ДНЕПРОПЕТРОВСКИЙ УНИВЕРСИТЕТ
имени АЛЬФРЕДА НОБЕЛЯ

Ю.К. Тараненко
М.Р. Кабанова
Н.А. Черняк

КОМПЬЮТЕРНАЯ ЛИНГВИСТИКА

ДНЕПРОПЕТРОВСКИЙ УНИВЕРСИТЕТ
имени АЛЬФРЕДА НОБЕЛЯ

Ю.К. ТАРАНЕНКО
М.Р. КАБАНОВА
Н.А. ЧЕРНЯК

КОМПЬЮТЕРНАЯ ЛИНГВИСТИКА

УЧЕБНОЕ ПОСОБИЕ

Днипро
2016

УДК 81'42:004.2
ББК 81.2-5с51
Т 19

Рекомендовано
учёным советом Днепропетровского университета
имени Альфреда Нобеля
(протокол № 8 от 20 октября 2016 г.)

Рецензенты:
Д.Г. Зеленцов, заведующий кафедрой информационных систем
Украинского государственного химико-технологического университета,
доктор технических наук, профессор;
В.В. Зирка, профессор кафедры английской филологии и перевода
Днепропетровского университета имени Альфреда Нобеля,
доктор филологических наук, профессор;
Е.В. Шкурко, доцент кафедры перевода и лингвистической подготовки иностранцев
Днепропетровского национального университета имени
Олеся Гончара, кандидат филологических наук, доцент.

У навчальному посібнику вперше розглянуто методи автоматичного, синтаксичного та латентно-семантичного аналізу (ЛСА) текстів. Перша частина посібника містить матеріали з лексико-синтаксичного аналізу речень англійської мови для вирішення завдань комп'ютерної лінгвістики. У другій частині посібника у формі практичних робіт детально викладено авторську методику ЛСА текстів у вигляді коротких анотацій, що містять набір ключових слів і словосполучень.

Навчальний посібник може бути корисним для студентів, аспірантів і тих, хто займається проблемами прикладної лінгвістики.

Т 19 **Тараненко Ю.К.**

Компьютерная лингвистика: учебное пособие / Ю.К. Тараненко, М.Р. Кабанова, Н.А. Черняк. – Днипро: Университет имени Альфреда Нобеля, 2016. – 112 с.

ISBN 978-966-434-373-9

В учебном пособии впервые рассмотрены методы автоматического, синтаксического и латентно-семантического анализа (ЛСА) текстов. Первая часть пособия содержит материалы по лексико-синтаксическому анализу предложений английского языка для решения задач компьютерной лингвистики. Во второй части пособия в форме практических работ подробно изложена авторская методика ЛСА текстов в виде коротких аннотаций, содержащих набор ключевых слов и словосочетаний.

Учебное пособие может быть полезно студентам, аспирантам и тем, кто занимается проблемами прикладной лингвистики.

УДК 81'42:004.2
ББК 81.2-5с51

ISBN 978-966-434-373-9

СОДЕРЖАНИЕ

Часть первая

АВТОМАТИЗИРОВАННЫЙ СИНТАКСИЧЕСКИЙ АНАЛИЗ

1. ВВЕДЕНИЕ

Все мы, так или иначе, каждый день сталкиваемся с необходимостью понять и оценить разнообразные тексты, предварительно подвергнув их анализу с учётом имеющихся задач. В программе обучения в средней школе учащиеся совершенствуют свои умения и навыки в понимании любого текста и постепенно вырабатывают собственную модель анализа текста художественного произведения, которая включает литературоведческие, лингвистические, лингво-стилистические, экстралингвистические ресурсы и т. д. Проблема заключается в том, что на современном этапе развития науки всё еще нет единого определения понятия «лингвистический анализ».

Несомненно, из-за множества подходов к этой проблеме возникает опасность или упустить какие-то важные моменты, или, например, удалиться только в одном направлении: в литературоведческий анализ, с его идеей исключительности художественных особенностей, или, к примеру, сосредоточиться на грамматическом строе языка, а возможно, на функционировании лексических средств в тексте.

Однако независимо от того, какова конечная цель нашего анализа, хорошо было бы решить некоторые предварительные задачи, ответив на следующие вопросы:

– Что мы знаем об авторе текста в плане историко-культурологического анализа [1; 2]?

– Есть ли чётко выраженные признаки авторского стиля речи (признаки стиля художественного произведения, научно-популярного, делового и т. п.) [2]?

– Можем ли мы определить тему, идею текста с учётом структурной композиции и анализа функционирования заглавия, начала, окончания текста [3]?

– Можно ли говорить об авторской системе образов в тексте произведения [9]?

– Что можно сказать об актуализации языковых единиц в тексте на морфемном, лексическом, синтаксическом уровнях [7; 8]?

Исходя из всего вышесказанного, логично утверждать, что для полноценного анализа необходимо владеть достижениями следующих дисциплин:

– **фонетики**, как раздела языкознания, изучающего способы образования и акустические свойства звуков речи человека;

– **морфологии**, изучающей слово как часть речи, включая его строение и изменения в соответствии с грамматическими категориями;

– **лексикологии**, которая изучает словарный состав языка, или лексику;

– **семантики**, подразумевающей содержательную сторону (т. е. план содержания) языковых единиц: морфем, слов, словосочетаний. При этом семантика непосредственно связана с аспектом семиотики (науки о знаковых системах, включая языковые знаки), изучая структуру и функционирование знаков в аспектах семантики, синтактики и прагматики;

– **грамматики**, как раздела науки о языке, предполагающего изучение строения, образования, изменения и функционирования слов, словосочетаний и предложений в плане их смысловой и формальной структур [4].

С введением в вузовскую программу обучения новых технологий мы получим возможность использовать компьютер для решения наших задач. Тем более всё актуальнее становится понятие «достоверность», а это значит, что нужны более точные и объективно выверенные результаты, в том числе и тестовые маркеры, шаблоны для процесса поиска необходимой текстовой информации и т. п. [12–19].

Таким образом, исследование языка любого текста все больше выходит за рамки сугубо лингвистического пони-

мания вопроса, вовлекая в поле зрения специалистов широкий круг явлений, присущих компьютерной лингвистике (или прикладной лингвистике). В руках профессионалов оказывается именно такой инструмент, как прикладная лингвистика, который облегчит понимание и изучение правил построения связного текста, методов передачи ко-референции мира и предмета, видо-временных презентаций и, конечно же, проблем актуального членения предложения. С другой стороны, не будем отрицать следующие возможности компьютерных реалий: программ, компьютерных технологий организации и обработки данных, что важно для моделирования функционирования языка в различных условиях, в том числе и в смежных с прикладной лингвистикой дисциплинах.

С учетом всего вышесказанного остановимся на решении тех задач, которые будут рассматриваться далее:

1. Идентификация последовательности слов, n-граммы, анализ структуры слова.

2. Морфологический анализ слов с целью узнать их значение.

Эти методы не касаются исследования предложений как грамматической конструкции, построенной из одного или нескольких слов определенного языка. Грамматическая конструкция содержит отдельную, относительно независимую мысль. Это смысловое, грамматическое и интонационное целое, выражающее какое-либо мнение в отношении ее к действительности (предикативность, созданная категориями модальности, времени и лица) одним словом или сочетанием слов.

Автоматизированный синтаксический анализ решает задачу обработки неограниченного множества произвольных предложений. Для решения такой сложной и трудоёмкой задачи не обойтись без разработки специального программного обеспечения для анализа структуры предложения и исследования значения этих предложений. С решением этой задачи тесно связана задача выбора оптимального, как с точки зрения гибкости, так и по возможностям обработки больших текстовых массивов, языка программирования, который изучают прикладные лингвисты. Здесь нет альтернативы свободно распространяемому достаточно мощнону и хорошо адаптированному для работы с корпусами текстов свободно компилированному языку Python.

2. РОЛЬ ГРАММАТИК В АВТОМАТИЗИРОВАННОМ СИНТАКСИЧЕСКОМ АНАЛИЗЕ

Начнём с одного из свойств предложений, по которому меньшее по количеству слов предложение становится составной частью большего. Рассмотрим примеры:

(1) a. ***Usain Bolt broke the 100m record***

b. The Jamaica Observer reported that **Usain Bolt broke the 100m record**

c. Andre said The Jamaica Observer reported that **Usain Bolt broke the 100m record**

d. I think Andre said the Jamaica Observer reported that **Usain Bolt broke the 100m record**

Заменим первое предложение на символ **S**, тогда следующие предложения строятся по шаблонам, такими как **Andre said S** и **I think S**. Эти шаблоны и подобные шаблоны (**S but S,** и **S when S**) позволяют на основе одного предложения построить большие предложения. По подобным шаблонам построено огромное предложение в сказке «Винни-Пух».

Winnie the Pooh story by A.A. Milne, In which Piglet is Entirely Surrounded by Water: [***You can imagine Piglet's joy when at last the ship came in sight of him.***] In after-years he liked to think that he had been in Very Great Danger during the Terrible Flood, but the only danger he had really been in was the last half-hour of his imprisonment, when Owl, who had just flown up, sat on a branch of his tree to comfort him, and told him a very long story about an aunt who had once laid a seagull's egg by mistake, and the story went on and on, rather like this sentence, until Piglet who was listening out of his window without much hope, went to sleep quietly and naturally, slipping slowly out of the window towards the water until he was only hanging on by his toes, at which moment, luckily, a sudden loud

squawk from Owl, which was really part of the story, being what his aunt said, woke the Piglet up and just gave him time to jerk himself back into safety and say, “How interesting, and did she?” when – well, ***you can imagine his joy when at last*** he saw the good ship, Brain of Pooh (Captain, C. Robin; 1st Mate, P. Bear) coming over the sea to rescue him...

Этому предложению соответствует простая структура, начиная с ***S but S when S***. Из этого примера можно сделать вывод, что языку свойственны конструкции, которые позволяют условно бесконечно расширять предложение. Кроме того, мы можем понять предложение произвольной длины, которые раньше никогда не слышали. Достаточно легко придумать совсем новое предложение, которое никогда ранее нигде в данном естественном языке не встречалось, а все носители языка его поймут.

Цель грамматики – дать явное описание естественного языка. Чтобы описать язык, нужно определить, что считать естественным языком, и изучить основные подходы к его представлению.

В данном случае мы рассмотрели формальное представление порождающей грамматики, согласно которой язык представляется как множество всех грамматически верных предложений. В свою очередь, грамматика как формальная система может быть использована для генерации элементов этого множества.

