МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Белгородский государственный технологический университет
им. В. Г. Шухова

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Методические указания к выполнению лабораторных работ
и расчетно-графического задания
для студентов 3-го курса направления подготовки
09.03.02 - Информационные системы и технологии

Белгород
2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Белгородский государственный технологический университет
им. В. Г. Шухова
Кафедра информационных технологий

Утверждено
научно-методическим советом
университета

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Методические указания к выполнению лабораторных работ
и расчетно-графического задания
для студентов 3-го курса направления подготовки
09.03.02 - Информационные системы и технологии

Белгород
2018

УДК 004(07)
ББК 32.973я7
Т38

Составитель:
канд. техн. наук, доц. И. А. Кочеткова

Рецензент канд. техн. наук, доц. Д. А. Юдин

Т38

Технологии обработки информации: Методические указания к выполнению лабораторных работ и расчетно-графического задания для студентов 3-го курса направления подготовки 09.03.02 - Информационные системы и технологии / сост. И. А. Кочеткова. – Белгород: Изд-во БГТУ, 2018. – 29с.

Методические указания подготовлены в соответствии с рабочей программой дисциплины «Технологии обработки информации» и требованиями ФГОС направления подготовки. Предназначены для приобретения студентами базовых знаний и практических навыков в области контент-анализа текста в информационно-поисковых системах, содержат теоретический материал и задания к выполнению пяти лабораторных работ и одного расчетно-графического задания.

Методические указания предназначены для студентов 3-го курса направления подготовки 09.03.02 – Информационные технологии**.**

Данное издание публикуется в авторской редакции.

УДК 004(07)
ББК 32.973я7

Содержание

Введение

Учебная дисциплина «Технология обработки информации» изучается студентами по направлению 09.03.02 – Информационные технологии в пятом семестре.

Цель курса - формирование знаний технологий обработки информации; навыков решения вычислительных, исследовательских и прикладных задач с использованием прикладного программного обеспечения; представлений о современных технологиях сбора, обработки и хранения информации.

Реализация алгоритмов обработки информации на различных языках программирования может быть осуществлена при помощи Microsoft Visual Studio. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств..

Лабораторная работа №1
Реализация алгоритмов поиска по тексту

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы поиска по тексту: прямой поиск; алгоритм Кнута, Морриса и Пратта; алгоритм Бойера-Мура.

Краткие теоретические сведения

Прямой поиск

Основная идея алгоритма прямым поиском заключается в посимвольном сравнении строки с подстрокой. В начальный момент происходит сравнение первого символа строки с первым символом подстроки, второго символа строки со вторым символом подстроки и т. д. Если произошло совпадение всех символов, то фиксируется факт нахождения подстроки. В противном случае производится сдвиг подстроки на одну позицию вправо и повторяется посимвольное сравнение (табл. 1). Символы, которые сравниваются, на рисунке выделены жирным. Рассматриваемые сдвиги подстроки повторяются до тех пор, пока конец подстроки не достиг конца строки или не произошло полное совпадение символов подстроки со строкой, то есть найдется подстрока.

Таблица.1

Демонстрация алгоритма прямого поиска

Строка	A	B	C	A	B	C	A	A	B	C	A	B	D
Подстрока	A	B A	C B A	A C B A	B A C B A	D B A C B A	D B A C B A	D B A C B A	D B A C B	D B A C	D B A	D B	D

Данный алгоритм является малозатратным и не нуждается в предварительной обработке и в дополнительном пространстве. Большинство сравнений алгоритма прямого поиска являются лишними.

Поэтому в худшем случае алгоритм будет малоэффективен, так как его сложность будет пропорциональна O((n-m+1)*m), где n и m – длины строки и подстроки соответственно.

Алгоритм Кнута, Морриса и Пратта

Основным отличием алгоритма Кнута, Морриса и Пратта от алгоритма прямого поиска заключается в том, что сдвиг подстроки выполняется не на один символ на каждом шаге алгоритма, а на некоторое переменное количество символов. Следовательно, перед тем как осуществлять очередной сдвиг, необходимо определить величину сдвига. Для повышения эффективности алгоритма необходимо, чтобы сдвиг на каждом шаге был бы как можно большим (табл. 2). На рисунке символы, подвергшиеся сравнению, выделены жирным шрифтом.

Если для произвольной подстроки определить все ее начала, одновременно являющиеся ее концами, и выбрать из них самую длинную (не считая, конечно, саму строку), то такую процедуру принято называть префикс-функцией. В реализации алгоритма Кнута, Морриса и Пратта используется предобработка искомой подстроки, которая заключается в создании префикс-функции на ее основе. При этом используется следующая идея: если префикс (он же суффикс) строки длиной i длиннее одного символа, то он одновременно и префикс подстроки длиной i-1. Таким образом, проверяем префикс предыдущей подстроки, если же тот не подходит, то префикс ее префикса, и т.д. Действуя так, находим наибольший искомый префикс.

