УДК 378.147:004.432

Рєзіна Ольга Василівна
кандидат педагогічних наук, доцент, доцент кафедри інформатики та інформаційних технологій
Центральноукраїнський державний педагогічний університет імені Володимира Винниченка,
м. Кропивницький, Україна
ORCID ID 0000-0001-6077-9413
olga.riezina@gmail.com

МЕТОДИЧНІ АСПЕКТИ НАВЧАННЯ СТУДЕНТІВ СТВОРЕННЮ ЦИФРОВИХ ЧАСТОТНИХ СЛОВНИКІВ

Анотація. Частотні словники створюються для виявлення найбільш використовуваних слів у природній мові, мові письменника, певному творі тощо. Цими словниками послуговуються при вивченні іноземних мов; розробці інших видів словників; проектуванні мовних експериментів; проведенні досліджень у галузях лексичної семантики, психолінгвістики, морфології та інших; створенні прикладних програм, орієнтованих на опрацювання природної мови. Інформаційно-комунікаційні технології, доступ до величезних електронних лінгвістичних корпусів, національних баз даних слів, заснованих на субтитрах телевізійних програм та фільмів, значно прискорили дослідження в галузі статистичного опрацювання текстів. Зважаючи на те, що частотні словники широко використовуються в різних сферах діяльності, а їх побудова передбачає розв'язання широкого кола лінгвістичних задач, доцільно розглянути технологію створення таких словників у процесі підготовки фахівців з прикладної лінгвістики та комп'ютерних наук, учителів інформатики. У статті розглядається методика навчання студентів створенню частотних словників. Запропоновано алгоритм побудови частотного словника. Аналізуються особливості реалізації кожного кроку алгоритму з використанням мови Python, яка є популярною мовою програмування з відкритим кодом та великими бібліотеками. Автор наводить програмні коди та обґрунтовує використання відповідних модулів, методів опрацювання рядків, функцій, констант, структур даних, регулярних виразів. Запропонована методика спрямована на: 1) підвищення мотивації студентів до навчання та 2) розкриття практичної значущості засвоєних ними методів та прийомів програмування. Автор вважає, що підхід, представлений у цій статті, може бути корисним викладачам дисциплін інформатичного циклу.

Ключові слова: частотний словник; мова програмування Python; програмний код; методика навчання.

1. ВСТУП

Постановка проблеми. Частота, з якою слово зустрічається в заданому тексті або корпусі, є найважливішою величиною в дослідженнях з опрацювання текстів [1], [2]. Певні статистичні дані про кожну лексему деякого текстового корпусу наводяться в частотних словниках, які містять щонайменше два види записів:

- список лексем, впорядкованих за алфавітом, із зазначенням статистичних даних про кожний запис;
- список лексем, впорядкованих за певним статистичним значенням (як правило, за частотою).

Методологічно частотний словник відрізняється від усіх інших типів словників тим, що обов'язково базується на лінгвістичному корпусі [3].

Метою створення частотних словників є виявлення найбільш використовуваних слів у природній мові, мові письменника, певному творі тощо. При вивченні іноземної мови використання частотного словника дає змогу учневі та вчителю оптимізувати

процес навчання, зосереджуючись на запам'ятовуванні тих слів, якими вони послуговуватимуться у повсякденному спілкуванні.

Показники частотності надають, зокрема, ще й такі можливості використання:

– у процесі розробки інших видів словників;
– у процесі проектування мовних експериментів чи проведення досліджень у галузях лексичної семантики, психолінгвістики, морфології та інших;
– у процесі створення різноманітних прикладних програм, орієнтованих на опрацювання природної мови [4].

Обчислення частотності слів здійснювалися ще в Давньому Єгипті. Дослідники з Александрії розрізняли рідкісні та звичайні слова гомерівської грецької мови. У X столітті талмудисти класифікували та підраховували слова в Торі [5]. Сьогодні засоби інформаційно-комунікаційних технологій і доступ до величезних цифрових лінгвістичних корпусів, а також до національних баз даних слів, заснованих на субтитрах телевізійних програм та фільмів, значно прискорили дослідження в цій галузі.

З огляду на вищесказане можна зазначити, що побудова частотного словника передбачає розв'язання широкого кола лінгвістичних задач. Використання інформаційно-комунікаційних технологій робить цей процес швидким, зручним та ефективним. Зважаючи на те, що частотні словники широко використовуються в різних сферах діяльності, й у вивченні іноземної мови зокрема, доцільно розглянути технологію їх створення в процесі підготовки фахівців з прикладної лінгвістики, комп'ютерних наук та вчителів інформатики.