3. ГРАММАТИКИ ДВУСМЫСЛЕННОСТИ (НЕОДНОЗНАЧНОСТИ) В ПРЕДЛОЖЕНИЯХ

Известный пример неоднозначности (2), из фильма Groucho Marx, Animal Crackers (1930):

(2) While hunting in Africa, I shot an elephant in my pajamas. How an elephant got into my pajamas I'll never know.

Для рассмотрения неоднозначности выражения "***I shot an elephant in my pajamas***" сначала нужно построить простую грамматику. Для этого воспользуемся следующими синтаксическими категориями (табл. 1.1).

Таблица 1.1

Синтаксические категории

Символ	Значение	Пример
S	sentence (предложение)	the man walked (человек шёл)
NP	noun phrase (фраза существительного)	a dog (собака)
VP	verb phrase (фраза глагола)	saw a park (видел парк)
PP	prepositional phrase (предложная фраза)	with a telescope (с телескопом)
Det	Determiner (детерминатив)	the
N	noun (существительное)	dog (собака)
V	verb (глагол)	walked (шёл)
P	preposition (предлог)	in (в)
|	The universal logic operator or	"saw" | "ate" | "walked"

Построим простую грамматику предложения "***I shot an elephant in my pajamas***"

(""""

S -> NP VP

```
PP -> P NP
NP ->Det N | Det N PP | 'I'
 VP -> V NP | VP PP
Det -> 'an' | 'my'
 N -> 'elephant' | 'pajamas'
V -> 'shot'
P -> 'in'
... """)
```

Для более сложных предложений нам может понадобиться полная таблица тегов всех частей речи и специальных символов (табл. 1.2).

Таблица 1.2

Теги частей речи и специальных символов

Tag	Definition	Tag	Definition
.	sentence closer (. ; ? *)	NN$	possessive singular noun
(	left paren	NNS	plural noun
)	right paren	NNS$	possessive plural noun
*	not, n't	NP	proper noun or part of name phrase
--	Dash	NP$	possessive proper noun
,	Comma	NPS	plural proper noun
:	Colon	NPS$	possessive plural proper noun
ABL	pre-qualifier (quite, rather)	NR	adverbial noun (home, today, west)
ABN	pre-quantifier (half, all)	OD	ordinal numeral (first, 2nd)
ABX	pre-quantifier (both)	PN	nominal pronoun (everybody, nothing)
AP	post-determiner (many, several, next)	PN$	possessive nominal pronoun
AT	article (a, the, no)	PP$	possessive personal pronoun (my, our)
BE	be	PP$$	second (nominal) possessive pronoun (mine, ours)
BED	were	PPL	singular reflexive/intensive personal pronoun (myself)
BEDZ	was	PPLS	plural reflexive/intensive personal pronoun (ourselves)

Продолжение табл. 1.2

Tag	Definition	Tag	Definition
BEG	being	PPO	objective personal pronoun (me, him, it, them)
BEM	am	PPS	3rd singular nominative pronoun (he, she, it, one)
BEN	been	PPSS	other nominative personal pronoun (I, we, they, you)
BER	are, art	PRP	Personal pronoun
BEZ	is	PRP$	Possessive pronoun
CC	coordinating conjunction (and, or)	QL	qualifier (very, fairly)
CD	cardinal numeral (one, two, 2, etc.)	QLP	post-qualifier (enough, indeed)
CS	subordinating conjunction (if, although)	RB	adverb
DO	do	RBR	comparative adverb
DOD	did	RBT	superlative adverb
DOZ	does	RN	nominal adverb (here, then, indoors)
DT	singular determiner/ quantifier (this, that)	RP	adverb/particle (about, off, up)
DTI	singular or plural determiner/quantifier (some, any)	TO	infinitive marker to
DTS	plural determiner (these, those)	UH	interjection, exclamation
DTX	determiner/double conjunction (either)	VB	verb, base form
EX	existential there	VBD	verb, past tense
FW	foreign word (hyphenated before regular tag)	VBG	verb, present participle/gerund
HV	have	VBN	verb, past participle
HVD	had (past tense)	VBP	verb, non 3rd person, singular, present
HVG	having	VBZ	verb, 3rd. singular present

Окончание табл. 1.2

Tag	Definition	Tag	Definition
HVN	had (past participle)	WDT	wh- determiner (what, which)
IN	preposition	WP$	possessive wh- pronoun (whose)
JJ	adjective	WPO	objective wh- pronoun (whom, which, that)
JJR	comparative adjective	WPS	nominative wh- pronoun (who, which, that)
JJS	semantically superlative adjective (chief, top)	WQL	wh- qualifier (how)
JJT	morphologically superlative adjective (biggest)	WRB	wh- adverb (how, where, when)
MD	modal auxiliary (can, should, will)	NC	cited word (hyphenated after regular tag)
		NN	singular or mass noun

1. Создаём модуль Proc_1.py на Python для анализа неоднозначности:

```
#!/usr/bin/env python
# coding: utf-8
Import nltk
groucho_grammar = nltk.parse_cfg("""
   S -> NP VP
   PP -> P NP
   NP ->Det N | Det N PP | 'I'
   VP -> V NP | VP PP
Det -> 'an' | 'my'
   N -> 'elephant' | 'pajamas'
   V -> 'shot'
   P -> 'in'
""")

sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
trees = parser.nbest_parse(sent)
for tree in trees:
    print tree
```

Результаты работы процедуры:

```
(S
  (NPI)
(VP
    (V shot)
     (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
(S
  (NP I)
  (VP
    (VP (V shot) (NP (Det an) (N elephant)))
          (PP (P in) (NP (Det my) (N pajamas)))))
```

Приведенная выше грамматика позволяет осуществить анализ предложения двумя способами в зависимости от того, является ли выражение **in my pajamas** предложным, а также описывает ли оно слова или событие «стрельбы». Программа создает две структуры, которые, с учетом скобок, можно представить в виде деревьев.

4. ГРАММАТИКИ БИГРАММ

Построим следующие примеры вычислений биграмм на основе детской истории – The Adventures of Buster Brown (http://www.gutenb

erg.org/files/22816/22816.txt):

(3) a. **He roared with me the pail slip down his back**

b. **The worst part and clumsy looking for whoever heard light**

Интуитивно известно, что эти последовательности – просто набор слов, но, возможно, важно определить, какие именно ошибки содержат эти последовательности. Если рассмотреть последовательность **the worst part and clumsy looking**, то можно заметить, что эти два выражения соединены союзом координации (**and, but, or**). Эту структуру координации упрощенно, согласно синтаксису, можно описать так:

Если **v1 и v2** выражения грамматической категории **X**, тогда **v1 and v2** также выражения категории **X**.

Несколько примеров таких структур (4). В первом примере NP существительное получено путем сочетания двух NPs, во втором – сочетание двух APs прилагательных позволило получить один AP.

(4) a. **The book's ending was (NP the worst part and the best part) for me.**

b. **On land they are (AP slow and clumsy looking).**

Совместить NP и AP невозможно, потому что выражение **the worst part and clumsy looking** является не грамматическим. Чтобы двигаться дальше, необходимо остановиться на понятии «структура составляющих», или «структура непосредственных составляющих».

Понятие структуры составляющих базируется на основе того, что слова могут сочетаться с другими словами в отдельные самостоятельные единицы. Определить, что последовательность слов выступает отдельной единицей, можно на основе признака – замещения, когда последовательность слов в «правильном» предложе-

нии может быть заменена более короткой последовательностью, и это не повлияет на «правильность» предложения. Для понимания этого утверждения рассмотрим следующее предложение:

(5) **The little bear saw the fine fatt routin the brook.**

Поскольку последовательность **The little bear** можно заменить на **He,** то это указывает, что **The little bear** – отдельная самостоятельная единица. Наоборот, нельзя аналогичным образом заменить **little bear saw**.

(6) a. **He saw the fine fat trout in the brook.**

b. ***The he the fine fat trout in the brook**.

5. ПРОСТЫЕ ГРАММАТИКИ

Рассмотрим простую контекстно-свободную грамматику. Согласно определению первым символом слева в первом правиле грамматики является специальный начальный символ **S**, и все деревья должны иметь этот символ как корень. В NLTK контекстно-свободная грамматика определяется в модуле **nltk.grammar**. В следующем примере в модуле Proc_2.py показано, каким образом можно определить грамматику и осуществить синтаксический анализ простого предложения, которое предусмотрено этой грамматикой:

```
#!/usr/bin/env python
 # coding: utf-8
importnltk
  grammar1 = nltk.parse_cfg("""
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  V -> "saw" | "ate" | "walked"
  NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
  N -> "man" | "dog" | "cat" | "telescope" | "park"
  P -> "in" | "on" | "by" | "with"
  """)
sent = "Mary saw Bob".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.nbest_parse(sent):
                        printtree
```

Результаты работы программы:

```
(S (NP Mary) (VP (V saw) (NP Bob)))
```

При разработке грамматики использовалось правило **VP -> V NP | V NP PP**, содержащее оператор дизъюнкции в правой части, и это заменяет запись двух правил **VP -> V NP** и **VP -> V NP PP.**

В результате синтаксического анализа предложения **The dog saw aman in the park** с использованием такой простой грамматики получаем два дерева.

Поскольку грамматика позволяет построить два дерева для этого предложения, то это предложение называют структурно неоднозначным. К неоднозначности приводит двусмысленность присоединения предложного выражения. Предложное выражение **PP in the park** может быть присоединено к двум местам в дереве: или это дочка **VP**, или дочь **NP**. В зависимости от места присоединения предложного выражения меняется не только синтаксическая структура предложения, но и его содержание.