Таблица 2

Демонстрация алгоритма Кнута, Морриса и Пратта

Строка	A	B	C	A	B	C	A	A	B	C	A	B	D
Подстрока	A	B	C	A A	B B	D C	A A	B B A	D C B	A C	B A	D B	D

Точный анализ рассматриваемого алгоритма весьма сложен. Д. Кнут, Д. Моррис и В. Пратт доказывают, что для данного алгоритма требуется порядка O(m+n) сравнений символов (где n и m – длины строки и подстроки соответственно), что значительно лучше, чем при прямом поиске.

Алгоритм Бойера и Мура

Существует множество вариаций алгоритма Бойера и Мура, рассмотрим простейший из них, который состоит из следующих шагов.

Первоначально строится таблица смещений для искомой подстроки. Далее идет совмещение начала строки и подстроки и начинается проверка с последнего символа подстроки. Если последний символ подстроки и соответствующий ему при наложении символ строки не совпадают, подстрока сдвигается относительно строки на величину, полученную из таблицы смещений, и снова проводится сравнение, начиная с последнего символа подстроки. Если же символы совпадают, производится сравнение предпоследнего символа подстроки и т.д. Если все символы подстроки совпали с наложенными символами строки, значит, найдена подстрока и поиск окончен. Если же какой-то (не последний) символ подстроки не совпадает с соответствующим символом строки, далее производим сдвиг подстроки на один символ вправо и снова начинаем проверку с последнего символа. Весь алгоритм выполняется до тех пор, пока либо не будет найдено вхождение искомой подстроки, либо не будет достигнут конец строки (табл. 3). На рисунке символы, подвергшиеся сравнению, выделены жирным шрифтом.

Величина сдвига в случае несовпадения последнего символа вычисляется, исходя из следующего: сдвиг подстроки должен быть минимальным, таким, чтобы не пропустить вхождение подстроки в строке. Если данный символ строки встречается в подстроке, то смещаем подстроку таким образом, чтобы символ строки совпал с самым правым вхождением этого символа в подстроке. Если же подстрока вообще не содержит этого символа, то сдвигаем подстроку на величину, равную ее длине, так что первый символ подстроки накладывается на следующий за проверявшимся символом строки.

Величина смещения для каждого символа подстроки зависит только от порядка символов в подстроке, поэтому смещения удобно вычислить заранее и хранить в виде одномерного массива, где каждому символу алфавита соответствует смещение относительно последнего символа подстроки.

Таблица 3

Демонстрация алгоритма Бойера и Мура

Строка	A	B	C	A	B	C	A	A	B	C	A	B	D
Подстрока	A	B	C	A	B	D A	B	C A	A B	B C	D A	B	D

Таким образом, данный алгоритм является наиболее эффективным в обычных ситуациях, а его быстродействие повышается при увеличении подстроки или алфавита. В наихудшем случае трудоемкость рассматриваемого алгоритма O(m+n).

Задание к лабораторной работе

Написать программу на выбранном языке программирования, реализующую описанные выше алгоритмы для поиска подстроки в строке. Программа должна запрашивать имя входного файла. Оценить трудоемкость рассматриваемых алгоритмов.

Контрольные вопросы

Что такое поиск подстроки в строке?

Принципы работы алгоритма прямого поиска.

Принципы работы алгоритма Кнута, Морриса и Пратта.

Принципы работы алгоритма Бойера и Мура.

Как оценить трудоемкость алгоритма поиска по тексту?

Каким образом задается величина сдвига в алгоритме Кнута, Морриса и Пратта?

Каким образом задается величина сдвига в алгоритме Бойера и Мура?

Лабораторная работа №2
Реализация алгоритмов Стемминга

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы поиска по тексту однокоренных слов на основе поиска Стеммера Портера.

Краткие теоретические сведения

Стеммер Портера – алгоритм стемминга (нахождения основы слова для заданного исходного слова), опубликованный Мартином Портером. Алгоритм не использует баз основ слов, а работает, последовательно применяя ряд правил отсечения окончаний и суффиксов.

Вводим ряд определений:

Гласные буквы – а, е, и, о, у, ы, э, ю, я (буква ё считается равнозначной букве е).

RV – область слова после первой гласной. Она может быть пустой, если гласные в слове отсутствуют.

R1 – область слова после первого сочетания "гласная-согласная".

R2 – область R1 после первого сочетания "гласная-согласная".

Пример
В слове

противоестественном: RV =

тивоестественном.

R1 =

ивоестественном. R2 =

остественном.

Теперь определим несколько классов окончаний слова (табл. 4).

