Лабораторна робота №5

Тема: Автокорекція тексту з використанням мінімальної відстані редагування

Варіант №6

```
import re
import numpy as np
import pandas as pd
from collections import Counter
import string
import nltk
```

ЗАВДАННЯ 1: Завантаження корпусу (Варіант 6) та Словник частотності

```
def load_and_process_corpus():
    """
    Завантажує корпус 'twitter_samples' (Варіант 6),
    об'єднує його та повертає список оброблених слів.
    """
    print("Завантаження NLTK корпусу 'twitter_samples' (Варіант 6)...")
    nltk.download('twitter_samples', quiet=True)
    from nltk.corpus import twitter_samples

    # Завантажуємо позитивні та негативні твіти
    all_tweets = twitter_samples.strings('positive_tweets.json')
    all_tweets.extend(twitter_samples.strings('negative_tweets.json'))

    # Об'єднуємо всі твіти в один гігантський рядок
    # і приводимо до нижнього регістру
    corpus_text = " ".join(all_tweets).lower()

    # re.findall(r'\w+', ...) знаходить всі "слова" (літери, цифри, _)
    # і автоматично ігнорує пунктуацію.
    words = re.findall(r'\w+', corpus_text)

    print(f"Корпус 'twitter_samples' оброблено.")
    return words

def get_count(word_l):
    """
    Створює словник частотності слів.
    (Використовуємо Counter для ефективності)
    """
    return Counter(word_l)
```

ЗАВДАННЯ 2: Обчислення ймовірностей слів

```
def get_probs(word_count_dict):
    """
    Обчислює ймовірності слів P(c) на основі словника частотності.
    """
    probs = {}
    total_count = sum(word_count_dict.values())

    for word, count in word_count_dict.items():
        probs[word] = count / total_count
    return probs
```

ЗАВДАННЯ 2: Функції для маніпуляції рядками

```
def delete_letter(word, verbose=False):
    """Видаляє одну літеру зі слова."""
    split_l = [(word[:i], word[i:]) for i in range(len(word))]
    delete_l = [L + R[1:] for L, R in split_l if R]
    if verbose: print(f"input word {word}, \nsplit_l = {split_l}, \ndelete_l = {delete_l}")
    return delete_l

def switch_letter(word, verbose=False):
    """Міняє місцями дві сусідні літери."""
    split_l = [(word[:i], word[i:]) for i in range(len(word))]
    switch_l = [L + R[1] + R[0] + R[2:] for L, R in split_l if len(R) > 1]
    if verbose: print(f"Input word = {word} \nsplit_l = {split_l} \nswitch_l = {switch_l}")
    return switch_l

def replace_letter(word, verbose=False):
    """Замінює одну літеру іншою."""
    letters = 'abcdefghijklmnopqrstuvwxyz'
    split_l = [(word[:i], word[i:]) for i in range(len(word))]
    replace_l = [L + c + R[1:] for L, R in split_l if R for c in letters if L + c + R[1:] != word]
    if verbose: print(f"Input word = {word} \nsplit_l = {split_l} \nreplace_l = {replace_l}")
    return replace_l

def insert_letter(word, verbose=False):
    """Вставляє одну літеру."""
    letters = 'abcdefghijklmnopqrstuvwxyz'
    split_l = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    insert_l = [L + c + R for L, R in split_l for c in letters]
    if verbose: print(f"Input word {word} \nsplit_l = {split_l} \ninsert_l = {insert_l}")
    return insert_l
```

ЗАВДАННЯ 4: Пошук слів на відстані 1 та 2

```
def edit_one_letter(word, allow_switches=True):
    """
    Повертає набір слів, які знаходяться на відстані одного редагування.
    """
    edit_set = set()
    edit_set.update(delete_letter(word))
    edit_set.update(replace_letter(word))
    edit_set.update(insert_letter(word))
    if allow_switches:
        edit_set.update(switch_letter(word))
    return edit_set

def edit_two_letters(word, allow_switches=True):
    """
    Повертає набір слів, які знаходяться на відстані двох редагувань.
    """
    edit_set = set()
    for w in edit_one_letter(word, allow_switches):
        edit_set.update(edit_one_letter(w, allow_switches))
    return edit_set
```

