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

# ================================================================================
# ЗАВДАННЯ 1: Підготовка даних (Варіант 6: Twitter Corpus)
# ================================================================================

def load_corpus():
    """
    Завантажує та об'єднує позитивні та негативні твіти з корпусу twitter_samples.
    """
    print("Завантаження корпусу twitter_samples...")
    nltk.download('twitter_samples', quiet=True)
    from nltk.corpus import twitter_samples

    # Отримуємо всі твіти (позитивні та негативні)
    all_tweets = twitter_samples.strings('positive_tweets.json') + twitter_samples.strings('negative_tweets.json')
    print(f"Завантажено {len(all_tweets)} твітів.")
    return all_tweets

def process_data(tweets):
    """
    Обробляє список твітів: розбиває на слова, переводить у нижній регістр.
    Використовує регулярний вираз для вилучення слів (ігноруючи пунктуацію та смайли).
    """
    text = " ".join(tweets).lower()
    # \w+ знаходить послідовності букв, цифр та підкреслення
    words = re.findall(r'\w+', text)
    return words

def get_count(word_l):
    """
    Створює словник частотності слів (Word Count).
    """
    return Counter(word_l)

def get_probs(word_count_dict):
    """
    ЗАВДАННЯ 3: Обчислює ймовірності слів P(w).
    """
    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: Операції редагування
# ================================================================================

def delete_letter(word):
    """Генерує варіанти з видаленням однієї літери."""
    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]
    return delete_l

def switch_letter(word):
    """Генерує варіанти з перестановкою сусідніх літер."""
    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]
    return switch_l

def replace_letter(word):
    """Генерує варіанти із заміною однієї літери на іншу (a-z)."""
    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 c != R[0]]
    return replace_l

def insert_letter(word):
    """Генерує варіанти зі вставкою однієї літери."""
    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]
    return insert_l

# ================================================================================
# ЗАВДАННЯ 4: Пошук кандидатів (Edit Distance 1 & 2)
```

```
# ЗАВДАННЯ 4: Пошук кандидатів (Edit Distance 1 & 2)
# ============================================================================

def edit_one_letter(word, allow_switches=True):
    """Повертає множину слів на відстані 1 редагування."""
    edit_set = set()
    edit_set.update(delete_letter(word))
    edit_set.update(insert_letter(word))
    edit_set.update(replace_letter(word))
    if allow_switches:
        edit_set.update(switch_letter(word))
    return edit_set

def edit_two_letters(word, allow_switches=True):
    """Повертає множину слів на відстані 2 редагувань."""
    edit_set = set()
    edit_one = edit_one_letter(word, allow_switches)
    for w in edit_one:
        if w:
            edit_set.update(edit_one_letter(w, allow_switches))
    return edit_set

# ============================================================================
# ЗАВДАННЯ 5: Мінімальна відстань редагування (Levenshtein)
# ============================================================================

def min_edit_distance(source, target, ins_cost=1, del_cost=1, rep_cost=2):
    """
    Обчислює мінімальну вартість перетворення source в target
    за допомогою динамічного програмування.
    """
    m = len(source)
    n = len(target)
    D = np.zeros((m+1, n+1), dtype=int)

    # Ініціалізація нульового рядка та стовпця
    for row in range(1, m + 1):
        D[row, 0] = D[row-1, 0] + del_cost
    for col in range(1, n + 1):
        D[0, col] = D[0, col-1] + 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

            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: Система автокорекції
# ============================================================================

def get_corrections(word, probs, vocab, n=2, verbose=False):
    """
    Повертає n найбільш ймовірних виправлень для слова.
    Пріоритет:
    1. Слово є в словнику.
    2. Відстань 1.
    3. Відстань 2.
    """
    suggestions = []
    n_best = []

    # 1. Перевіряємо, чи слово вже правильне
    if word in vocab:
        suggestions = [word]

    # 2. Відстань 1
    if not suggestions:
        suggestions = list(edit_one_letter(word).intersection(vocab))

    # 3. Відстань 2
    if not suggestions:
        suggestions = list(edit_two_letters(word).intersection(vocab))
```

```
        # 4. Якщо нічого не знайдено, повертаємо саме слово
        if not suggestions:
            suggestions = [word]

        # Сортуємо за ймовірністю
        best_words = {}
        for w in suggestions:
            best_words[w] = probs.get(w, 0)

        n_best = sorted(best_words.items(), key=lambda x: x[1], reverse=True)[:n]

        if verbose:
            print(f"Вхідне слово: {word}")
            print(f"Кандидати: {n_best}")

        return n_best

# =============================================================================
# ГОЛОВНИЙ БЛОК ТЕСТУВАННЯ
# =============================================================================

if __name__ == "__main__":
    # 1. Підготовка моделі
    tweets = load_corpus()
    word_list = process_data(tweets)
    vocab = set(word_list)
    word_counts = get_count(word_list)
    probs = get_probs(word_counts)

    print(f"Розмір словника: {len(vocab)} унікальних слів.")
    print(f"Всього слів у корпусі: {len(word_list)}")
    print("-" * 30)

    # 2. Тестування автокорекції (Завдання 7)
    print("ТЕСТУВАННЯ АВТОКОРЕКЦІЇ:")
    # Створюємо штучні помилки, типові для соцмереж
    test_cases = [
        "hapy",      # happy (видалення)
        "brthday",   # birthday (видалення)
        "follwo",    # follow (перестановка)
        "thnks",     # thanks (видалення голосних)
        "awsome",    # awesome (фонетичне написання)
        "messsage",  # message (вставка)
        "goood"      # good (подовження, типове для твітів)
    ]

    for test_word in test_cases:
        res = get_corrections(test_word, probs, vocab, n=3, verbose=False)
        print(f"{test_word:<10} -> {res[0][0] if res else 'Не знайдено'} \t(Всі варіанти: {[w for w, p in res]})")

    print("-" * 30)

    # 3. Тестування Min Edit Distance
    print("ТЕСТУВАННЯ MIN EDIT DISTANCE:")
    pairs = [("play", "stay"), ("intention", "execution")]
    for s, t in pairs:
        matrix, dist = min_edit_distance(s, t, rep_cost=2)
        print(f"Відстань між '{s}' та '{t}': {dist}")
```

```
Завантаження корпусу twitter_samples...
Завантажено 10000 твітів.
Розмір словника: 20819 унікальних слів.
Всього слів у корпусі: 114836
------------------------------
ТЕСТУВАННЯ АВТОКОРЕКЦІЇ:
hapy       -> happy     (Всі варіанти: ['happy', 'hay'])
brthday    -> birthday  (Всі варіанти: ['birthday'])
follwo     -> follow    (Всі варіанти: ['follow'])
thnks      -> thnks     (Всі варіанти: ['thnks'])
awsome     -> awesome   (Всі варіанти: ['awesome', 'awsme'])
messsage   -> message   (Всі варіанти: ['message'])
goood      -> goood     (Всі варіанти: ['goood'])
------------------------------
ТЕСТУВАННЯ MIN EDIT DISTANCE:
Відстань між 'play' та 'stay': 4
Відстань між 'intention' та 'execution': 8
```