Таблица 4

Классы окончаний слова

Название класса	Окончания слов
PERFECTIVE	Группа 1*: в, вши, вшись.
GERUND	Группа 2: ив, ивши, ившись, ыв, ывши, ывшись.
ADJECTIVE	ее, ие, ые, ое, ими, ыми, ей, ий, ый, ой, ем, им, ым, ом, его, ого, ему, ому, их, ых, ую, юю, ая, яя, ою, ею.
PARTICIPLE	Группа 1*: ем, нн, вш, ющ, щ.
	Группа 2: ивш, ывш, ующ.
REFLEXIVE	ся, сь.
VERB	Группа 1*: ла, на, ете, йте, ли, й, л, ем, н, ло, но, ет, ют, ны, ть, ешь, нно.
	Группа 2: ила, ыла, ена, ейте, уйте, ите, или, ыли, ей, уй, ил, ыл, им, ым, ен, ило, ыло, ено, ят, ует, уют, ит, ыт, ены, ить, ыть, ишь, ую, ю.
NOUN	а, ев, ов, ие, ье, е, иями, ями, ами, еи, ии, и,ией, ей, ой, ий, й, иям, ям, ием, ем, ам, ом, о, у, ах, иях, ях, ы, ь, ию, ью, ю, ия, ья, я.
SUPERLATIVE	ейш, ейше.
DERIVATIONAL	ост, ость.
ADJECTIVAL	ADJECTIVAL определяется как ADJECTIVE или PARTICIPLE + ADJECTIVE. *Например: бегавшая = бега + вш + ая.*

* Окончаниям из группы 1 должна предшествовать буква а или я.

Правила:

1. При поиске окончания из всех возможных выбирается наиболее длинное. Например, в слове величие выбираем окончание ие, а не е.
2. Все проверки производятся над областью RV. Так, при проверке на PERFECTIVE GERUND предшествующие буквы а и я также должны быть внутри RV. Буквы перед RV не участвуют в проверках вообще.

Шаг 1

Найти окончание PERFECTIVE GERUND. Если оно существует – удалить его и завершить этот шаг.

Иначе, удаляем окончание REFLEXIVE (если оно существует). Затем в следующем порядке пробуем удалить окончания: ADJECTIVAL, VERB, NOUN. Как только одно из них найдено – шаг завершается.

Шаг 2

Если слово оканчивается на и – удаляем и.

Шаг 3

Если в R2 найдется окончание DERIVATIONAL – удаляем его.

Шаг 4

Возможен один из трех вариантов:

1. Если слово оканчивается на нн – удаляем последнюю букву.

2. Если слово оканчивается на SUPERLATIVE – удаляем его и снова удаляем последнюю букву, если слово оканчивается на нн.

3. Если слово оканчивается на ь – удаляем его.

Задание к лабораторной работе

Написать программу на выбранном языке программирования, реализующую описанный выше алгоритм для поиска по тексту однокоренных слов. Программа должна запрашивать имя входного файла и слово для поиска. Результатом работы программы должен быть файл, содержащий список однокоренных слов.

Контрольные вопросы

1. Что такое Стемминг?
2. Что такое Стеммер Портера?
3. Принципы работы алгоритма стемминга.
4. Роль морфологического анализа слов в системах поиска.
5. Аналоги Стеммер Портера.
6. Преимущества алгоритма Стемминга перед алгоритмами со словарем.
7. Недостатки алгоритма Стемминга.

Лабораторная работа №3
Кластеризация текстов по тематикам

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы кластеризации текстов по тематикам на основе латентно-семантического анализа.

Краткие теоретические сведения

Латентно-семантический анализ

Латентно-семантический анализ отображает документы и отдельные слова в так называемое «семантическое пространство», в котором и производятся все дальнейшие сравнения. При этом делаются следующие предположения:

1) Документы это просто набор слов. Порядок слов в документах игнорируется. Важно только то, сколько раз то или иное слово встречается в документе.

2) Семантическое значение документа определяется набором слов, которые как правило идут вместе. Например, в биржевых сводках, часто встречаются слова: «фонд», «акция», «доллар»

3) Каждое слово имеет единственное значение. Это, безусловно, сильное упрощение, но именно оно делает проблему разрешимой.

Пример

Для примера возьмем несколько заголовков из различных новостей. На первом шаге из этих заголовков исключаем, так называемые, стоп-символы. Это слова которые встречаются в каждом тексте и не несут в себе смысловой нагрузки, это, прежде всего, все союзы, частицы, предлоги и множество других слов. Полный список использованных стоп-символов можно посмотреть в Приложении 1.

Далее необходимо выполнить операцию стемминга. Она не является обязательной, если набор текстов достаточно большой, или если тексты на английском языке, в силу того, что количество вариаций той или иной словоформы в английском языке существенно меньше чем в русском. В нашем же случае, пропускать этот шаг не стоит т.к. это приведет к существенной деградации результатов.