6. РАЗРАБОТКА СОБСТВЕННЫХ ГРАММАТИК

При разработке собственной контекстно-свободной грамматики ее правила можно записывать и редактировать в обычном текстовом файле, например mygrammar.txt, и использовать в NLTK. Можно загружать из файла mygrammar.txt и использовать ее для синтаксического анализа, как показано в следующем примере (модуль Proc_3.py):

```
#!/usr/bin/env python
# coding: utf-8
importnltk
f = open('mygrammar.txt', "r")
text=f.read()
f.close()
grammar1 = nltk.parse_cfg(text)
sent = "Mary saw Bob".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.nbest_parse(sent):
        prin ttree
```

При разработке контекстно-свободной грамматики с целью ее использования в NLTK нельзя в правой части правил сочетать грамматические категории и лексические единицы, например, правило **PP -> 'of' NP не допускается**. Если нужно в правиле использовать лексическую единицу из нескольких слов, то вместо **NP -> 'NewYork'** нужно записать **NP -> 'New_York' instead.**

7. РЕКУРСИЯ В СИНТАКСИЧЕСКИХ СТРУКТУРАХ

Грамматику называют рекурсивной, если категории с левой части ее правил также встречаются и в правых частях, как показано в следующем примере. Правило **Nom -> AdjNom** содержит прямую рекурсию категории **Nom**, тогда как непрямая рекурсия **S** возникает из комбинации двух правил **S -> NP VP** и **VP -> V S.**

```
grammar2 = nltk.parse_cfg("""
  S  -> NP VP
  NP ->DetNom | PropN
Nom ->AdjNom | N
  VP -> V Adj | V NP | V S | V NP PP
  PP -> P NP
PropN -> 'Buster' | 'Chatterer' | 'Joe'
Det -> 'the' | 'a'
  N -> 'bear' | 'squirrel' | 'tree' | 'fish' | 'log'
Adj  -> 'angry' | 'frightened' |  'little' | 'tall'
  V ->  'chased'  | 'saw' | 'said' | 'thought' | 'was' | 'put'
  P -> 'on'
        """)
```

Рекурсия этой грамматики позволяет строить, например, деревья с вложенными существительными и вложенными предложениями.

8. СИНТАКСИЧЕСКИЙ АНАЛИЗ НА ОСНОВЕ КОНТЕКСТНО-СВОБОДНОЙ ГРАММАТИКИ

Парсер – это программа, которая обрабатывает входящие предложения согласно правилам грамматики и строит одну или несколько синтаксических структур, которые отвечают (согласовываются) этой грамматике. Грамматика – это декларативная спецификация определенных закономерностей, это на самом деле лента (набор лент), а не программа. Анализатор осуществляет процедурную интерпретацию грамматики – ищет среди леса деревьев, которые может порождать грамматика, одно нужное дерево, соответствующее входному предложению.

Далее будут рассмотрены два простых алгоритма синтаксического анализа: алгоритм рекурсивного спуска, который относится к стратегии сверху-вниз, и алгоритм перемещения-сворачивания, который относится к стратегии снизу-вверх. Также будет рассматриваться более сложный алгоритм анализа, который предполагает осуществление анализа сверху-вниз с фильтрованием снизу-вверх – алгоритм left-cornerparser.

9. АЛГОРИТМ РЕКУРСИВНОГО СПУСКА

Простейший тип анализатора интерпретирует грамматику как спецификацию (указания) для преобразования (разделения) элемента высшего уровня на несколько элементов низшего уровня. Начальная задача – найти символ (элемент) S. Правило S → NP VP позволяет анализатору заменить (преобразовать) этот элемент на два других элемента: найти сначала NP а потом VP. Каждый из этих элементов может быть преобразован в другие элементы на основе правил грамматик, в левых частях которых есть правила NP и VP. Такой процесс будет продолжаться до тех пор, пока не будет поставлена задача найти (заменить) элемент на слово, например **telescope**. Тогда происходит сравнение этого слова со словом из входной последовательности слов, и если устанавливается соответствие между этими словами, то алгоритм продолжает свою работу. Если соответствие не устанавливается, то анализатор возвращается на шаг назад и пробует рассмотреть другие варианты.

Алгоритм рекурсивного спуска синтаксического анализа строит дерево разбора согласно описанному выше процессу. Для детального анализа работы данного алгоритма целесообразно воспользоваться демонстрационным примером (C:\Python27\Lib\site-packages\nltk\app\ rdparser.py или в IDLE **nltk.app.rdparser()**.

Во время своей работы анализатор часто вынужден выбирать между двумя возможными правилами. Например, при переходе от шага 3 к шагу 4 программа пробует найти правила с N в левой части. Сначала будет найдено N → man. Когда это правило использовать не удается, происходит возврат на шаг назад и снова осуществляется поиск правила N до тех пор, пока не будет найдено N → dog, которое соответствует следующему слову в предложении. Гораздо позже, как показано на шаге 5, происходит завершение построения (нахождения) полного дерева. Это дерево полностью охватывает входное предложение без висячих ребер. Как только дерево построено, можно выполнять другие действия – со-

вершать дополнительные шаги анализа. Это предполагает возврат на шаг назад и исследование других альтернативных правил, если они позволяют так же получить результаты анализа – построить дерево.

В NLTK реализован синтаксический анализатор рекурсивного спуска. Воспользоваться им можно способом, приведенным в модуле Proc_4.py Python:

```
#!/usr/bin/envpython
# coding: utf-8
importnltk
grammar1 = nltk.parse_cfg("""
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  V -> "saw" | "ate" | "walked"
  NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
  N -> "man" | "dog" | "cat" | "telescope" | "park"
  P -> "in" | "on" | "by" | "with"
  """)
rd_parser = nltk.RecursiveDescentParser(grammar1)
sent = 'Marysaw a dog'.split()
for tinrd_parser.nbest_parse(sent):
print t
```

Результаты работы модуля:

```
(S (NP Mary) (VP (V saw) (NP (Det a) (N dog)))
```

RecursiveDescentParser() может иметь необязательный параметр **trace**. Если значение этого параметра больше нуля, то анализатор выводит на экран результаты выполнения всех шагов синтаксического анализа текста.

Анализатор рекурсивного спуска имеет три основных недостатка. Первый – леворекурсивные правила вида **NP -> NP PP** приводят к бесконечному циклу. Второй – при анализе много времени тратится на рассмотрение слов и структур (деревьев), которые не соответствуют введенному предложению. Третий недостаток перебора с возвратом состоит в том, что отвергаются проанализирован-

ные составляющие, и построение семантического дерева начинается сначала. Например, при возврате на шаг назад после построения фрагмента (поддерева) **over VP -> V NP** поддерево для NP будет потеряно. Если анализатор будет обрабатывать правило VP -> V NP PP, то поддерево NP снова будет необходимо построить.

Алгоритм рекурсивного спуска соответствует стратегии осуществления синтаксического анализа сверху-вниз. Согласно этой стратегии грамматика используется для предсказания того, что будет на входе (входными данными является предложение) перед просмотром этих входных данных. Поскольку входящее предложение известно, то целесообразно начинать анализ с рассмотрения этого предложения. На таком подходе базируется стратегия снизу-вверх, согласно которой работает алгоритм синтаксического анализа перемещения-сворачивания.

10. АЛГОРИТМ ПЕРЕМЕЩЕНИЯ-СВОРАЧИВАНИЯ

Простейший анализатор снизу-вверх – это анализатор перемещения-сворачивания. Так же, как и другие анализаторы снизу-вверх, этот анализатор пытается найти последовательность слов и словосочетаний (выражений), которые соответствуют правой части правила грамматики, и заменяет их на левую часть правила до тех пор, пока предложение не «свернется» до символа S.

Анализатор перемещения-сворачивания неоднократно помещает следующее входное слово в стек, и эта операция называется перемещением. Если N элементов с вершины стека соответствуют N элементам в правой части некоторого правила грамматики, тогда удаляют из стека элемент левой части правила, и он помещается в стек. Эта замена N элементов на вершине стека на один элемент называется операцией сворачивания. Эта операция осуществляется только по отношению к вершине стека, сокращение элементов, которые находятся глубже в стеке, должно быть завершено до того, как на вершину стека попадают новые элементы. Анализатор завершает свою работу, когда проработаны все элементы, поступающие ему на вход, и когда в стеке находится только один элемент – дерево разбора с корнем S. Анализатор перемещения-сворачивания строит дерево вывода в течение всего процесса выполнения. В каждый из моментов извлечения из стека N элементов анализатор компонует из этих элементов частичное дерево вывода и помещает его в стек. Подробно работу анализатора перемещения-сворачивания можно исследовать, воспользовавшись демонстрационным примером **nltk.app.srparser()**.

В NLTK реализовано **ShiftReduceParser()** как простой анализатор перемещения-сворачивания. Эта программа синтаксического анализа не предусматривает осуществление операций возврата

на шаг назад, что не гарантирует построение дерева разбора для текста, даже если такое дерево существует. Кроме этого, строится только одно дерево, даже если возможно построить несколько деревьев. При работе этой программы можно использовать параметр для контроля за работой анализатора и отображения результатов анализа по шагам. Алгоритм перемещения-сворачивания реализован в модуле Proc_5.py:

```
#!/usr/bin/envpython
# coding: utf-8
importnltk
grammar1 = nltk.parse_cfg("""
  S -> NP VP
  VP -> V NP | V NP PP
  PP -> P NP
  V -> "saw" | "ate" | "walked"
  NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
  N -> "man" | "dog" | "cat" | "telescope" | "park"
  P -> "in" | "on" | "by" | "with"
  """)
sr_parse = nltk.ShiftReduceParser(grammar1)
sent = 'Marysaw a dog'.split()
printsr_parse.parse(sent)
```

Анализатор перемещения-сворачивания может попадать в тупики, что не позволит получить какие-либо результаты анализа, даже если входное предложение не противоречит грамматике. Если это происходит, значит уже нет входных слов, и стек содержит элементы, которые нельзя свернуть в S. Эти проблемы возникают вследствие действий на предыдущих шагах. Возможна ситуация, когда есть более чем один вариант осуществления сворачивания и когда можно осуществлять как сворачивание, так и перемещение.

В анализаторе обычно реализуются определенные методы для решения таких конфликтов. Например, для разрешения конфликта сворачивания-перемещения, перемещение выполняется только тогда, когда никакое сворачивание уже невозможно. Конфликт перемещения-сворачивания решается путем выбора той операции сворачивания, которая охватывает больше элементов из стека.

Преимуществом анализатора перемещения-сворачивания над анализатором рекурсивного спуска является то, что он строит синтаксическую структуру, которая соответствует входной последовательности слов. Кроме этого, он строит каждую подструктуру отдельно. Например, **NP(Det(the), N(man))** строится независимо от того, будет ли она использоваться при сворачивании правила **VP -> V NP PP** или **NP -> NP PP**.

11. АЛГОРИТМ LEFT-CORNER

Одной из проблем анализатора рекурсивного спуска является его циклевка, если встречается правило с левой рекурсией. Это происходит потому, что правила грамматики применяются без рассмотрения входного предложения. Анализатор left-corner – это сочетание анализаторов снизу-вверх и сверху-вниз.