Виконання студентами завдання на створення частотного словника сприяє формуванню в них ряду важливих спеціальних компетентностей:

– здатності до використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення та аналізу алгоритмів, оцінювання їх ефективності та складності;
– здатності до розроблення та реалізації комп'ютерних програм з використанням відповідних моделей, алгоритмів обчислень, методів програмування та структур даних;
– здатності до опрацювання й аналізу результатів виконання алгоритму, комп'ютерної програми.

Формування зазначених компетентностей передбачено освітньо-професійними програмами, за якими відбувається підготовка студентів за спеціальностями 035.10 Філологія (Прикладна лінгвістика), 122 Комп'ютерні науки, 014.09 Середня освіта (Інформатика).

Аналіз останніх досліджень і публікацій. Проблемам створення і використання частотних словників присвячено ряд робіт. У дослідженні [6] описано процес створення комплекту словників, що налаштовується для конкретного користувача та застосовується для допомоги при введенні тексту до електронного пристрою. Такі користувацькі словники базуються на частотних списках слів, які збираються із текстів користувача, представлених у різних джерелах – електронна пошта, чат, повідомлення, коментар, оновлення статусу, твіт, пошуковий запит. Особливості побудови ієрархічного словника на основі методів зберігання слів розглянуто в дослідженні [7]. Важливо, що метод зберігання слова обирається в залежності від його частотності в одному чи більше текстах. Н. П. Дарчук розглядає лінгвістичні засади семантичного розмічування Корпусу української мови на основі частотного словника публіцистичного стилю обсягом 40 тисяч лексем [8]. Н. С. Угольнікова та М. В. Чухненко аналізують можливості використання частотних словників у процесі класифікації творів масової літератури [9]. У роботі [10] розглянуто процес написання

комп'ютерної програми підрахунку частотності кожного слова для деякого заданого рядка. Як засіб використано мову програмування Python. Статтю [11] присвячено методичним особливостям використання електронних словників у процесі формування лексикографічної компетентності майбутніх учителів початкової школи.

Метою статті є висвітлення окремих компонентів методичної системи навчання студентів створенню частотних словників, а саме: визначено цілі, окреслено зміст, обґрунтовано добір засобів та методів навчання.

2. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Розглянемо методичні особливості створення частотного словника. Передусім необхідно зазначити, що створення частотних словників – це процес, який складається з таких кроків:

1. Нормалізація тексту (видалення розділових знаків та переведення символів у нижній регістр) для подальшого аналізу текстових даних.
2. Створення списку лексем, до якого входять слова тексту, абревіатури, загальноприйняті скорочення тощо.
3. Видалення з отриманого списку стоп-слів (функціональних форм дієслів, артиклів, сполучників, функціональних морфем тощо).
4. Створення словника, ключами якого є значущі слова заданого тексту, а значеннями – числа, що визначають, скільки разів дане слово зустрічається в тексті.
5. Сортування сформованого словника за ключами (алфавітом) і значеннями (частотністю елементів).

Нормалізація тексту означає перетворення його в стандартну форму, зручну для подальшого опрацювання [12]. Ключовим етапом процесу нормалізації є виокремлення слів із заданого тексту, або токенізація. У більшості мов слова розділяються пробілами, але часто наявність пробілів не є достатньою для виділення слова. Узагалі тексти можуть містити нестандартні послідовності слів, у тому числі назви одиниць вимірювання, назви грошових одиниць, значення дат і часу в різних форматах, телефонні номери, дроби, адреси електронної пошти, веб-адреси, хеш-теги, емотикони, субстандартну лексику та багато іншого. Ідентифікація всіх зазначених елементів і створення для них окремих списків у частотному словнику є складною задачею. На початковому етапі навчання створенню частотних словників доцільно розглянути базовий варіант токенізації – видалення із заданого тексту знаків пунктуації та чисел.

Задача видалення із тексту знаків пунктуації (clean up text) є класичною задачею опрацювання текстів. Існує багато способів її розв'язання. Розглянемо деякі з них. Як засіб виберемо мову програмування Python, яка надає користувачам потужні засоби опрацювання текстових даних.