Дальше исключаем слова встречающиеся в единственном экземпляре. Это тоже необязательный шаг, он не влияет на конечный результат, но сильно упрощает математические вычисления. В итоге у нас остались, так называемые, индексируемые слова, они выделены жирным шрифтом:

1. Британская **полиц**ия знает о местонахождении **основател**я **WikiLeaks**

2. В **суде США** начинается процесс **прот**ив россиянина, рассылавшего спам

3. **Церемон**ию **вручен**ия **Нобелевск**ой **прем**ии мира бойкотируют 19 **стран**

4. В **Великобритан**ии **арестова**н **основател**ь сайта **Wikileaks** Джулиан Ассандж

5. Украина игнорирует **церемон**ию **вручен**ия **Нобелевск**ой **прем**ии

6. Шведский **суд** отказался рассматривать апелляцию **основател**я **Wikileaks**

7. НАТО и **США** разработали планы обороны **стран** Балтии **прот**ив России

8. **Полиц**ия **Великобритан**ии нашла **основател**я **WikiLeaks**, но, не **арестова**ла

9.В Стокгольме и Осло сегодня состоится **вручен**ие **Нобелевск**их **прем**ий

Латентно семантический анализ

На первом шаге требуется составить частотную матрицу индексируемых слов. В этой матрице строки соответствуют индексированным словам, а столбцы — документам. В каждой ячейке матрицы указано какое количество раз слово встречается в соответствующем документе.

Таблица 5

Частотная матрица индексируемых слов

	T1	T2	T3	T4	T5	T6	T7	T8	T9
Wikileaks	1	0	0	1	0	1	0	1	0
Арестова	0	0	0	1	0	0	0	1	0
Великобритан	0	0	0	1	0	0	0	1	0
Вручен	0	0	1	0	1	0	1	0	0
Нобелевск	0	0	1	0	1	0	0	0	1
Основател	1	0	0	1	0	1	0	1	0
Полиц	1	0	0	0	0	0	0	1	0
Прем	0	0	1	0	1	0	0	0	1
Прот	0	1	0	0	0	0	1	0	0
Стран	0	0	1	0	0	0	1	0	0
суд	0	1	0	0	0	1	0	0	0
Сша	0	1	0	0	0	0	1	0	0
Церемон	0	0	1	0	1	0	0	0	0

Следующим шагом необходимо провести сингулярное разложение полученной матрицы. Т.е. исходную матрицу M мы представляем в виде:

$$M = U*W*Vt,$$

где U и Vt – ортогональные матрицы, а W – диагональная матрица. Причем диагональные элементы матрицы W упорядочены в порядке убывания. Диагональные элементы матрицы W называются сингулярными числами.

Таблица 6

Сингулярное разложение частотной матрицы индексируемых слов (матрица U)

Wikileaks	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	-0.64
Арестова	0.34	0	0.07	0.41	-0.42	-0.02	0.1	0.17	0.01
Великобритан	0.34	0	0.07	0.41	-0.42	-0.02	0.1	0.17	0.01
Вручен	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.2	-0.07
Нобелевск	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.2	0.32
Основател	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	0.64
Полиц	0.31	0	0.05	0.07	0.57	-0.6	0.29	0.37	0
Прем	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.25
Прот	0.02	0.03	-0.06	0.13	-0.05	-0.22	0	-0.25	0
Стран	0.01	0.22	-0.31	0.39	0.41	0.56	-0.22	0.4	0
суд	0.12	0.01	-0.38	-0.62	-0.3	0.12	0.21	0.55	0
Сша	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
Церемон	0	0.38	0.03	0.02	0.08	0.31	0.82	-0.29	0

Таблица 7

Сингулярное разложение частотной матрицы индексируемых слов (матрица W)

3.41	0	0	0	0	0	0	0	0
0	3.3	0	0	0	0	0	0	0
0	0	2.27	0	0	0	0	0	0
0	0	0	1.49	0	0	0	0	0
0	0	0	0	1.19	0	0	0	0
0	0	0	0	0	0.98	0	0	0
0	0	0	0	0	0	0.71	0	0
0	0	0	0	0	0	0	0.43	0
0	0	0	0	0	0	0	0	0

Таблица 8

Сингулярное разложение частотной матрицы индексируемых слов (матрица Vt)

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
0	0.02	0.65	-0.01	0.59	0	0.09	-0.01	0.47
0.03	-0.7	-0.04	0.06	0.1	-0.16	-0.67	0.09	0.09
-0.22	-0.24	0.15	0.28	-0.11	-0.68	0.44	0.33	-0.13
0.69	-0.32	0.22	-0.49	-0.12	-0.03	0.27	-0.02	-0.19
-0.27	-0.34	0.44	0.29	-0.13	0.45	0.12	-0.31	-0.45
-0.03	0.3	0.14	-0.17	0.44	-0.15	-0.3	0.24	-0.71
-0.3	0.12	0.4	-0.39	-0.53	0.12	-0.23	0.46	0.13
0.35	0.35	0.35	0.35	-0.35	-0.35	-0.35	-0.35	0

