Dokumentace k úlohám na zkoušku z Úvodu do programování (ZS 2023/2024)
Vypracoval: Bc. Jakub Kraft

Úloha 1 – Výpočet počtu znaků, slov, vět v textu (5)

1. Zadání úlohy

Pro vstupní text zahrnující písmena “A-Ž”, “a-ž”, číslovky “0-9”, speciální znaky “,.?!;” oddělené mezerami, určete celkový počet znaků (včetně mezer, bez mezer), počet slov a počet vět. Pokud budou v textu nepodporované znaky, ignorujete je. Vstupní data načtěte z textového souboru, výslednou statistiku uložte do textového souboru.

2. Rozbor problému

Hlavním cílem programu je extrahovat kvantitativní informace z textového souboru. Mezi zvolené ukazatele o textovém souboru patří počet znaků (včetně mezer i bez mezer), slov a vět. Poskytuje tak základní přehled o textu, které běžný uživatel počítače běžně využívá. Program je možné dále modifikovat, takže jeho využití může být i mnohem širší.

Je nutné vzít v potaz to, že program je přizpůsoben českému jazyku a jeho specifikům. Za zřejmé specifikum se dá považovat některá diakritická znaménka, tedy háčky a čárky. Tyto znaky musí být správně identifikovány a zpracovány, protože mohou mít významový dopad na slova. Dalším příkladem může být problematika slov nebo slovních spojení, které jsou na sebe napojené spojovníkem, jako například Frýdek-Místek, Rakousko-Uhersko, kulturně-politický, česko-německý slovník, 3-methylpentan, ping-pong, chceme-li (Wikipedie 2023). Počet slov u těchto příkladů by mohl být diskutabilní, ale vzhledem k zadání však tato problematika řešena není. Je také nezbytné se vypořádat s nepodporovanými znaky, což je klíčové pro přesnost analýzy. Při psaní kódu je samozřejmě důležité, aby se program držel základních principů programování.

3. Struktura a popis programu

Byl použit importovaný modul **`re`**, což je standardní Python modul pro práci s regulárními výrazy. Regulární výrazy se používají pro vyhledávání a manipulaci s textem na základě specifických vzorů (Python Docs 2024).

```
class TextAnalyzer:
    def __init__(self, file_path):
        with open(file_path, 'r', encoding='utf-8') as file:
            self.text = re.sub(r'[^A-Ža-ž0-9 ,.?!;]', '', file.read())
```

Nejprve byla definovaná třída **`TextAnalyzer`**. Metoda **`def __init__`** je parametrický inicializátor, který inicializuje instanci třídy s textem. Cesta k souboru s textem je v parametru **`file_path`**.

```
with open(file_path, 'r', encoding='utf-8') as file
```

Otevírá soubor pro čtení v režimu **`'r'`**(čtení) s explicitním určením kódování **`'utf-8'`** pro správné čtení českých znaků.

```
self.text = re.sub(r'[^A-Ža-ž0-9 ,.?!;]', '', file.read())
```

Čte celý text souboru a očistí ho od nevyžádaných znaků a výsledný textový soubor uloží do proměnné **`self.text`**.

```
def count_characters(self, include_spaces=True):
    if include_spaces:
        return len(self.text)
    else:
        return len(self.text.replace(' ', ''))
```

Tato metoda počítá počet znaků v textu. Parametr **include_spaces** určuje, zda se mají započítat i mezery. **len(self.text)** vrací celkový počet znaků v textu včetně mezer, **len(self.text.replace(' ', ''))** nejprve odstraní mezery z textu a pak vrátí počet zbývajících znaků bez mezer.

```
def count_words(self):
    words = self.text.split()
    return len(words)
```

Metoda **count_words** má za cíl spočítat počet slov. **split()** rozdělí text na slova na základě mezer. Výsledný seznam slov je uložen v proměnné **words**.

```
def count_sentences(self):
    sentences = re.split(r'[.?!]', self.text)
    sentences = [s for s in sentences if s.strip()]
    return len(sentences)
```

Účelem metody **count_sentences** je vrácení počtu vět v textu. **re.split(r'[.?!]', self.text)** rozdělí text na věty pomocí regulárního výrazu, který hledá tečky, otazníky a vykřičníky. **sentences = [s for s in sentences if s.strip()]** odstraňuje ze seznamu vět prázdné celky nebo jen mezery.

```
def write_statistics(self, output_file):
    with open(output_file, 'w', encoding='utf-8') as file:
        file.write(f"Celkový počet znaků (včetně mezer):
{self.count_characters(include_spaces=True)}\n")
        file.write(f"Celkový počet znaků (bez mezer):
{self.count_characters(include_spaces=False)}\n")
        file.write(f"Počet slov: {self.count_words()}\n")
        file.write(f"Počet vět: {self.count_sentences()}\n")
```