Перший спосіб. Формування нового рядка, що не містить пунктуаційних знаків.

```
import string
#ініціалізація рядка
s = '''A c(omputer can do "some tasks" better than a <p>erson can. Any
tasks!
a computer can do better? than a </p>erson is, by \\definition\\, the
tasks
requiring no human. '12', "3;", &, @, 4creativity or ingenuity.)
'''
#видалення пунктуаційних знаків
s_without_punct = ""
```

```
for char in s:
   if char not in string.punctuation:
       s_without_punct+=char
#виведення результату
print(s_without_punct
```

У програмі підключено модуль string і використано його константу punctuation, яка є рядком символів ASCII, що вважаються знаками пунктуації – """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~""" [13]. Результат сформовано у змінній s_without_punct, якій спочатку надано значення порожнього рядка. Далі до змінної s_without_punct додані тільки ті символи рядка s, що не є знаками пунктуації.

Час виконання програми – 0.004692289217788375 сек.

Другий спосіб. Заміна пунктуаційних знаків порожнім рядком.

```
import string
#ініціалізація рядка
s = '''A c(omputer can do "some tasks" better than a <p>erson can. Any
tasks!
a computer can do better? than a </p>erson is, by \\definition\\, the
tasks
requiring no human. '12', "3;", &, @, 4creativity or ingenuity.)
'''
#видалення пунктуаційних знаків
for char in string.punctuation:
    s = s.replace(char, "")
#виведення результату
print(s)
```

Використано метод replace(): у наведеному прикладі він повертає копію рядка s, у якій всі символи константи punctuation замінено на порожній рядок.

Час виконання програми – 0.005752258796413232 сек.

Третій спосіб. Використання методів translate() та maketrans().

```
import string
#ініціалізація рядка
s = '''A c(omputer can do "some tasks" better than a <p>erson can. Any
tasks!
a computer can do better? than a </p>erson is, by \\definition\\, the
tasks
requiring no human. '12', "3;", &, @, 4creativity or ingenuity.)
'''
#створення таблиці перекладу
translator = s.maketrans("", "", string.punctuation)
#заміна символів відповідно до таблиці перекладу
s=s.translate(translator)
#виведення результату
print(s)
```

При застосуванні методів translate() та maketrans(), на відміну від replace(), можна виконати множинну заміну. За допомогою методу maketrans() створюється таблиця заміни одного символу іншим (словник перекладу). Далі ця таблиця передається методу translate(), щоб здійснити переклад. У наведеному прикладі два перших аргументи методу maketrans() – порожній рядок і порожній рядок указують на те, що заміна символів не відбувається, а третій аргумент – string.punctuation указує на те, які символи видаляються. Далі метод translate() повертає копію рядка s, в якій кожний

символ замінено та видалено відповідно до заданої методом maketrans() таблиці перекладу.

Час виконання програми – 0.005547153441767145 сек.

Четвертий спосіб. Використання регулярних виразів.

```
import string, re
#ініціалізація рядка
s = '''A c(omputer can do "some tasks" better than a <p>erson can. Any
tasks!
a computer can do better? than a </p>erson is, by \\definition\\, the
tasks
requiring no human. '12', "3;", &, @, 4creativity or ingenuity.)
'''
#видалення пунктуаційних знаків
s = re.sub(r"[^a-zA-Z0-9\s]", "", s)
#виведення результату
print(s)
```

Крім модуля string імпортовано модуль re, що надає можливість використання регулярних виразів. Існує кілька способів побудови шаблону регулярного виразу для розв'язання поставленої задачі. Розглянемо наведений. Запис a-zA-Z0-9 визначає алфавітно-цифрові символи, \s – пробільні символи. Квадратні дужки та карет ^ визначають недопустимі символи. Отже шаблон r"[^a-zA-Z0-9\s]" трактується як послідовність символів, що не є буквою, цифрою, пробільним символом. Такий шаблон є простим для розуміння та зручним для подальшої видозміни з метою видалення цифр із тексту. Використання методу sub() дає змогу замінити в тексті всі підрядки, що збігаються з шаблоном, на порожні рядки, тобто видалити їх.

Час виконання програми – 0.006757306063733463 сек.

Для запропонованого значення змінної s результатом роботи наведених програм є рядок:

```
A computer can do some tasks better than a person can Any tasks
a computer can do better than a person is by definition the tasks
requiring no human 12 3   4creativity or ingenuity
```

Час виконання розглянутих програм є приблизно однаковим, тобто за своєю ефективністю програми суттєво не відрізняються. Тому варто запропонувати студентам обрати той спосіб розв'язання, що є для них особисто зрозумілішим.