Согласно простым правилам произведения матриц, видно, что столбцы и строки, соответствующие меньшим сингулярным значениям, дают наименьший вклад в итоговое произведение. Например, мы можем отбросить последние столбцы матрицы U и последние строки матрицы V^t, оставив только первые 2. Важно, что при этом гарантируется, оптимальность полученного произведения. Разложение такого вида называют двумерным сингулярным разложением:

Таблица 9

Двумерное сингулярное разложение (матрица U)

Wikileaks	0.57	-0.01
Арестова	0.34	0
Великобритан	0.34	0
Вручен	0	0.52
Нобелевск	0	0.52
Основател	0.57	-0.01
Полиц	0.31	0
Прем	0	0.52
Прот	0.02	0.03
Стран	0.01	0.22
суд	0.12	0.01
Сша	0.02	0.03
Церемон	0	0.38

Таблица 10

Двумерное сингулярное разложение (матрица W)

3.41	0
0	3.3

Таблица 11

Двумерное сингулярное разложение (матрица Vt)

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
0	0.02	0.65	-0.01	0.59	0	0.09	-0.01	0.47

На практике, конечно, количество групп будет намного больше, пространство будет не двумерным, а многомерным, но сама идея остается той же. Можно определять местоположения слов и статей в нашем пространстве и использовать эту информацию для, например, определения тематики статьи.

В полученном факторном пространстве документы и термины концентрируются областями, имеющими общий семантический и латентный смысл.

Применительно к кластеризации, получаемые области и есть кластеры. С помощью математических преобразований можно определить центры кластеров.

Задание к лабораторной работе

Написать программу на выбранном языке программирования, реализующую описанный выше алгоритм для кластеризации заголовков по темам. Количество заголовков задать не более 10 (оптимально от 7 до 10), количество кластеров не более 3. Программа должна запрашивать заголовки (они могут храниться в файлах). После получения двумерного сингулярного разложения осуществить прямую кластеризацию самостоятельно выбранным и изученным методом (иерархические алгоритмы, поиск k-средних и т.д.). Результатом работы программы должен быть файл, содержащий заголовки, разбитые по кластерам.

Контрольные вопросы

1. Что такое латентно-семантический анализ?
2. Для решения каких задач в сети интернет используется ЛСА?
3. Что такое сингулярное разложение?
4. Как строится частотная матрица?
5. Приведите примеры алгоритмов кластеризации.
6. Что такое факторное пространство?
7. Как строится факторное пространство?

Лабораторная работа №4
Реализация алгоритма индексирования документов и поиска по индексу

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы индексирования документов и осуществлять поиск по индексу.

Краткие теоретические сведения

Инвертированный индекс

Инвертированный индекс — структура данных, в которой для каждого слова коллекции документов в соответствующем списке перечислены все места в коллекции, в которых оно встретилось. Инвертированный индекс используется для поиска по текстам.

Булев поиск

Булев поиск опирается на использование инвертированного индекса ключевых слов, т. е. таблицы, в которой для каждого ключевого слова перечисляются все документы, где оно встречается. Главным достоинством этого алгоритма является возможность связывания слов запроса логическими операциями. К недостаткам этого алгоритма следует отнести невозможность определения релевантности запросу полученной выборки документов и, как следствие, невозможность ее сортировки.

Опишем как решается задача нахождения документов в которых встречаются все слова из поискового запроса. При обработке однословного поискового запроса, ответ уже есть в инвертированном индексе — достаточно взять список соответствующий слову из запроса. При обработке многословного запроса берутся списки, соответствующие каждому из слов запроса и пересекаются.

Пример.

Пусть у нас есть корпус из трех текстов T_0 = "it is what it is", T_1 = "what is it" и T_2 = "it is a banana", тогда инвертированный индекс будет выглядеть следующим образом:

"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}

Здесь цифры обозначают номера текстов, в которых встретилось соответствующее слово. Тогда отработка поискового "what is it" запроса даст следующий результат $\{0,1\} \cap \{0,1,2\} \cap \{0,1,2\} = \{0,1\}$.

Таблицы индекса

Для эффективной организации поиска документов необходимо задействовать несколько таблиц в базе данных. В самом простом случае используются следующие три.

Таблица документов Documents. В ней хранится информация обо всех документах, проиндексированных системой, а именно название документа, его авторы, тип файла, путь к файлу/URL и т. д. При этом каждому документу необходимо присвоить уникальный идентификатор Doc_id.

Таблица ключевых слов/словарь Words. Здесь хранятся все ключевые слова системы и соответствующие им номера Word_id.

Инвертированный индекс Inverse, используемый для поиска. В этой таблице хранится идентификатор слова Word_id и соответствующий ему список документов, содержащих это слово.