Грамматика grammar1 позволяет осуществить анализ ***John saw Mary***.

Эта грамматика содержит следующие правила для ***NP***:

(7) a. **NP → dt nom (stale!)**

b. **NP → dt nom pp (stale!)**

c. **NP → propn (stale!)**

Взглянув на дерево (7), понимаем, какое из правил анализатор рекурсивного спуска должен применить в первую очередь (7c). Почему не имеет смысла применять (7a) или (7b)? Ни одно из этих правил не позволяет получить последовательность слов, где первое John. То есть понятно, что успешный анализ **John saw Mary** возможен, если ***NP*** соответствует последовательность **John *α*.** Если говорить обобщенно, категория ***B*** является левым элементом (left-corner) дерева с корнем ***A*** если $A \Rightarrow^* B\,\alpha$.

Анализатор left-cornerparser – это анализатор сверху-вниз с фильтрованием снизу-вверх. В отличие от анализатора рекурсивного спуска, он не имеет проблем с леворекурсивными правилами. Перед началом работы анализатор предварительно обрабатывает контекстно-свободную грамматику и строит таблицу, где каждая строка содержит две ячейки, в первой хранятся символы из левых частей правил, а во второй – набор возможных левых элементов в правилах grammar1. В табл. 1.3 показан результат такой предварительной обработки грамматики grammar 2.

Таблица 1.3

Left-Corners in grammar 2

Category	Left-Corners (pre-terminals)
S	NP
NP	Det, PropN
VP	V
PP	P

Каждый раз, когда анализатор обрабатывает правило, осуществляется проверка, соответствует ли следующее входное слово хотя бы одной из pre-terminal категорий из табл. 1.2.

Для более глубокого понимания автоматизированного синтаксического и семантического анализа англоязычных текстов пошагово рассмотрим словообразование в английском языке.

12. ЛЕКСИКО-СИНТАКСИЧЕСКИЕ ОСОБЕННОСТИ АНГЛИЙСКОГО ЯЗЫКА

12.1. Словообразование в английском языке

По своей структуре слова в английском языке делятся на:
1) простые;
2) производные;
3) сложные;
4) составные.

12.1.1. Простые слова

К простым словам относятся слова, не имеющие в своем составе ни префиксов, ни суффиксов: chair – стул, red – красный, one – один, to speak – говорить и т. д.

Простые слова могут образовываться от других простых слов следующими способами:

1. Без всякого изменения в произношении и написании слова. Вопрос о том, какой частью речи является такое слово, разрешается на основании его формальных и синтаксических признаков. Такое совпадение слов по форме особенно часто встречается у *существительных* и *глаголов*.

Существительные	**Глаголы**
Answer – ответ	*To answer* – отвечать
Change – изменение	*To change* – менять
Order – приказ	*To order* – приказывать
Work – работа	*To work* – работать

Совпадение форм встречается также у *прилагательных* и *глаголов:*

Прилагательные	**Глаголы**
Clean – чистый	*To clean* – чистить
Empty – пустой	*To empty* – опустошать
Free – свободный	*To free* – освобождать

В некоторых случаях совпадение форм встречается у нескольких частей речи. Так, например, слово light может быть существительным со значением *свет*, прилагательным – *светлый* и глаголом – *зажигать*, *освещать*.

2. Путём изменения ударения (без изменения в произношении и написании слова), причём существительные имеют ударение на первом слоге, а соответствующие им глаголы – на втором (*в современном английском языке новые слова таким способом не образуются*).

Существительные	**Глаголы**
Increase – увеличение	*To increase* – увеличивать(ся)
Export – экспорт	*To export* – экспортировать
Import – импорт	*To import* – импортировать

3. Путём чередования последнего согласного звука, который является глухим в существительном и звонким в соответствующем ему глаголе. При этом в ряде случаев чередование последнего согласного звука сопровождается чередованием корневого гласного звука и изменением написания слова (в современном английском языке новые слова таким способом не образуются).

Существительные	**Глаголы**
Use [ju:s] – потребление	*To use* [ju:z] – употреблять
Advice [ad'vais] – совет	*To advise* [ad'vaiz] – советовать
Life [1aif] – жизнь	*To live* [1 i v] – жить
Choice [tʃois] – выбор	*To choose* [tʃu:z] – выбирать
Loss [1os] – потеря	*To lose* [1oz] – терять

Некоторые существительные и глаголы различаются только чередованием гласных звуков при соответствующем изменении написания слова:

Существительные	**Глаголы**
Food – пища	*To feed* – питать(ся)
Shot – выстрел	*To shoot* – стрелять
Song – песня	*To sing* – петь

12.1.2. Производные слова

К производным словам относятся слова, в состав которых входят *префиксы* или *суффиксы*, или одновременно и те, и другие.

Префикс стоит в начале слова и изменяет значение слова, но, как правило, не меняет его принадлежность к той или иной части речи:

Order (сущ.) – порядок	Disorder (сущ.) – беспорядок
Happy (прил.) – счастливый	Unhappy (прил.) – несчастный
To appear (глаг.) – появляться	*To reappear* (глаг.) – вновь появляться

Лишь в редких случаях присоединение префикса меняет принадлежность слова к той или иной части речи:

Circle (сущ.) – круг	*To encircle* (глаг.) – окружать

Суффиксы стоят в конце слова и в большинстве случаев служат для образования одной части речи от другой. Таким образом, суффикс показывает, к какой части речи относится данное слово:

Use (сущ.) – польза	*Useful* (прил.) – полезный
Strength (сущ.) – сила	*To strengthen* (глаг.) – усиливать(ся)
Happy (прил.) – счастливый	*Happiness* (сущ.) – счастье
Bad (прил.) – плохой	*Badly* (нареч.) – плохо
To read (глаг.) – читать	*Reader* (сущ.) – читатель

Значительно реже суффиксы образуют новые слова внутри одной и той же части речи:

Friend (сущ.) – друг	*Friendship* (сущ.) – дружба
Child (сущ.) – ребёнок	*Childhood* (сущ.) – детство

Производные слова часто образуются от других производных слов и, таким образом, они могут иметь в своём составе как префикс, так и суффикс (или два суффикса):

Un*happy* – несчастный	***Un****happi****ness*** – несчастье
Dis*honest* – нечестный	***Dis****honestly* – нечестно
*Care****ful*** – тщательный	*Care****fully*** – тщательно
*Use****less*** – бесполезный	*Use****lessness*** – бесполезность

12.1.3. Сложные слова

К сложным словам относятся слова, образованные путём соединения двух слов в одно. Некоторые сложные слова пишутся слитно, а другие – через дефис:

1. Сложные существительные

Bedroom – спальня (bed – кровать + room – комната)
Newspaper – газета (news – новость + paper – бумага)
Schoolboy – школьник (school – школа + boy – мальчик)

Следует выделить сложные существительные, образованные из сочетания герундия и существительного, такие сложные существительные пишутся через дефис:

Dining-room – столовая (a room for dining)
Sleeping-car – спальный вагон (a car for sleeping)

В сложных существительных ударение обычно падает на первое слово, входящее в его состав:

Newspaper, reading-room

Некоторые сложные существительные состоят из двух существительных с предлогом между ними. В этом случае они всегда пишутся через дефис:

Commander-in-Chief – главнокомандующий

2. Сложные прилагательные

Dark-blue – темно-синий (dark – тёмный + blue – синий)

Следует выделить сложные прилагательные, образованные сочетанием прилагательного и существительного с суффиксом **-e** (этот суффикс совпадает по форме с суффиксом **PAST PARTICIPLE**):

Blue-eyed – голубоглазый (blue – голубой + eye – глаз)
Kind-hearted – добросердечный (kind – добрый + heart – сердце)

Сложные прилагательные имеют двойное ударение:

Dark-blue, Blue-eyed

3. Сложные местоимения
Somebody – кто-то, кто-нибудь
Everybody – каждый
Nothing – ничего

12.1.4. Составные слова

Составные слова представляют собой сочетания двух или более слов, которые выражают единое понятие. Составные слова пишутся раздельно без дефиса (обратите внимание, что сложные слова пишутся слитно или через дефис).

1. Составные глаголы

Составные глаголы представляют собой сочетание глаголов с наречиями, выражающими одно понятие. В некоторых случаях значение сочетания глагола с наречием вытекает из значений слов, входящих в состав сочетания:

To go down – опускаться (to go – идти + down – вниз)
To go up – подниматься (to go – идти + up – вверх)
To come in – входить (to come – приходить + in – внутрь)

В некоторых же случаях значение сочетания глагола с наречием не соответствует значениям слов, входящих в его состав:

To ring up – звонить по телефону (to ring – звонить + up – вверх)
To give up – бросать, прекращать (to give – давать + up – вверх)

В английском языке имеется большое количество составных глаголов.

2. Составные числительные

Four hundred and fifty – четыреста пятьдесят
Two thousand three hundred and twenty – две тысячи триста двадцать

3. Составные наречия

By then – к тому времени
Since then – с тех пор
So far – до сих пор, пока
Forever – навечно

4. Составные союзы

As long as – пока
As soon as – как только
As if, as though – как если бы, как будто
In spite of the fact – несмотря на то, что [5; 6].