Для подальшого розгляду виберемо четвертий спосіб, що передбачає використання регулярних виразів. Для видалення з рядка цифр змінимо шаблон r"[^a-zA-Z0-9\s] " на r"[^a-zA-Z\s], що дає змогу позбавитися цифр і залишити тільки послідовності букв та пробільних символів.

Нормалізація тексту також передбачає переведення символів у нижній регістр, що можна здійснити за допомогою оператора

```
s_norm=s.casefold().
```

Результат:

```
a computer can do some tasks better than a person can any tasks
a computer can do better than a person is by definition the tasks
requiring no human     creativity or ingenuity
```

Другий крок алгоритму створення частотного словника виконується з використанням методу split(), який повертає список слів, що містяться у заданому рядку:

```
full_words = s_norm.split()
print(full_words)
```

Результат:

```
['a', 'computer', 'can', 'do', 'some', 'tasks', 'better', 'than', 'a',
'person', 'can', 'any', 'tasks', 'a', 'computer', 'can', 'do',
'better', 'than', 'a', 'person', 'is', 'by', 'definition', 'the',
'tasks', 'requiring', 'no', 'human', 'creativity', 'or', 'ingenuity']
```

Список – найбільш гнучка та універсальна структура даних, доступна у Python, для якої передбачений потужний набір методів опрацювання [14].

Третій крок алгоритму виконується з використанням списку стоп-слів [15], що є доступним ресурсом NLTK (Natural Language Toolkit) – провідної платформи для побудови Python-програм, призначених для опрацювання природних людських мов (http://www.nltk.org/). Програму, що містить список стоп-слів, можна оформити окремим модулем з іменем nltk_stopwords та викликати цей модуль в програмі побудови частотного словника [16]. Фрагмент програми, призначений для видалення стоп-слів, має вигляд:

```
stop_words_list=nltk_stopwords.stopwords()
words=[word for word in full_words if word not in stop_words_list]
```

У наведеному фрагменті:

– функція stopwords() модуля nltk_stopwords надає доступ до списку стоп-слів;
– змінна stop_words_list містить список стоп-слів, перелічених у модулі nltk_stopwords;
– змінна full_words містить список, елементами якого є всі слова тексту;
– у змінній words формується список, елементами якого є значущі слова тексту.

Результат:

```
['computer', 'tasks', 'better', 'person', 'tasks', 'computer',
'better', 'person', 'definition', 'tasks', 'requiring', 'human',
'creativity', 'ingenuity']
```

Четвертий крок алгоритму передбачає створення словника, ключами якого є значущі слова заданого тексту, а значеннями – числа, що визначають, скільки разів дане слово зустрічається в тексті. Доцільним є використання вбудованої у Python структури даних «словник» [14]. Програмний код може мати такий вигляд:

```
dict = {}
for word in words:
    if word in dict:
        dict[word] += 1
    else:
        dict[word] = 1
print(dict)
```

Словник формується у змінній dict, якій спочатку надається значення порожнього словника {}. Далі аналізуються всі слова списку words і, якщо слова немає в словнику,

то воно додається із значенням 1, у протилежному випадку частотність збільшується на 1. Результат:

```
{'definition': 1, 'better': 2, 'creativity': 1, 'tasks': 3, 'human':
1, 'person': 2, 'requiring': 1, 'ingenuity': 1, 'computer': 2}
```

Для візуалізації процесу створення словника та кращого розуміння наведеного фрагменту програмного коду можна запропонувати студентам наступну схему (рис. 1):