Эффективная организация словаря

Одна из самых важных и трудных проблем индексации текстов связана с созданием и пополнением словаря ключевых слов. Главная сложность ее заключается в том, что для эффективной работы системы необходимо рассматривать только базовые словоформы ключевых слов..

Еще одна проблема индексирования связана с выявлением и удалением из текста так называемых стоп-слов. Они не несут смысловой нагрузки в текущей предметной области, и для эффективной работы системы их следует удалять при индексировании. Как правило, стоп-словами являются предлоги, союзы, артикли, вводные слова и т. п. Они очень часто встречаются в документах, но малоинформативны. Для их удаления можно либо использовать отдельный словарь стоп-слов, либо считать все слова с высокими частотами встречаемости в базе данных текстов стоп-словами и удалять их при индексировании.

Ранжирование

Обычно в поисковых системах после построения с помощью инвертированного индекса списка документов, содержащих слова из запроса, идет ранжирование документов из списка.

Для каждого запроса необходимо вычислить значение Score документа – показатель релевантности документа запросу, на основании которого и производится ранжирование.

Для расчета Score предлагается использовать аддитивную модель. В качестве слагаемых в данной модели предлагаются следующие:

встречаемость слов из запроса в документе (W_{single}), встречаемость пар слов из запроса в документе (W_{pair}) и встречаемость текста запроса целиком (W_{Phrase}). Помимо этого есть два слагаемых, дающих преимущество за наличие всех слов запроса в документе ($W_{AllWords}$).Итоговая формула выглядит следующим образом:

$$Score = W_{single} + W_{pair} + k_1 * W_{AllWords} + k_2 * W_{Phrase}$$

где $k_1 = 1$, $k_2 = 1/350$

Задание к лабораторной работе

Написать программу на выбранном языке программирования, реализующую индексацию документов (не менее 5 документов) описанным выше алгоритмом и осуществить поиск документа, удовлетворяющего заданный запрос. Осуществить ранжирование документов по их релевантности запросу. Программа должна запрашивать имена входных файлов и выводить заголовки и выдержки из найденных по запросу документов.

Контрольные вопросы

1. Что такое индекс?
2. Что такое инвертированный индекс?
3. Что такое прямой индекс?
4. Принципы работы алгоритма инвертированного индекса.
5. Какие форматы файлов потенциально пригодны для индексирования?
6. Что такое таблица индекса?
7. Как определять релевантность документа?
8. Что такое ранжирование документа?
9. На основе каких параметров осуществляется ранжирование документов?
10. Особенности булевого поиска.
11. Особенности вероятностного поиска.

Лабораторная работа №5
Реализация алгоритма обнаружения нечетких дубликатов

Цель работы: научиться реализовывать на выбранном языке программирования алгоритмы обнаружения нечетких дубликатов на основе алгоритма шинглов.

Краткие теоретические сведения

Проблема обнаружения нечетких дубликатов является одной из наиболее важных и трудных задач анализа данных и поиска информации. Актуальность этой проблемы определяется разнообразием приложений, в которых необходимо учитывать «похожесть», например, текстовых документов — это и улучшение качества индекса и архивов поисковых систем за счет удаления избыточной информации, и объединение новостных сообщений в сюжеты на основе сходства этих сообщений по содержанию, и фильтрация спама (как почтового, так и поискового), и установление нарушений авторских прав при незаконном копировании информации (проблема плагиата или копирайта), и ряд других.

Алгоритм шинглов

Алгоритм шинглов — алгоритм, разработанный для поиска копий и дубликатов рассматриваемого текста в веб-документе, мощный инструмент, призванный бороться с проявлениями плагиата в интернете.

Реализация алгоритма подразумевает несколько этапов:

- канонизация текстов;
- разбиение текста на шинглы;
- нахождение контрольных сумм;
- поиск одинаковых подпоследовательностей.

Канонизация текста

Канонизация текста - приведение оригинального текста к единой нормальной форме через очищение его от всех вспомогательных единиц текста (предлогов, союзов, знаков препинания, тегов и прочее), которые не должны участвовать в сравнении. Часто предполагается также удаление имен прилагательных, поскольку они, как правильно, несут эмоциональную, а не смысловую нагрузку.

Канонизация текста также требует приведения имен существительных в именительный падеж, единственное числе, а иногда – оставление только их корневых значений.

После проведения всех указанных операций получается «чистый» текст, пригодный для сравнения.

Контрольная сумма

В самом общем своем виде контрольная сумма представляет собой некоторое значение, построенное по определенной схеме на основе кодируемого сообщения.

Алгоритм CRC базируется на свойствах деления с остатком двоичных многочленов, то есть многочленов над конечным полем $GF(2^N)$. Значение CRC является по сути остатком от деления многочлена, соответствующего входным данным, на некий фиксированный порождающий многочлен.