12.2. Аффиксация как один из способов образования слов в английском языке

12.2.1. Основные префиксы и их значения

Префикс	К какой части речи относится	Основное значение	Примеры, перевод
anti-	сущ., прил.	анти-, противо-	antisocial – антиобщественный
be-	глаг., сущ., прил.	Изменяет часть речи	belittle – уменьшать
co-	глаг., сущ.	со-, общность действия	cooperate – сотрудничать coauthor – соавтор
counter-	глаг.	контр-, противо-	counteraction – противодействие counter-irritant – противовоспалительное (средство)
de-	глаг., сущ.	де-, отрицательное значение	deformation – деформация deoxidize – вытеснять кислород,
dis-	глаг, сущ., прил.	раз-, рас-, дез-, обез-, отрицательное значение	disability – нетрудоспособность disadvantage – невыгода, недостаток
en- (em-)	сущ., прил.	Изменяет часть речи	enlarge – увеличивать
im-, in-, ir-, il-	прил., сущ.	не-	illogical – нелогичный invisible – невидимый

Префикс	К какой части речи относится	Основное значение	Примеры, перевод
im-, in-, ir-, il-	глаг., сущ.	в-, при-, внутри-	inborn – врожденный, прирожденный inbred – рожденный от родителей, состоящих в кровном родстве inflow – приток
inter-	глаг., прил.	между-, взаимо-	interaction – взаимодействие
mal-	глаг., прил., сущ.	Плохо, недостаточно неправильно	malpractice – преступная небрежность врача в лечении больного malnutrition – недоедание, недостаточное или неправильное питание
mis-	глаг., сущ.	Отрицательное значение	misunderstand – неправильно понять
out-	глаг.	пере-, превосходить ч.-либо. Изменяет часть речи	outweigh – перевешивать, превосходить в весе
over-	глаг., прил.	пере-, сверх-	overpeopled – перенаселённый
re-	глаг.	Снова, вновь, пере-	rewrite – переписать
trans-	глаг., прил.	пере-, транс	transplant – пересадить
ultra-	сущ., прил.	Превосходящее обычное, ультра-	ultrasonic – ультразвуковой
under-	глаг., прил.	недо- /ниже нормы	underestimate – недооценивать

12.2.2. Основные суффиксы существительных

Суффикс	К какой части речи добавляется	Основное значение	Примеры, перевод
-age	глаг., сущ., прил.	Действие, состояние	shortage – нехватка
-al	глаг.	Действие	removal – удаление
-ance -ence	глаг., прил.	Действие, состояние	silence – молчание, тишина
-ant -ent	глаг.	Принадлежность к профессии	assistant – ассистент
-dom	глаг., прил.	Состояние, качество	freedom – свобода
-er -or	глаг.	Действующее лицо; механизм, производящий действие	reader – читатель
-hood	сущ.	Состояние	childhood – детство
-ian	сущ.	Профессия	technician – техник
-ics	сущ.	Название науки	physics – физика
-ing	глаг.	Процесс, действие, состояние	reading – чтение
-ty	прил.	Качество или состояние	activity – активность
-ment	глаг.	Результат действия	equipment – оборудование
-ness	прил.	Качество	sadness – печаль, readiness – готовность, forgetfulness – забывчивость

12.2.3. Основные суффиксы прилагательных

Суффикс	К какой части речи добавляется	Основное значение	Примеры, перевод
-able	глаг. сущ.	Способный ч.-либо сделать или пригодный для ч.-либо	countable – поддающийся счёту
-al	сущ.	Наличие качества	electrical – электрический
-ant, -ent	глаг.	Наличие качества, свойства	different – различный
-ar	сущ., глаг.	–	revolutionary – революционный
-ful	сущ.	Наличие качества	painful – болезненный
-ish	сущ.	Наличие признака в слабой степени	whitish – беловатый
-ive	глаг.	Наличие качества, свойства	creative – созидательный
-less	сущ.	Отсутствие качества	painless – безболезненный
-ous	сущ.	Наличие качества, свойства	dangerous – опасный

12.2.4. Основные суффиксы наречий

Суффикс	К какой части речи добавляется	Основное значение	Примеры, перевод
-ly	прил.	Изменяет часть речи	quickly – быстро
-ward	сущ., прил.	Направление	backward(s) – назад

12.2.5. Основные суффиксы глаголов

Суффикс	Примеры, перевод
-ate	activate – активизировать
-en	soften – смягчить
-fy	intensify – усиливать
-ize	summarize – суммировать

Для дополнительной реализации задачи, поставленной нами в данном пособии, рассмотрим порядок слов в повествовательном предложении.

12.3. Повествовательные предложения (Declarative Sentences)

Повествовательные предложения служат для того, чтобы сообщить что-то собеседнику или читателю. Они содержат утверждение какого-либо факта (повествовательные утвердительные предложения) или отрицание какого-либо факта (повествовательные отрицательные предложения). Повествовательные предложения произносятся с понижением голоса на последнем ударном слоге. Английские повествовательные предложения имеют четкий порядок слов, т. е. каждый член предложения имеет свое определенное место, поскольку в английском языке отсутствуют падежные окончания и в основном место слова определяет его функцию в предложении. Следующий порядок слов является обычным для английского повествовательного предложения: 1) подлежащее, 2) сказуемое, 3) дополнение, 4) обстоятельство:

I	received	a letter	yesterday
(подлежащее)	*(сказуемое)*	*(дополнение)*	*(обстоятельство)*
Я	получил	письмо	вчера

Пятый член предложения – определение – не имеет постоянного места в предложении и может стоять при любом члене предложения, выраженном существительным:

The capital **of France** is Paris.	Столица Франции – Париж.
I have bought *an* **interesting** *book*.	Я купил интересную книгу.
They live in *a* **new** *house*.	Они живут в новом доме.

12.3.1. Схема порядка слов в традиционном повествовательном предложении

Подлежащее	Сказуемое	Дополнение			Обстоятельство		
		беспредложное косвенное	прямое	предложное косвенное	образа действия	места	времени
I He She We I	bought sent sent received met	his father	a tablet an e-mail it a fax him	to his father from Paul	by chance	at the theatre	a few days ago

Повествовательные предложения могут быть сформулированы еще и следующим образом:

– с помощью формального подлежащего ***it***;

– с помощью оборотов ***there is, there are***;

– с помощью неопределенных подлежащих ***one*** и ***they***.

12.3.2. Особенности порядка слов в повествовательном предложении с формальным подлежащим *it*

В английских безличных предложениях употребляется формальное подлежащее, выраженное местоимением ***it***, поскольку в английском языке подлежащее является обязательным элементом предложения.

Местоимение ***it*** употребляется в качестве *формального подлежащего* в следующих безличных предложениях:

1. При сообщениях о *явлениях природы*:

It is spring.	Весна.
It is warm.	Тепло.
It is getting cold.	Становится холодно.
It was a frosty sunny day	Был морозный солнечный день.

2. При глаголах, обозначающих *состояние погоды:*

It was freezing.	Морозило.
It has been snowing since 5 a.m.	Снег идет с пяти часов утра.
It often rains in October.	В октябре часто идет дождь.

3. При обозначениях *времени и расстояния*:

It is 7 o'clock.	Семь часов.
It is an early morning.	Раннее утро.
It is midnight.	Полночь.
It is about 1 kilometer from our university to the central library.	От нашего университета до центральной библиотеки приблизительно один километр.
It is not far to the rail-way station.	До вокзала недалеко.

Местоимение ***it*** в функции формального подлежащего употребляется с некоторыми глаголами *в страдательном* залоге. Такие страдательные обороты соответствуют в русском языке неопределенно-личным оборотам:

It is said...	Говорят…
It is supposed...	Полагают...
It is expected...	Ожидают...

Местоимение ***it*** в функции формального подлежащего употребляется также и при наличии подлежащего предложения, выраженного инфинитивом, герундием или придаточным предложением и стоящего после сказуемого:

It was difficult to find the suitable steamer	Было трудно найти подходящий пароход.
It is useless to tell him about it.	Бесполезно говорить ему об этом.
It was clear that he would not come.	Было ясно, что он не придет.

12.3.3. Случаи употребления предложений со сказуемым, выраженным оборотом *there is/are*

Для выражения наличия или существования в определенном месте или отрезке времени какого-нибудь лица или предмета, факта или явления, ещё неизвестного собеседнику или читателю, употребляется особый тип простого сказуемого, выраженного оборотом **there is (are)** со значением *имеется, находится, есть, существует*. Оборот **there is (are)** стоит в начале предложения; за ним стоит подлежащее, за которым следует обстоятельство места или времени. Соответствующие русские предложения начинаются, как правило, с обстоятельства места или времени.

There is a telephone in that room.	В той комнате есть (имеется) телефон.
There are many apple trees in the garden.	В саду (имеется) много яблонь.
There was a meeting at the Institute yesterday.	Вчера в институте было собрание.

There в обороте **there is (are**) не имеет самостоятельного значения и составляет одно целое с **is (are**). Если по смыслу предложения требуется наличие наречия **there** со значением *там,* то **there** повторяется в конце предложения: **There are** many children **there**. *Там много детей.*

После оборота **there is** исчисляемые существительные в единственном числе употребляются с классифицирующим артиклем, а

исчисляемые существительные во множественном числе и неисчисляемые существительные – с местоимением **some (any):**

There is **a lamp** on the table.	На столе (имеется, есть) лампа.
There are **some lamps** on the table.	На столе (имеется, есть) несколько ламп.
There is **some cheese** and **some butter** on the plate.	На тарелке (имеется, есть) сыр и масло.

Глагол **to be** в обороте **there is** может употребляться в разных формах времени: **there is, there are** *есть, находится(-ятся), имеется(-ются)*,* **there was, there were** *был (были), находился (-ись), имелся(-ись),* **there will be** *будет (будут), будет (будут) находиться, будет (будут) иметься* и т. п.:

There are very many French books in this library.	В этой библиотеке (**имеется)** очень много французских книг.
There was a meeting at the club yesterday.	Вчера в клубе **было** собрание.
There will be a good wheat crop in the Ukraine this year.	В этом году в Украине **будет** хороший урожай пшеницы.

В вопросительных предложениях глагол **to be** ставится перед **there.** Если глагол **to be** употреблен в сложной форме времени, то перед **there** ставится вспомогательный глагол:

Is there a telephone in your room?	Есть ли телефон в вашей комнате?
Was there a meeting at the Institute yesterday?	Было ли собрание вчера в институте?
Will there be many people there?	Будет ли там много народу?

Краткие ответы на вопрос с оборотом **there is** состоят из **yes** или **no** и оборота **there is (are)** в утвердительной или отрицательной форме:

Is there a telephone in your room?	Yes, **there is.** No, **there isn't.**
Was there a meeting at the Institute yesterday?	Yes, **there was.** No, **there wasn't.**

Если глагол **to be** в вопросе употреблен в сложной форме времени, то в кратком ответе после **there** ставится только вспомогательный глагол:

Will there be many people there?	Yes, there will. No, there **won't.**

*T**here is** и **there are** иногда на русский язык не переводятся.

Следует также обратить внимание, что обороты **there isn't (aren't)** соответствуют в русском языке слову ***нет*** в функции сказуемого, заменяющему отсутствующее в русском языке настоящее время от **глагола *быть* с отрицанием**.

12.3.4. Повествовательные предложения с неопределенным подлежащим *one, they, we, you*

В том случае, когда в английском языке действующее лицо мыслится неопределенно или обобщенно, в функции подлежащего употребляется местоимение ***one*** (каждый, всякий человек, люди) и местоимение ***they*** со значением ***люди*** (кроме говорящего).