ЗАВДАННЯ 5: Алгоритм мінімальної відстані редагування (Min Edit Distance)

```
def min_edit_distance(source, target, ins_cost=1, del_cost=1, rep_cost=2):
    """
    Обчислює мінімальну відстань редагування між двома рядками.
    """
    m = len(source)
    n = len(target)
    D = np.zeros((m+1, n+1), dtype=int)

    # Ініціалізація
    for row in range(m+1):
        D[row, 0] = row * del_cost
    for col in range(n+1):
        D[0, col] = col * ins_cost

    # Заповнення матриці
    for row in range(1, m+1):
        for col in range(1, n+1):
            r_cost = rep_cost
            if source[row-1] == target[col-1]:
                r_cost = 0 # Вартість "заміни" на той самий символ = 0

            D[row, col] = min(
                D[row-1, col] + del_cost,      # видалення
                D[row, col-1] + ins_cost,      # вставка
                D[row-1, col-1] + r_cost       # заміна
            )
```

```
    return D, D[m, n]
```

ЗАВДАННЯ 6: Створення системи автоматичної корекції

```
def get_corrections(word, probs, vocab, n=2, verbose=False):
    """
    Повертає n найбільш ймовірних корекцій для введеного слова.
    (ПОВЕРНЕНО ДО ВЕРСІЇ, ЩО ДАВАЛА ~41.67%)
    """
    suggestions = []

    # 1. Якщо слово є у словнику, воно - кандидат
    if word in vocab:
        suggestions.append(word)

    # 2. Генеруємо кандидатів на відстані 1 (ЗАВЖДИ)
    # (& vocab = перетин множин, залишає тільки ті, що є у словнику)
    edit_one_set = edit_one_letter(word) & vocab
    suggestions.extend(list(edit_one_set))

    # 3. Якщо не знайдено ВЗАГАЛІ НІЧОГО (навіть слова в vocab), шукаємо на відстані 2
    if not suggestions:
        edit_two_set = edit_two_letters(word) & vocab
        if edit_two_set:
            suggestions = list(edit_two_set)
        # 4. Якщо нічого не знайдено, повертаємо оригінальне слово
        else:
            suggestions = [word]

    # 5. Обчислюємо ймовірності для унікальних пропозицій
    # (Використовуємо set() для видалення дублікатів, напр. 'folow' з vocab і 'folow' з edit_one_set)
    best_words = {}
    for w in set(suggestions): # <--- Важливо використати set()
        if w in vocab:
            best_words[w] = probs.get(w, 0)
        else:
            best_words[w] = 0 # Для слів, яких немає у vocab (напр., оригінальне слово з помилкою)

    # 6. Повертаємо n слів з найвищими ймовірностями
    n_best = sorted(best_words.items(), key=lambda x: x[1], reverse=True)[:n]

    if verbose:
        print(f"entered word = {word}, \nsuggestions = {list(set(suggestions))}")

    return n_best
```

ЗАВДАННЯ 7: Оцінка точності та ефективності

```
def evaluate_model(test_data, probs, vocab):
    """
    Оцінює точність системи автокорекції.
    test_data: список кортежів ('помилка', 'правильне_слово')
    """
    correct_predictions = 0
    total = len(test_data)

    if total == 0:
        return 0.0

    print(f"\n--- Початок оцінки на {total} тестових словах ---")

    for misspelled, correct_word in test_data:
        # Отримуємо 1 найкращу пропозицію
        prediction_list = get_corrections(misspelled, probs, vocab, n=1)

        if prediction_list:
            predicted_word = prediction_list[0][0]
            if predicted_word == correct_word:
                correct_predictions += 1
            # else:
            #     # (Для аналізу помилок)
            #     print(f"Помилка: введено '{misspelled}', очікувано '{correct_word}', отримано '{predicted_word}'")

    accuracy = correct_predictions / total
    return accuracy
```

