    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`

```
2 import os

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

8 from scipy import stats

10 root = os.getcwd()
   corpus_path = 'corpus/mergers'
12 figure_path = 'figures'
   textname = 'songs'
14
   with open(f'{root}/{corpus_path}/{textname}.txt', 'r') as file:
16       lines = file.read()

18 text = lines.split('\n')

20 lengths = []
   vocabs = []
22
   step = 100
24
   for i in range(100, len(text), step):
26       lengths.append(i)
         vocabs.append(len(set(text[:i])))
28

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

```
print(f'Vocabulary_size_is_{len(set(text))}')
32 slope, intercept, r, p, se = stats.linregress(x=np.log(lengths[
    start:]), y=np.log(vocabs[start:]))
34 print(f'slope_is_{slope:.4}')
36 print(f"Pearson's_correlation_is_{r:.4}")

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

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

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

44 ax.legend()

46 plt.show()

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

Додаток Д

Програмний код merger.py

```
import os
2
import shutil
4
home = os.getcwd()
6 corpus_folder = 'corpus'
  texts_folder = 'unity_code'
8
with open(f'{home}/{corpus_folder}/mergers/{texts_folder}.txt', '
  wb') as destination_file:
10   for filename in os.listdir(f'{home}/{corpus_folder}/{
    texts_folder}'):
        with open(f'{home}/{corpus_folder}/{texts_folder}/{
          filename}', 'rb') as current_file:
12         shutil.copyfileobj(current_file, destination_file)
```