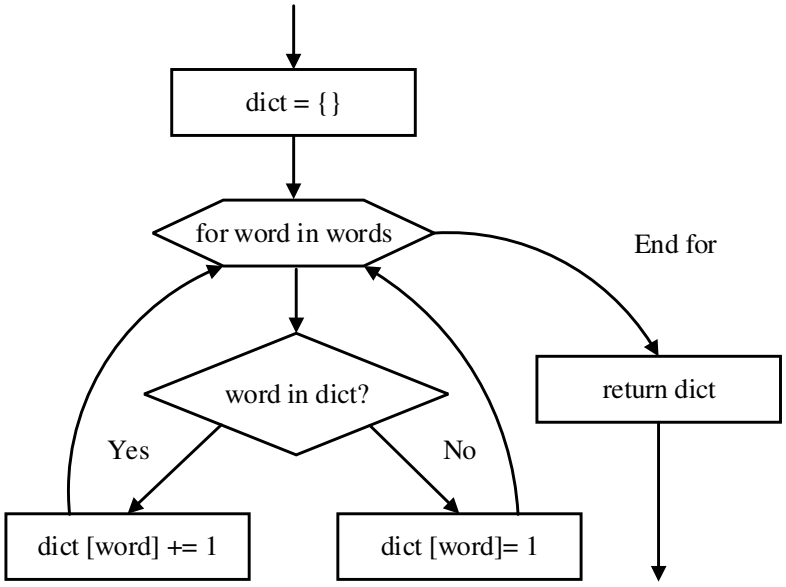

Рис. 1. Схема створення частотного словника

На п’ятому кроці алгоритму сформований словник необхідно відсортувати за ключами (алфавітом) і значеннями (частотністю елементів). У даному випадку доцільно скористатися функціями sorted() та lambda. Функція sorted() сортує елементи заданої ітерабельної структури (рядка, кортежу, списку, множини, словника) у визначеному порядку – за зростанням чи спаданням. Lambda-оператор, або lambda-функція, використовується для створення невеликих безіменних функцій у Python.

Сортування за алфавітом можна організувати так:

```
dict_sort_alph = sorted(dict.items(), key=lambda element:element[0])
print("Sorted dictionary by keys:")
for key,value in dict_sort_alph:
    print('{:15} : {:4}'.format(key,value))
```

При поясненні студентам цього фрагмента програмного коду необхідно:

- звернути увагу на використання функції sorted: усі елементи словника dict будуть відсортовані за зростанням;
- показати, що dict.items() викликається для того, щоб елементи словника перетворити на структуру даних «кортеж» [14]: [('better', 2), ('computer', 2), ('creativity', 1), ('definition', 1), ('human', 1), ('ingenuity', 1), ('person', 2), ('requiring', 1), ('tasks', 3)];
- констатувати, що lambda-функція використовується як ключ для сортування, а lambda element є окремим елементом у dict.items();

– визначити, що lambda-функція виконує сортування за element[0] – елементами кортежів з індексом 0, наприклад, для кортежа ('computer', 2) це є слово 'computer';
– повідомити, що результат функції sorted() – список кортежів, що зберігається у змінній dict_sort_alph;
– підсумувати, що використання циклу for дає змогу вивести на екран пари ключ:значення на екран.

Результат:

```
Sorted dictionary by keys:
better          :    2
computer        :    2
creativity      :    1
definition      :    1
human           :    1
ingenuity       :    1
person          :    2
requiring       :    1
tasks           :    3
```

Організувати сортування словника за частотністю можна аналогічно, змінивши у lambda-функції вираз element[0] на element[1], що буде вказувати на елементи кортежів з індексом 1, тобто частотність. Також треба змінити порядок сортування на спадний (reverse=True), щоб найбільш використовувані слова були розташовані на початку списку. Відповідний фрагмент програми набуде вигляду:

```
dict_sort_freq = sorted(dict.items(),key=lambda
element:element[1],reverse=True)
print("Sorted dictionary by values:")
for key,value in dict_sort_freq:
    print('{:15} : {:4}'.format(key,value))
```

Результат:

```
Sorted dictionary by values:
tasks           :    3
person          :    2
better          :    2
computer        :    2
requiring       :    1
creativity      :    1
ingenuity       :    1
human           :    1
definition      :    1
```

У цілому Python-програма створення частотного словника має вигляд:

```
import string, re, nltk_stopwords
#ініціалізація рядка
s = '''A c(omputer can do "some tasks" better than a <p>erson can. Any
tasks!
a computer can do better? than a </p>erson is, by \\definition\\, the
tasks
requiring no human. '12', "3;", &, @, 4creativity or ingenuity.)
'''
#нормалізація тексту
```

```
s = re.sub(r"[^a-zA-Z\s]", "", s)
print(s_norm)
#створення списку лексем
full_words = s_norm.split()
print(full_words)
#видалення стоп-слів
stop_words_list=nltk_stopwords.stopwords()
words=[word for word in full_words if word not in stop_words_list]
print(words)
#створення словника
dict = {}
for word in words:
    if word in dict:
        dict[word] += 1
    else:
        dict[word] = 1
print(dict)
#сортування словника за алфавітом та виведення результату
dict_sort_alph = sorted(dict.items(), key=lambda element:element[0])
print("Sorted dictionary by keys:")
for key,value in dict_sort_alph:
    print('{:15} : {:4}'.format(key,value))
#сортування словника за частотністю та виведення результату
dict_sort_freq = sorted(dict.items(), key=lambda element:element[1],
reverse=True)
print("Sorted dictionary by values:")
for key,value in dict_sort_freq:
    print('{:15} : {:4}'.format(key,value))
```

Наступним важливим кроком вивчення теми є використання в програмі текстових файлів: для читання – файли, що містять різножанрові тексти, для запису – побудовані частотні словники, які, у такому випадку, можна зручно опрацьовувати та зберігати тривалий час.

3. ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Важливість навчання студентів створенню частотних словників зумовлена широким їх використанням у практиках вивчення іноземних мов. Досвід показує, що вивчення цієї теми підвищує мотивацію студентів до навчання, сприяє розкриттю практичної значущості засвоєних ними методів та прийомів програмування, позитивно впливає на процес формування інформатичних компетентностей.

Напрями подальших досліджень доцільно зосередити на методичних аспектах створення частотних словників, що базуються на текстах українською мовою. Для цього необхідно врахувати:

- синтетичність української мови, в якій форми слів утворюються за допомогою префіксів, суфіксів та закінчень, що ускладнює машинну обробку тексту;
- особливості побудови шаблонів регулярних виразів для українських текстів;
- можливість створення списку стоп-слів української мови.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] M. Brysbaert, and B. New, " Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American

English," Behavior Research Methods, vol. 41, no. 4, pp. 977-990, Nov. 2009. doi: https://doi.org/10.3758/BRM.41.4.977.
[2] M. Davies, and D. Gardner, *A Frequency Dictionary of American English: Word Sketches, Collocates and Thematic Lists*. London, UK: Routledge, 2010.
[3] C. Lehmann, "Frequency dictionary", Christianlehmann.eu, 2018. [Електронний ресурс]. Доступно: https://www.christianlehmann.eu/ling/ling_meth/ling_description/lexicography/frequency_dict.html. Дата звернення: Липень 17, 2018.
[4] Word frequency data, Wordfrequency.info, 2018. [Електронний ресурс]. Доступно: https://www.wordfrequency.info/uses.asp. Дата звернення: Липень 17, 2018.
[5] J. DeRocher, M. Miron, S. Patten and C. Pratt, *The Counting of Words: A Review of the History, Techniques and Theory of Word Counts with Annotated Bibliography.* New York, NY, USA: Syracuse University Research Corporation, 1973.
[6] E. Tseng, S. A. Gandhi, A. D. Kramer, and L. S. Clair, "Blending customized user dictionaries based on frequency of usage - Google Patents", Patents.google.com, 2018. [Електронний ресурс]. Доступно: https://patents.google.com/patent/US9977774B2/en. Дата звернення: Липень 17, 2018.
[7] R. Meier, J. Hausmann, H. Urbschat, and T. Wanschura, "Hierarchical Dictionary with Statistical Filtering Based on Word Frequency - Google Patents", Patents.google.com, 2018. [Електронний ресурс]. Доступно: https://patents.google.com/patent/US20170220679A1/en. Дата звернення: Липень 17, 2018.
[8] Н. Дарчук, "Можливості семантичної розмітки корпусу української мови (КУМ)", *Науковий часопис НПУ імені М. П. Драгоманова. Серія 9: Сучасні тенденції розвитку мов*, випуск №15, с. 18-28, 2017.
[9] Н. Угольнікова, та М. Чухненко, "Особливості класифікації творів масової літератури", *Вісник НТУ «ХПІ». Актуальні проблеми розвитку українського суспільства*, № 29 (1251), с. 91-97, 2017.
[10] W. Turkel, and A. Crymble, "Counting Word Frequencies with Python", Programming historian.org, 2018. [Електронний ресурс]. Доступно: https://programminghistorian.org/en/lessons/counting-frequencies. Дата звернення: Липень 17, 2018.
[11] С. Помирча, та І. Пучков, "Електронні словники з української мови як засіб формування лексикографічної компетентності майбутніх учителів початкової школи", *Інформаційні технології і засоби навчання*, т. 59, №3, с. 104-113, 2017.
[12] D. Jurafsky, and J. Martin, *Speech and language processing, 2nd ed*. Upper Saddle River, NJ, USA: Prentice Hall, Inc., 2009.
[13] Common string operations – Python 3.7.0 documentation, Docs.python.org, 2018. [Електронний ресурс]. Доступно:https://docs.python.org/3/library/string.html. Дата звернення: Липень 18, 2018.
[14] Data Structures – Python 3.7.0 documentation, Docs.python.org, 2018. [Електронний ресурс]. Доступно: https://docs.python.org/3/tutorial/datastructures.html. Дата звернення: Липень 18, 2018.
[15] S. Bird, E. Klein, and E. Loper,"Accessing Text Corpora and Lexical Resources", in *Natural language processing with Python – Analyzing Text with the Natural Language Toolkit*, Nltk.org, 2018. [Електронний ресурс]. Доступно:https://www.nltk.org/book/ch02.html. Дата звернення: Липень 18, 2018.
[16] О. Рєзіна, "Технології статистичного опрацювання текстів та методика їх навчання", *Науковий часопис НПУ імені М. П. Драгоманова, Серія 2: Комп'ютерно-орієнтовані системи навчання*, випуск №19(26), с. 98-103, 2017.