ГОЛОВНИЙ БЛОК ВИКОНАННЯ

```
if __name__ == "__main__":

    # --- Завдання 1: Завантаження та обробка корпусу 'twitter_samples' ---
    word_l = load_and_process_corpus()

    if word_l:
        # --- Завдання 1 (продовження): Словник частотності ---
        word_count_dict = get_count(word_l)
        print(f"Корпус оброблено. Загальна кількість слів: {len(word_l)}")
        print(f"Кількість унікальних слів (розмір словника): {len(word_count_dict)}")

        # --- Завдання 3: Обчислення ймовірностей ---
        probs = get_probs(word_count_dict)
        # Створюємо set() для швидкого пошуку у словнику (O(1))
        vocab = set(word_count_dict.keys())
        print(f"Ймовірності слів обчислено. P('the') = {probs.get('the', 0):.6f}")

        # --- Демонстрація Завдання 5: Min Edit Distance ---
        print("\n--- Демонстрація (Завдання 5): Min Edit Distance ---")
```

```
test_pairs = [('play', 'stay'), ('sunday', 'saturday'), ('intention', 'execution')]
for source, target in test_pairs:
    D, med = min_edit_distance(source, target)
    print(f"Мінімальна відстань редагування між '{source}' та '{target}': {med}")
    # print("Матриця відстаней:")
    # print(pd.DataFrame(D, index=['#'] + list(source), columns=['#'] + list(target)))
    # print()

# --- Демонстрація Завдання 6: Система автокорекції ---
print("\n--- Демонстрація (Завдання 6): Автокорекція ---")

# Приклад 1: Типова помилка в соцмережах
test_word_1 = "folow" # Правильно: "follow"
corrections_1 = get_corrections(test_word_1, probs, vocab, n=2, verbose=True)
print(f"Top 2 corrections = {corrections_1}\n")

print("-" * 20)

# Приклад 2: Інша типова помилка
test_word_2 = "happpy" # Правильно: "happy"
corrections_2 = get_corrections(test_word_2, probs, vocab, n=2, verbose=True)
print(f"Top 2 corrections = {corrections_2}\n")

print("-" * 20)

# Приклад 3: Перестановка
test_word_3 = "todya" # Правильно: "today"
corrections_3 = get_corrections(test_word_3, probs, vocab, n=2, verbose=True)
print(f"Top 2 corrections = {corrections_3}\n")


# --- Завдання 7: Оцінка точності ---
# Створюємо тестовий набір (слова, які точно є у twitter_samples)
# Цей набір тепер повинен працювати краще з логікою з опису
test_set = [
    ('folow', 'follow'),     # Вставка
    ('tomorow', 'tomorrow'), # Заміна
    ('happpy', 'happy'),     # Видалення
    ('goood', 'good'),       # Видалення
    ('sooo', 'so'),          # Видалення (сленг)
    ('im', 'i'),             # (i'm -> i m), 'im' - помилка
    ('u', 'you'),            # Заміна (сленг)
    ('plz', 'please'),       # Відстань 2
    ('grt', 'great'),        # Вставка
    ('thx', 'thanks'),       # Відстань 2
    ('awsm', 'awesome'),     # Відстань 2
    ('luv', 'love')          # Заміна (сленг)
]

accuracy = evaluate_model(test_set, probs, vocab)
print("-------------------------------------------------")
```

```
            print(f"Точність моделі на тестовому наборі: {accuracy * 100:.2f}%")
            print("--------------------------------------------------")

        else:
            print("Не вдалося завантажити корпус. Роботу скрипта зупинено.")
```

```
Завантаження NLTK корпусу 'twitter_samples' (Варіант 6)...
Корпус 'twitter_samples' оброблено.
Корпус оброблено. Загальна кількість слів: 114836
Кількість унікальних слів (розмір словника): 20819
Ймовірності слів обчислено. P('the') = 0.017460

--- Демонстрація (Завдання 5): Min Edit Distance ---
Мінімальна відстань редагування між 'play' та 'stay': 4
Мінімальна відстань редагування між 'sunday' та 'saturday': 4
Мінімальна відстань редагування між 'intention' та 'execution': 8

--- Демонстрація (Завдання 6): Автокорекція ---
entered word = folow,
suggestions = ['follow', 'folow', 'flow']
Top 2 corrections = [('follow', 0.003918631787941064), ('flow', 2.6124211919607092e-05)]

--------------------
entered word = happpy,
suggestions = ['happpy', 'happy']
Top 2 corrections = [('happy', 0.0019331916820509248), ('happpy', 8.70807063986903e-06)]

--------------------
entered word = todya,
suggestions = ['today']
Top 2 corrections = [('today', 0.0019767320352502 7)]


--- Початок оцінки на 12 тестових словах ---
--------------------------------------------------
Точність моделі на тестовому наборі: 41.67%
--------------------------------------------------
```