Каждой конечной последовательности битов $a_0,a_1,\ldots,a_{N-1}$ взаимно однозначно сопоставляется двоичный полином, последовательность коэффициентов которого представляет собой исходную последовательность. Например, последовательность битов 1011010 соответствует многочлену:

$$P(x) = 1 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x^1 = x^6 + x^4 + x^3 + x^1$$

Значение контрольной суммы в алгоритме с порождающим многочленом G(x) степени N определяется как битовая последовательность длины N, представляющая многочлен R(x), получившийся в остатке при делении многочлена P(x), представляющего входной поток бит, на многочлен G(x):

$$R(x) = P(x) * x^N \bmod G(x) ,$$

где R(x) — многочлен, представляющий значение CRC, P(x) — многочлен, коэффициенты которого представляют входные данные, G(x) — порождающий многочлен, N — степень порождающего многочлена.

Рассмотрим общую схему алгоритма расчета CRC:

1. Выбрать полином P, в результате автоматически становится известна его степень N.
2. Добавить к исходной двоичной последовательности сообщения N нулевых битов. Это добавление делается для гарантированной обработки всех битов исходной последовательности.
3. Выполнив деление дополненной N нулями исходной строки S на полином P по правилам CRC арифметики. Запомнить остаток от деления, который и будет являться CRC.

Задание к лабораторной работе

Написать программу на выбранном языке программирования, реализующую поиск нечетких дубликатов заданных текстов описанным выше алгоритмом. Программа должна запрашивать имена входных файлов и выводить схожие документы и степень их схожести (в процентах).

Контрольные вопросы

1. Что такое нечеткий дубликат?
2. Что такое хеширование?
3. Что такое контрольная сумма?
4. Принципы работы алгоритма шинглов.
5. Принципы работы алгоритма CRC.
6. Что такое CRC-арифметика?
7. Что такое канонизация текстов?
8. Как формировать шинглы?
9. Как выбирать порождающий полином?

Расчетно-графическое задание и методические указания по его выполнению

Цифровой сигнал — сигнал данных, у которого каждый из представляющих параметров описывается функцией дискретного времени и конечным множеством возможных значений.

Сигналы представляют собой дискретные электрические или световые импульсы. При таком способе вся емкость коммуникационного канала используется для передачи одного сигнала. Цифровой сигнал использует всю полосу пропускания кабеля. Полоса пропускания — это разница между максимальной и минимальной частотой, которая может быть передана по кабелю. Каждое устройство в таких сетях посылает данные в обоих направлениях, а некоторые могут одновременно принимать и передавать. Узкополосные системы (baseband) передают данные в виде цифрового сигнала одной частоты.

Дискретный цифровой сигнал сложнее передавать на большие расстояния, чем аналоговый сигнал, поэтому его предварительно модулируют на стороне передатчика, и демодулируют на стороне приёмника информации. Использование в цифровых системах алгоритмов проверки и восстановления цифровой информации позволяет существенно увеличить надёжность передачи информации.

Замечание. Следует иметь в виду, что реальный цифровой сигнал по своей физической природе является аналоговым. Из-за шумов и изменения параметров линий передачи он имеет флуктуации по амплитуде, фазе/частоте (джиттер), поляризации. Но этот аналоговый сигнал (импульсный и дискретный) наделяется свойствами числа. В результате для его обработки становится возможным использование численных методов (компьютерная обработка).

Для того, чтобы представить аналоговый сигнал последовательностью чисел конечной разрядности, его следует сначала превратить в дискретный сигнал, а затем подвергнуть квантованию. Квантование является частным случаем дискретизации, когда дискретизация происходит по одинаковой величине называемой квантом. В результате сигнал будет представлен таким образом, что на каждом заданном промежутке времени известно приближённое (квантованное) значение сигнала, которое можно записать целым числом.

Если записать эти целые числа в двоичной системе, получится последовательность нулей и единиц, которая и будет являться цифровым сигналом (рисунок).

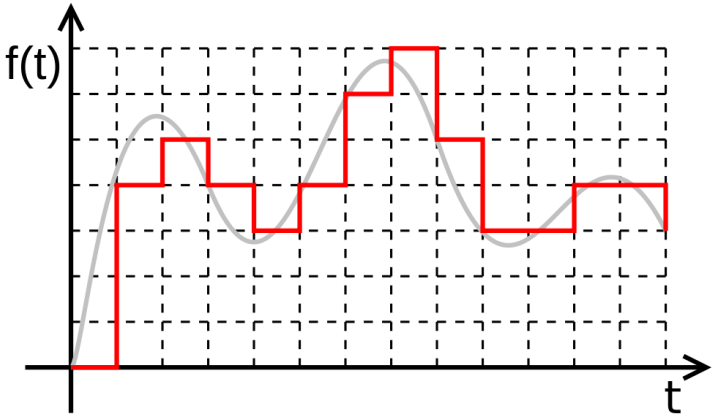

Рис. Преобразование аналогово сигнала в цифровой сигнал

Задание

На выбранном языке программирования реализовать алгоритм преобразования аналогового сигнала F(x) в цифровой сигнал, а также отобразить аналоговый и цифровой сигнал графически на промежутке от 0 до 10 сек. Параметры задания аналогового сигнала и частоту дискретизации взять из табл.