Poslední metoda **write_statistics** ukládá výsledky do souboru. Soubor je otevřen v režimu **'w'** pro zápis a opět s kódováním **'utf-8'**. Řádky **file.write** zapisují výsledky jednotlivých výpočtů do souboru s tím, že každý výsledek je na novém řádku.

```
input_file = 'C:\\Users\\Jakub\\Desktop\\2. Mgr\\Úvod do
programování\\Text.txt'
# Zde napište přesné umístění textového souboru, který má program
analyzovat
output_file = 'C:\\Users\\Jakub\\Desktop\\2. Mgr\\Úvod do
programování\\Text_counter.txt'
# Zde napište přesné umístění, kde má být textový soubor s výsledky uložen

analyzer = TextAnalyzer(input_file)
analyzer.write_statistics(output_file)
```

Tato část kódu je zaměřena na jeho použití. Zde musí být nadefinovány cesty k vstupnímu textovému souboru a výstupnímu textovému souboru s výsledky.

```
analyzer = TextAnalyzer(input_file)
```

Vytvoří instanci `TextAnalyzer` s cestou k vstupnímu souboru.

```
analyzer.write_statistics(output_file)
```

Spustí metodu `write_statistics` pro zápis statistik do výstupního souboru.

4. Popis vstupních a výstupních dat

Text pro analýzu je načten z textového souboru s příponou `.txt`. Vstupní data jsou tedy datového typu `string`. Program je přizpůsoben českému jazyku, takže pro správné výsledky by měl být text napsán v češtině. Jak bylo již uvedeno výše, vstupní data zahrnují písmena "A-Ž", "a-ž", číslovky "0-9" a speciální znaky ",.?!;". To jsou znaky podporované. Naopak nepodporovanými znaky jsou považovány jakékoliv jiné znaky a jsou v programu ignorovány nebo odstraněny.

Výstupní soubor obsahuje výsledky analýzy textu a je také ve formátu `.txt`. Výsledky zahrnují 4 řádky, které obsahují celkový počet znaků (včetně mezer), celkový počet znaků (bez mezer), počet slov a počet vět. Za každou z těchto kategorií se nachází ": " a vypočítaná hodnota.

Příklad:

Celkový počet znaků (včetně mezer): 47

Celkový počet znaků (bez mezer): 39

Počet slov: 8

Počet vět: 3

5. Problematická místa a možná vylepšení

Za nejvíce problematickou část se dají považovat vstupní data. Program rozděluje jednotlivé věty na základě přítomnosti tečky, vykřičníku nebo otazníku. Z toho pak vyplívají některé nesrovnalosti programu s realitu, protože tečka může být použita i ve zkratkách, nebo se může některé z těchto tří interpunkčních znamínek vyskytovat v přímé řeči, kde by nutně nemusely značit konec větného celku. Ukazatel počtu vět je poměrně problematický i v tom, že počítá pouze větné celky. Nerozlišuje mezi větou jednoduchou a souvětím o vyšším počtu jednoduchých vět. Vylepšení určení počtu větných celků a vytvoření nového ukazatele počtu vět jednoduchých můžou být dvě potenciální vylepšení. K tomu by ale bylo zapotřebí použití mnohem složitějších metod, které by zahrnovali metody z oblasti Natural Language Processing, jakožto oblast umělé inteligence, která je zaměřena na schopnost počítače porozumět lidskému jazyku a pak s ním dále pracovat (IBM 2024).

Jak již bylo zmíněno v kapitole 2, nejasnosti mohou budit i některá složená slova spojená spojovníkem. V tomto případě by bylo vhodné pro případné vylepšení použití složitější tokenizace. Zřejmou nedokonalostí je i omezený počet podporovaných znaků, které program počítá. Není ale možno pokrýt potřeby všech uživatelů, takže je zde spíš prostor pro individuálnější modifikace.

Dalším aspektem, který by se mohl vylepšit, je grafická prezentace programu. Zdokonalení grafického uživatelského rozhraní je například možné s použitím knihovny TkInter (Python Wiki 2024).

6. Zdroje

WIKIPEDIE (2023): Spojovník. Dostupné z: https://cs.wikipedia.org/wiki/Spojovn%C3%ADk [cit. 19. 1. 2024].

PYTHON DOCS (2024): re — Regular expression operations. Python 3.12.1 documentation. Dostupné z: https://docs.python.org/3/library/re.html [cit. 20. 1. 2024].

IBM (2024): What is natural language processing (NLP)?. Dostupné z: https://www.ibm.com/topics/natural-language-processing [cit. 20. 1. 2024].

PYTHON WIKI (2024): GUI Programming in Python. Dostupné z: https://wiki.python.org/moin/GuiProgramming [cit. 21. 1. 2024].