Предложения с неопределенным подлежащим ***one*** или ***they*** переводятся на русский язык неопределенно-личным предложением:

One should be careful when crossing the road.	**Нужно быть осторожным** при переходе через улицу.
They say the wheat crop in Ukraine will be fine this year.	**Говорят**, что в этом году будет прекрасный урожай пшеницы в Украине.

В качестве подлежащего с неопределенно-личным значением употребляются личные местоимения ***we*** и ***you***, как и соответствующие им местоимения ***мы*** и ***вы*** в русском языке:

We must be very attentive to old people.	**Мы** должны быть очень внимательны к старым людям.
You never know where to find that man.	**Вы** никогда не знаете (никогда не знаешь), где можно найти этого человека.

Подробнее см. [5; 6].

Часть вторая

АВТОМАТИЗИРОВАННЫЙ СЕМАНТИЧЕСКИЙ АНАЛИЗ

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ СЕМАНТИЧЕСКОГО АНАЛИЗА

Семантический (смысловой) анализ текста – одна из ключевых проблем как теории создания систем искусственного интеллекта, относящаяся к обработке естественного языка (**Natural Language Processing, NLP**), так и компьютерной лингвистики. Результаты семантического анализа могут применяться для решения задач в таких областях, как, например, психиатрия (для диагностирования больных), политология (прогнозирование результатов выборов), торговля (анализ «востребованности» тех или иных товаров на основе комментариев к данному товару), филология (анализ авторских текстов), поисковые системы, системы автоматического перевода и т. д. Но прежде чем использовать возможности семантического анализа в этих сферах, необходимо решить задачи самого семантического анализа (рис. 2.1).

Рис. 2.1. Задачи семантического анализа

Несмотря на востребованность практически во всех областях жизни человека, семантический анализ является одной из сложнейших математических задач. Вся сложность заключается в том, чтобы «научить» компьютер правильно трактовать образы, которые автор текста пытается передать своим читателям/слушателям. Способность «распознавать» образы считается основным свойством человеческих существ, как, впрочем, и других живых организмов. Образ представляет собой описание объекта. В каждое мгновение нашего бодрствования мы совершаем акты распознавания. Мы опознаем окружающие нас объекты и в соответствии с этим перемещаемся и совершаем определенные действия. Мы можем заметить в толпе друга и понять, что он говорит, можем узнать голос знакомого, можем отличить улыбку от злобной гримасы. Человеческое существо представляет собой очень сложную информационную систему – в какой-то степени это определяется чрезвычайно развитыми у человека способностями распознавать образы.

В настоящее время существуют различные подходы к созданию систем для **автоматизированного семантического анализа**. Среди огромного числа алгоритмов, которые используются для поиска и анализа информации, особое место занимают те из них, целью которых является обнаружение скрытых закономерностей или неочевидных зависимостей.

Используя их, мы можем сказать, например, что два текста похожи, даже если эта похожесть выражена косвенно. Или, например, «лыжи» и «автомобиль» по отдельности относятся к разным категориям, но, будучи использованы вместе, могут быть интерпретированы в таких категориях, как «спорт» и «отдых». Один из самых перспективных методов, который применяется для рекомендательных систем (коллаборативная фильтрация), информационного семантического поиска, разделения текстов по тематикам без обучения и многих других, и будет использован при обучении по дисциплине «**Автоматизированный семантический анализ**». Метод этот называется латентно-семантическим анализом (LSA – **Latent semantic analysis**). Следует также обратить внимание на слово «автоматизированный» – человеко-машинная система для сбора, хранения, накопления, поиска, передачи, обработки информации с использованием вычислительной техни-

ки. Изучать дисциплину «**Автоматизированный семантический анализ**» без её реализации на компьютере для современного лингвиста уже не актуально. Поэтому основное внимание в этом курсе будет уделено реализации LSA на языке Python.

Рассмотрим более подробно, что это за метод и как он работает. Уже из названия можно сделать вывод о том, что он должен делать, а именно находить скрытые смысловые взаимосвязи между объектами (будь то слова в тексте или товары в магазине). Для текстов на естественных языках такой скрытой закономерностью может быть, например, **наличие определенного набора слов в определенной теме**. Представим себе такую задачу: у нас есть коллекция документов, и мы хотим научиться отвечать на вопрос: «Два документа близки по тематике или нет?». Вывод о схожести можно сделать, основываясь на том, какие слова и в каких пропорциях входят в каждый из документов.

Чтобы подготовить данные для этой задачи, используют подход, который называется «мешок слов». Его суть состоит в том, что для нас не важен порядок слов в документе, в каких морфологических формах они представлены, а важно только количество вхождений конкретных слов.

Предположим, что каждую тему можно охарактеризовать определенным набором слов и частотой их появления. Если в тексте конкретный набор слов употребляется с определенными частотами, то текст принадлежит к определенной теме. Основываясь только на этой информации, можно построить таблицу «Слово – документ», где строки соответствуют словам (а точнее, их леммам), а столбцы – документам. В каждой ячейке хранится 1, если слово есть в документе, и 0 – если нет. Хотя такой вариант и самый простой, но не самый лучший.

Вместо 0 и 1 можно использовать, например, частоту слова в документе или TF-IDF слова. Такой способ представления текстов в виде таблицы (или матрицы) называется векторной моделью текста. Теперь, для того чтобы сравнить два документа, нужно определить меру схожести двух столбцов таблицы. Сделать это можно по-разному:

– скалярное произведение векторов:

$$(V = b_1 \times b_2 = x_1 + x_2 + y_1 \times y_2 \ldots\ldots\ldots\ldots xn \times xn_1 + yn \times yn_1)$$

– косинусное расстояние: $CosQ = \frac{|A \times B|}{|A| \times |B|}$, ***A, B*** – векторы;

– евклидово расстояние;
– манхэттенское расстояние.

Чтобы лучше понять все вышесказанное, представим это графически на простом примере двух небольших текстов (рис. 2.2). Один текст о письменности, другой – о неопределенности Гейзенберга. Стоп-слова удалены, а остальные приведены к основной форме (без окончаний).

Каждая точка на графике – слово. На осях отложено, сколько раз слово встретилось в каждом документе. Т. е. если слово встретилось в тексте о неопределенности 3 раза, а в тексте о письменности 2 раза, то на рис. 2.2 это слово изобразим точкой с координатами (3, 2).

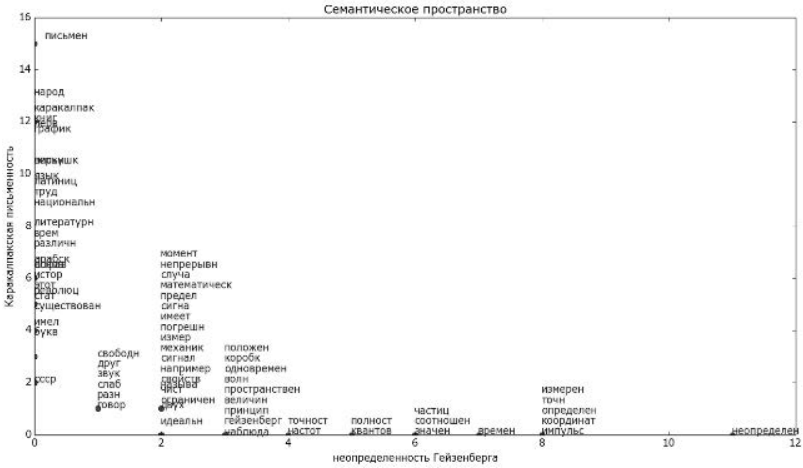

Рис. 2.2. Распределение слов по частоте употребления в каждом документе (документы различной тематики)

Видно, что в этом примере (рис. 2.2) некоторые слова встречались и в одном, и в другом тексте документов приблизительно одинаково часто («свобода», «друг», «звук» и т. д.). Такие

слова не дают возможности отличить тексты один от другого и в принципе сравнимы со стоп-словами. Но есть слова, которые характерны только для одного текста. Имея такое представление текста, мы можем определять близость каждого слова к теме (как косинус угла между вектором с началом в (0; 0) и концом в точке слова и осью, соответствующей документу). Если же такого слова в коллекции нет, то о нем мы ничего не можем сказать. Для сравнения документов можно подсчитать сумму векторов-слов, которые в них входят, и, опять же, оценить расстояние между ними.

В рассмотренном примере слова распределились хорошо, так как тематики существенно разные. А если тематики схожи, то может получиться такая картина (рис. 2.3).

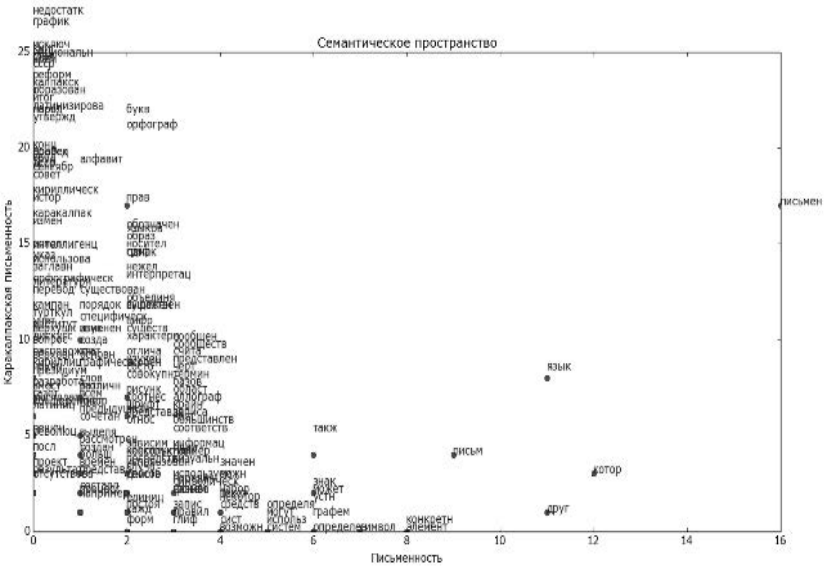

Рис. 2.3. Распределение слов по частоте употребления в каждом документе (документы близких тематик)

По сравнению с предыдущим графиком (рис. 2.2) видно, что документы похожи, и, кроме того, есть слова, которые характеризуют общую тематику для обоих текстов (например, «язык»

и «письмен»). Такие слова можно назвать ключевыми для данной темы. Имея такое представление текстов, мы теоретически можем сгруппировать документы по близости их содержимого и построить тематическое разбиение коллекции текстов. **Может оказаться, что каждый документ – это отдельная тема. Также можно искать документы по запросу, при этом могут находиться документы, которые не содержат слов из запроса, но близки ему по теме.**

В реальности оказывается, что документов и слов очень много (гораздо больше, чем тем), и возникают следующие проблемы:

– **размерность** (вычисление близости между векторами становится медленной процедурой);

– **зашумленность** (например, посторонние небольшие вставки текста не должны влиять на тематику);

– **разряженность** (большинство ячеек в таблице будут нулевыми).