Матеріал надійшов до редакції 26.07.2018 р.

МЕТОДИЧЕСКИЕ АСПЕКТЫ ОБУЧЕНИЯ СТУДЕНТОВ ПО СОЗДАНИЮ ЦИФРОВЫХ ЧАСТОТНЫХ СЛОВАРЕЙ

Резина Ольга Васильевна
кандидат педагогических наук, доцент, доцент кафедры информатики и информационных технологий
Центральноукраинский государственный педагогический университет имени Владимира Винниченко, г. Кропивницкий, Украина
ORCID ID 0000-0001-6077-9413
olga.riezina@gmail.com

Аннотация. Частотные словари создаются для выявления наиболее используемых слов в естественном языке, языке писателя, определенном произведении и т.д. Эти словари

применяют при изучении иностранных языков; разработке других видов словарей; проектировании языковых экспериментов; проведении исследований в областях лексической семантики, психолингвистики, морфологии и других; создании прикладных программ, ориентированных на обработку естественного языка. Информационно-коммуникационные технологии, доступ к огромным цифровым лингвистическим корпусам, национальным базам данных слов, основанных на субтитрах телевизионных программ и фильмов, значительно ускорили исследования в области статистической обработки текстов. Учитывая то, что частотные словари широко используются в различных сферах деятельности, а их построение предполагает решение широкого круга лингвистических задач, целесообразно рассмотреть технологию создания таких словарей в процессе подготовки специалистов по прикладной лингвистике и компьютерным наукам, учителей информатики. В статье рассматривается методика обучения студентов созданию частотных словарей. Предложен алгоритм построения частотного словаря. Анализируются особенности реализации каждого шага алгоритма с использованием языка Python, который является популярным языком программирования с открытым исходным кодом и обширными библиотеками. Автор представляет программные коды и обосновывает использование соответствующих модулей, методов обработки строк, функций, констант, структур данных, регулярных выражений. Предложенная методика направлена на: 1) повышение мотивации студентов к обучению и 2) раскрытие практической значимости усвоенных ими методов и приемов программирования. Автор считает, что подход, представленный в статье, может быть полезен преподавателям дисциплин информатического цикла.

Ключевые слова: частотный словарь; язык программирования Python; программный код; методика обучения.

METHODOLOGICAL ASPECTS OF TEACHING STUDENTS TO CREATE DIGITAL FREQUENCY DICTIONARIES

Olga V. Riezina
PhD of Pedagogical Sciences, Associate Professor at the Department of Computer Science and Technology
Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytskyi, Ukraine
ORCID ID 0000-0001-6077-9413
olga.riezina@gmail.com

Abstract. Frequency dictionaries are created to identify the most frequently used words in a natural language, a writer's language, a particular literary work, etc. These dictionaries are used while learning foreign languages, creating other kinds of dictionaries, conducting language experiments, carrying out research in the fields of lexical semantics, psycholinguistics, morphology, etc. and designing applications focused on natural language processing. Information and communication technology and access to large digital linguistic corpora and national word corpora based on the subtitles of movies and TV shows have accelerated research in the field of statistical text processing. Due to frequency dictionaries being widely used in various fields of activity and their creation offering the solution to a wide range of linguistic issues, it seems reasonable to analyze the technology of creating such dictionaries in the process of training future specialists in Applied Linguistics and Computer Science as well as teachers of Computer Science. The paper runs about the methods of teaching students to create frequency dictionaries. An algorithm of creating a frequency dictionary is offered. The peculiarities of realizing each stage of the algorithm with the help of the popular Python programming language, which has an open code and extensive libraries, are analyzed. The author provides program codes and justifies the use of corresponding modules, string methods, functions, constants, data structures, and regular expressions. The proposed methodology is aimed at 1) raising the students' motivation to study and 2) revealing the practical significance of the acquired programming methods and techniques. The author argues that the approach presented in this paper can be of advantage to teachers of computer-related courses.