Таблица

Последняя цифра номера студенческого билета	0	1	2	3	4	5	6	7	8	9
F(x)	sin	cos	sin	cos	sin	cos	sin	cos	sin	cos
Период F(x), сек	3	4	5	3	4	5	3	4	5	3
Частота дискретизации, сек	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25

Приложения

Приложение 1 Стоп-символы русского языка

Стоп символы, они же стоп-слова, это слова, встречающиеся практически во всех текстах и не несущие специальной смысловой нагрузки. В русском языке, к стоп символам относятся предлоги, суффиксы, причастия, междометия, частицы и т.п. Неполный список стоп-слов представлен ниже:

-	еще	него	сказать
а	ж	нее	со
без	же	ней	совсем
более	жизнь	нельзя	так
больше	за	нет	такой
будет	зачем	ни	там
будто	здесь	нибудь	тебя
бы	и	никогда	тем
был	из	ним	теперь
была	из-за	них	то
были	или	ничего	тогда
было	им	но	того
быть	иногда	ну	тоже
в	их	о	только
вам	к	об	том
вас	кажется	один	тот
вдруг	как	он	три
ведь	какая	она	тут
во	какой	они	ты
вот	когда	опять	у
впрочем	конечно	от	уж
все	которого	перед	уже
всегда	которые	по	хорошо
всего	кто	под	хоть
всех	куда	после	чего
всю	ли	потом	человек
вы	лучше	потому	чем
г	между	почти	через
где	меня	при	что
говорил	мне	про	чтоб
да	много	раз	чтобы
даже	может	разве	чуть
два	можно	с	эти
для	мой	сам	этого
до	моя	свое	этой
другой	мы	свою	этом

его	на	себе	этот
ее	над	себя	эту
ей	надо	сегодня	я
ему	наконец	сейчас	
если	нас	сказал	
есть	не	сказала	

Помимо указанных слов имеет смысл еще фильтровать цифры, отдельные буквы и знаки препинания.

Приложение 2 Пример оформления лабораторной работы

Лабораторная работа №1
Реализация алгоритмов поиска по тексту

студента группы ИТ-31
Андросовой Татьяны Владимировны

*Выполнение:*_____ *Защита:*_____

Содержание отчета:
1. Титульный лист.
2. Цель работы.
3. Код программы на выбранном языке программирования.
4. Скриншоты результатов работы программы
5. Выводы.

Библиографический список

1. Бабич А. В. Эффективная обработка информации. Mind mapping для студентов и профессионалов [Электронный учебник] : учебное пособие / Бабич А. В., 2011, БИНОМ. Лаборатория знаний, Интернет-Университет Информационных Технологий (ИНТУИТ). - 223 с.
2. Гринберг А. С. Информационный менеджмент [Электронный учебник] :учебное пособие / Гринберг А. С., 2012, ЮНИТИ-ДАНА. - 415 с.
3. Рудинский И. Д. Технология проектирования автоматизированных систем обработки информации и управления [Электронный учебник] : учебное пособие / Рудинский И. Д., 2011, Горячая линия - Телеком. - 304 с.
4. Балюкевич Э. Л. Теория информации [Электронный учебник] : учебное пособие / Балюкевич Э. Л., 2009, Евразийский открытый институт. - 215 с.
5. Белов В. М. Теория информации [Электронный учебник] : учебное пособие / Белов В. М., 2012, Горячая линия - Телеком. - 143 с.
6. Лидовский В. В. Теория информации [Электронный учебник] : учебное пособие / Лидовский В. В., 2013, МЦНМО. - 111 с.
7. Стефанова Н. Л. Основы математической обработки информации [Электронный учебник] : учебное пособие для организации самостоятельной деятельности студентов / Стефанова Н. Л., 2011, Российский государственный педагогический университет им. А.И. Герцена. - 133 с.
8. Петров, И.Б., Лобанов, А.И. Лекции по вычислительной математике: Учебное пособие .- г. Москва : БИНОМ, 2006. – 235 с.

Учебное издание

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Методические указания к выполнению лабораторных работ
и расчетно-графического задания
для студентов 3-го курса направления подготовки
09.03.02 - Информационные системы и технологии

Составитель: **Кочеткова** Инесса Андреевна

Подписано в печать 11.05.18. Формат 60x84/16. Усл. печ. л.
Тираж экз. Заказ Цена
Отпечатано в Белгородском государственном технологическом университете им. В. Г. Шухова
308012, г. Белгород, ул. Костюкова, 46