Вместо таблицы «Слово – документ» следует использовать «Слово – тема» и «Тема – документ». Решение именно такой задачи предлагает LSA.

Выводы. 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.

2. ПРОЦЕДУРА ПРОВЕДЕНИЯ LSA

1. Из анализируемых документов исключаются стоп-слова. 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 приведен в статье [20].

stopwords=['-', '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', 'ни будь', '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']

Стоп-слова для других языков могут быть взяты из литературы или определены с учётом специфики текста.

2. Из анализируемых документов необходимо отфильтровать цифры, отдельные буквы и знаки препинания.

3. Исключить слова, встречающиеся один раз. Это не влияет на конечный результат, но сильно упрощает математические вычисления.

4. Со всеми словами из документов должна быть проведена операция стемминга – получение основы слова. Для этого используется алгоритм Портера [21], основным преимуществом которого является отсутствие словарей. Это преимущество особенно полезно при автоматизации семантического анализа.

5. Составляется частотная матрица индексируемых слов. В этой матрице строки соответствуют индексированным словам, а столбцы – документам. В каждой ячейке матрицы должно быть указано, сколько раз слово встречается в соответствующем документе.

6. Проводится проверка, встречается ли хотя бы одно из отобранных слов в каждом документе. Если нет, нужно исключить из рассмотрения эти документы и провести отбор слов по приведенной методике с начала, пока не будет выполнено условие: каждое слово встречается в документе.

7. Полученную частотную матрицу следует нормализовать. Стандартный способ нормализации матрицы – TF-IDF [23].

TF (*term frequency* – частота слова) – отношение числа вхождения некоторого слова к общему количеству слов документа. Таким образом, оценивается важность слова t_i в пределах отдельного документа:

$$tf(t,d) = \frac{n_i}{\sum_k n_k}, \qquad (2.1)$$

где n_i – число вхождений слова в документ, а в знаменателе – общее число слов в данном документе.

IDF (*inverse document frequency* – обратная частота документа) – инверсия частоты, с которой некоторое слово встречается в документах коллекции. Основоположником данной концепции является Карен Спарк Джонс [24]. Учёт IDF уменьшает вес широкоупотребительных слов. Для каждого уникального слова в пределах конкретной коллекции документов существует только одно значение IDF:

$$idf(t,D) = \log \frac{|D|}{|(D_i \supset t_i)|}, \qquad (2.2)$$

где: |D| – количество документов в корпусе; $|(d_i \supset t_i)|$, – количество документов, в которых встречается t_i (когда $n_i \neq 0$).

Выбор основания логарифма в формуле не имеет значения, поскольку изменение основания приводит к изменению веса каждого слова на постоянный множитель, что не влияет на соотношение весов.

Таким образом, мера **TF-IDF** является произведением двух сомножителей:

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D) \qquad (2.3)$$

Большой вес в **TF-IDF** получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

8. В рамках следующего шага проводится сингулярное разложение полученной матрицы. Сингулярное разложение [24] – это математическая операция, раскладывающая матрицу на три составляющие. Исходная матрица *M* представляется в виде:

$$M = U \times W \times Vt, \qquad (2.4)$$

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.

9. Отбрасываются последние столбцы матрицы *U* и последние строки матрицы *V*^t, оставляются только первые 2. Это, соответственно, координаты *X, Y* каждого слова для матрицы U и координаты *X, Y* для каждого документа в матрице *V*^*t*. Разложение такого вида называют двумерным сингулярным разложением.

10. Отмечаются на графике семантического пространства точки, соответствующие отдельным текстам и словам. Определяется местоположение слов и документов в семантическом пространстве, эта информация используется для определения тематики документа.

ПРАКТИЧЕСКИЕ РАБОТЫ

№ 1. Установка Python 3.4 и подключение необходимых модулей

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки для настройки среды программирования **Python.**

Ход выполнения работы (ПК должен быть подключён к Internet, а при выполнении работы в компьютерном классе п. 1–9 изучить, но не выполнять):

1. Скачайте с сайта https://www.python.org/getit/ и установите последнюю версию дистрибутива **Python-3.4.1.msi.**

2. Скачайте с сайта [25] и установите загрузчик модулей, например **pip-Win_1.6.exe**.

3. При помощи загрузчика установите модуль **nltk**, как показано на рис. 1.

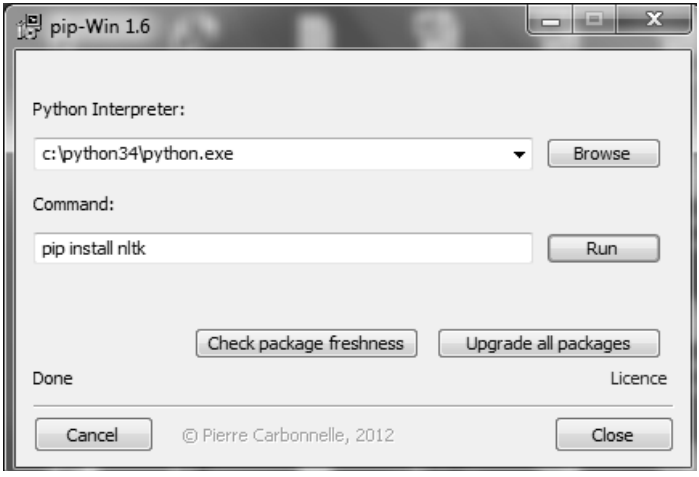

Рис. 1. Установка модуля

Для загрузки модуля впишите после pip install имя модуля, например **nltk** (без версии программа находит версию сама). Если в строке **Python Interpreter** правильно установлен путь **c:\python34\python.exe**, снизу нажмите кнопку **Run.** После появления второго окна **C:\Windows\system32\cmd.exe** дождитесь результатов загрузки и нажмите любую клавишу после соответствующей команды на русском языке.

4. Проверить факт установки любого модуля можно при помощи кнопки Check package freshness (рис. 1) в появившемся окне (рис. 2).

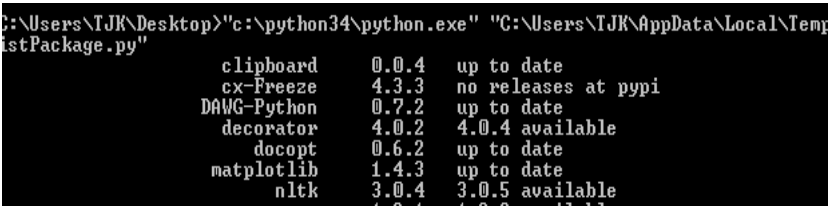

Рис. 2. Окно для просмотра установленных модулей Python

5. После успешной установки, например up to date или available (рис. 2), нужно запустить сам Python – кнопка Пуск – Все программы – Python 3.4-IDLE(Python GUI - 32 bit). В строке приглашения набрать: **import nltk** (рис. 3).

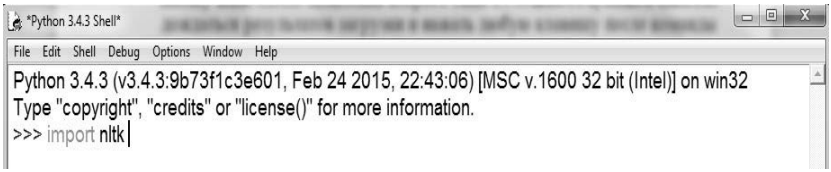

Рис. 3. Проверка работы модуля nltk

6. **Внимание!** После нажатия клавиши Enter программа Python запросит для правильной работы nltk дополнительные

модули в зависимости от версии Python. Все эти модули нужно загрузить, как описано в п. 3–4, с учётом комментариев к рис.1, 2.

7. В случае если pip-Win_1.6.exe не находит нужного модуля, его нужно скачать с сайта [27] (осуществить поиск можно и непосредственно на веб-странице, набирая только http://www.lfd.uci.edu/~gohlke/pythonlibs). После появления дистрибутива модуля на Вашем ПК его можно загрузить через pip-Win_1.6.exe, введя полное его название, как в дистрибутиве. Однако для пользования данной полной библиотекой нужно сначала скачать для распаковки архивов whl соответствующий модуль wheel (рис. 4).

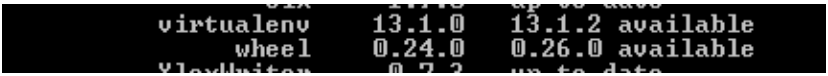

Рис. 4. Архиватор wheel в среде Python

8. После установки всех модулей происходит загрузка nltk, как показано на рис. 5.

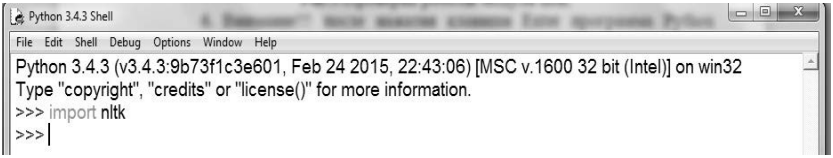

Рис. 5. Результат успешной загрузки nltk в среде Python

Среда **Python** может запрашивать сокращённое название модуля, например **yaml**, а искать нужно **PyYAML**, или **dateuntil**, а искать нужно **python-dateuntil**.

9. После успешного импортирования **import nltk** нужно набрать в строке приглашения nltk.download() и подключить все Интернет-ресурсы Python (рис. 6–9).