Keywords: frequency dictionary; Python programming language; program code; teaching methods.

REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] M. Brysbaert, and B. New, " Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English," Behavior Research Methods, vol. 41, no. 4, pp. 977-990, Nov. 2009. doi: https://doi.org/10.3758/BRM.41.4.977. (in English)

[2] M. Davies, and D. Gardner, *A Frequency Dictionary of American English: Word Sketches, Collocates and Thematic Lists*. London, UK: Routledge, 2010. (in English)

[3] C. Lehmann, "Frequency dictionary", Christianlehmann.eu, 2018. [Online]. Available: https://www.christianlehmann.eu/ling/ling_meth/ling_description/lexicography/frequency_dict.html. Accessed on: Jul. 17, 2018. (in English)

[4] Word frequency data, Wordfrequency.info, 2018. [Online]. Available: https://www.wordfrequency.info/uses.asp. Accessed on: Jul. 17, 2018. (in English)

[5] J. DeRocher, M. Miron, S. Patten and C. Pratt, *The Counting of Words: A Review of the History, Techniques and Theory of Word Counts with Annotated Bibliography.* New York, NY, USA: Syracuse University Research Corporation, 1973. (in English)

[6] E. Tseng, S. A. Gandhi, A. D. Kramer, and L. S. Clair, "Blending customized user dictionaries based on frequency of usage - Google Patents", Patents.google.com, 2018. [Online]. Available: https://patents.google.com/patent/US9977774B2/en. Accessed on: Jul. 17, 2018. (in English)

[7] R. Meier, J. Hausmann, H. Urbschat, and T. Wanschura, "Hierarchical Dictionary with Statistical Filtering Based on Word Frequency - Google Patents", Patents.google.com, 2018. [Online]. Available: https://patents.google.com/patent/US20170220679A1/en. Accessed on: Jul. 17, 2018. (in English)

[8] N. Darchuk, " Capabilities of Semantic Tagging Within the Ukrainian Corpus", *Scientific journal of National Drahomanov Pedagogical University. Series 9: Modern Trends in the Development of Languages*, No. 15, pp. 18-28, 2017. (in Ukrainian)

[9] N. Uholnikova, and M. Chukhnenko, "Features of the classification of works of mass literature", *Bulletin of NTU "KhPI". Series: Actual problems of Ukrainian society development*, No. 29 (1251), pp. 91-97, 2017. (in Ukrainian)

[10] W. Turkel, and A. Crymble, "Counting Word Frequencies with Python", Programminghistorian.org, 2018. [Online]. Available: https://programminghistorian.org/en/lessons/counting-frequencies. Accessed on: Jul. 17, 2018. (in English)

[11] S. Pomyrcha, and I. Puchkov, "Electronic dictionaries in ukrainian as a mean of forming lexicographical competence of future primary school teachers", *Information Technologies and Learning Tools*, vol. 59, no. 3, pp. 104-113, 2017. (in Ukrainian)

[12] D. Jurafsky, and J. Martin, *Speech and language processing, 2nd ed.* Upper Saddle River, NJ, USA: Prentice Hall, Inc., 2009. (in English)

[13] Common string operations – Python 3.7.0 documentation, Docs.python.org, 2018. [Online]. Available: https://docs.python.org/3/library/string.html. Accessed on: Jul. 18, 2018. (in English)

[14] Data Structures – Python 3.7.0 documentation, Docs.python.org, 2018. [Online]. Available: https://docs.python.org/3/tutorial/datastructures.html. Accessed on: Jul. 18, 2018. (in English)

[15] S. Bird, E. Klein, and E. Loper,"Accessing Text Corpora and Lexical Resources", in *Natural language processing with Python – Analyzing Text with the Natural Language Toolkit*, Nltk.org, 2018. [Online]. Available: https://www.nltk.org/book/ch02.html. Accessed on: Jul. 18, 2018. (in English)

[16] O. Riezina, "Statistical text processing techniques and their teaching methods", *Scientific journal of National Drahomanov Pedagogical University. Series 2: Computer-oriented learning systems*, No. 19(26), pp. 98-103, 2017. (in Ukrainian)