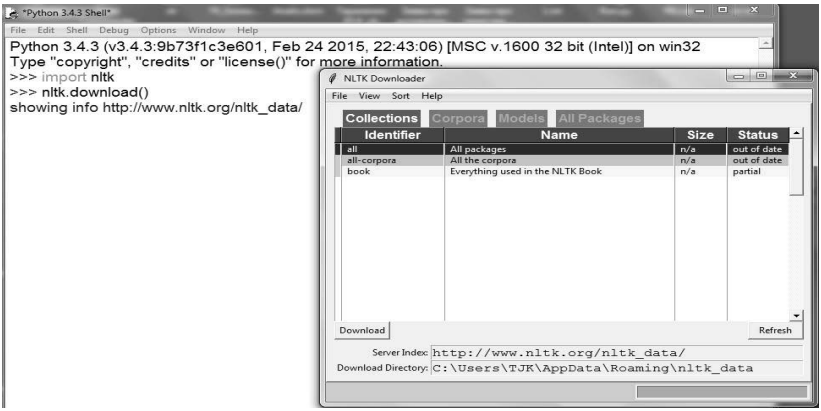

Рис. 6. Подключение коллекций при помощи выделения строки и нажатия кнопки Download

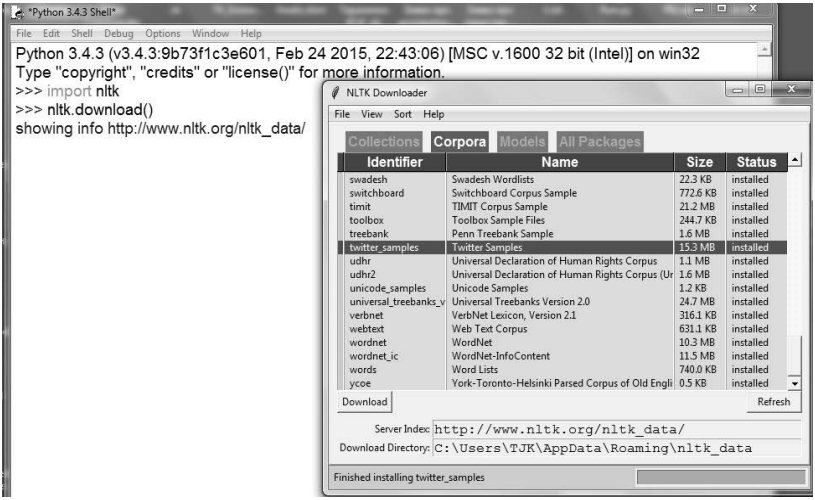

Рис. 7. Подключение корпусов при помощи выделения строки и нажатия кнопки Download (необходимо воспользоваться прокруткой)

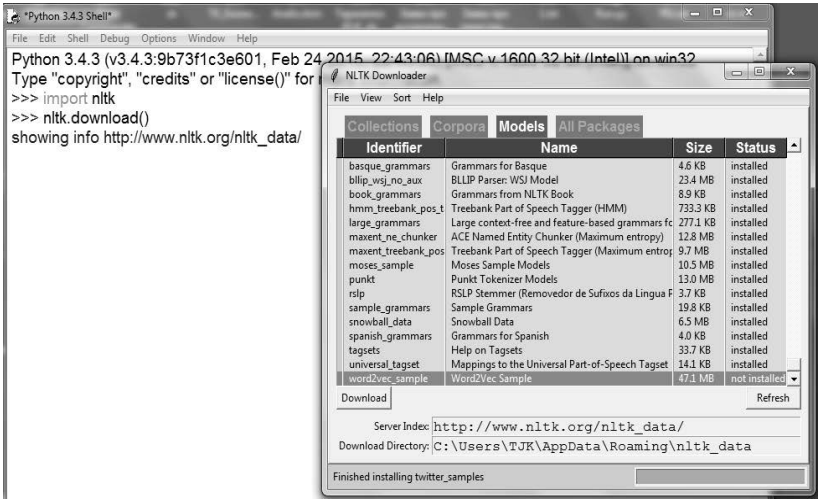

Рис. 8. Подключение моделей при помощи выделения строки и нажатия кнопки Download (необходимо воспользоваться прокруткой)

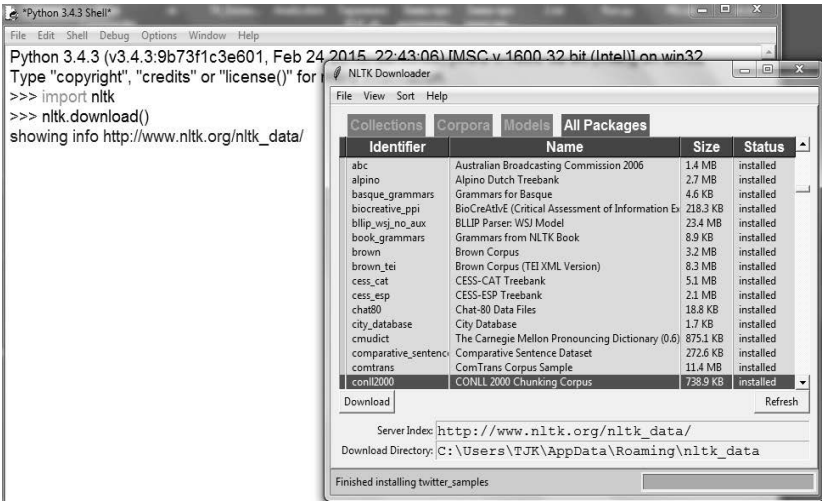

Рис. 9. Подключение пакетов при помощи выделения строки и нажатия кнопки Download (необходимо воспользоваться прокруткой)

10. Подключите модули **numpy**, **scipy, matplotlib**, как описано в п. 1–4.

11. Создание специального модуля settings.py для автоматизации семантического анализа. В этот модуль нужно загрузить исследуемые документы в списке (list), например под именем переменной docs. Для проведения стемминга необходимо в переменную, например stem='english', установить любой из следующих языков: 'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian', 'italian', 'norwegian', 'porter', 'portuguese', 'romanian', 'russian', 'spanish', 'swedish'. Необходимо также предусмотреть список (list), например в переменной stopwords. Остаётся только ввести переменную wordd для слова, от которой будет отсчитываться условное расстояние до других слов и документов в семантическом пространстве.

Пример упрощённого листинга для **settings.py:**

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
docs =[
 "Human machine interface for ABC computer applications",
 "A survey of user opinion of computer system response time",
"The EPS user interface management system",
"System and human system engineering testing of EPS",
 "Relation of user perceived response time to error measurement",
"The generation of random, binary, ordered trees",
"The intersection graph of paths in trees",
"Graph minors IV: Widths of trees and well-quasi-ordering",
"Graph minors: A survey"
]
stopwords=['The', 'IV', 'in', 'for', 'to', 'of', 'and', 'ABC']
wordd='computer'
stem='english'
#'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian',
#'italian', 'russian', 'norwegian', 'porter', 'portuguese', 'romanian', 'russian', #'spanish', 'swedish'.
```

12. Проведите исследование применения функции **SnowballStemmer (stem) nltk** Python для решения задач процеду-

ры проведения LSA. Подробно с этим можно ознакомиться по ссылке [27].

Пример стемминга:

```
>>> import nltk
>>> from nltk.stem import SnowballStemmer
>>> stem='russian'
>>> stemmer = SnowballStemmer(stem)
>>> w='пенсией'
>>> st=stemmer.stem(w)
>>> st
'пенс'
```

Вопросы к практической работе № 1

1. Перечислите, какие этапы LSA, исключая установку модулей, реализованы в данной работе на Python.

2. Какие особенности загрузки модулей по сообщениям Python об их отсутствии?

3. Что нужно делать, когда **pip-Win_1.6.exe** не находит нужный модуль?

4. Как изменить settings.py, чтобы язык документов для операции стемминга можно было изменять из основного модуля?

5. Что можно установить через nlt.download()?

6. Чем отличается операция стемминга в Python от других способов стемминга?

Задания к практической работе № 1

1. Создать и заполнить два разных модуля settings.py для двух изучаемых языков: в каждом по 10 документов, в каждом документе НЕ МЕНЕЕ 15 слов, документы должны содержать НЕ МЕНЕЕ 3 групп с общей тематикой в каждой.

2. Ответы на вопросы и листинги модулей поместить в отчёт по практической работе № 1, оформленный в документе Word по следующей схеме **для всех следующих практических работ**:

2.1. Отчёт по практической работе №__ Название __Студента (ФИО)____ группы___ дата сдачи ___отметка преподавателя _____

2.2 Краткие теоретические сведения по теме первого радела этого методического пособия.

2.3. Ответы на вопросы по практической работе.

2.4. Результаты выполнения задания – листинги, снимки экрана и т. п.

2.5. Выводы: «В результате выполнения работы получил(а) знания _____навыки_____»

№ 2. Разработка интерфейса автоматизированного семантического анализа с использованием библиотеки модулей tkinter, встроенной в Python

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки по разработке интерфейса в среде программирования **Python.**

Ход выполнения работы:

1. Повторите уже пройдённый Вами материал по созданию графического интерфейса при помощи ***tkinter*** [28].

2. Создайте форму с тремя полями и меню в модуле menu.py с помощью кода:

```
 #!/usr/bin/env python
# -*- coding: utf-8 -*-
from tkinter.filedialog import *
from tkinter.messagebox import *
# Место для программного кода
tk= Tk()
main_menu = Menu(tk)
tk.config(menu=main_menu)
file_menu = Menu(main_menu)
main_menu.add_cascade(label="LSA", menu=file_menu)
txt = Text(tk, width=72,height=10,font="Arial 12",wrap=WORD)
txt.pack()
txt1= Text(tk, width=72,height=10,font="Arial 12",wrap=WORD)
txt1.pack()
txt2= Text(tk, width=72,height=10,font="Arial 12",wrap=WORD)
txt2.pack()
tk.mainloop()
```

3. Установите фиксированный размер формы, например: **tk.geometry('700x600'**).

4. Установите титульную надпись на форме, например: tk.title («Система для автоматического семантического анализа»).

5. Вставьте вспомогательную функцию для очистки всех полей формы и внесите её в меню:

```
def clear_all():
        txt.delete(1.0, END)
        txt1.delete(1.0, END)
        txt2.delete(1.0, END)

file_menu.add_command(label="Clear all fields", command= clear_all)
```

6. Проверьте работу созданной Вами функции. Для этого запустите форму, введите с клавиатуры в каждое поле формы любой текст, выполните через меню созданную Вами функцию очистки. При этом набранный Вами текст должен исчезнуть.

7. Введите функцию записи содержимого всех трёх текстовых полей в текстовый файл и занесите её в меню:

```
def save_text():
save_as = asksaveasfilename()
try:
        x =txt.get(1.0, END)
        x1 =txt1.get(1.0, END)
        x2 =txt2.get(1.0, END)
        f = open(save_as, "w")

st=x+'\n'+x1+'\n'+x2+'\n'
        f.write(st)
        f.close()
except:
        pass

file_menu.add_command(label="Save text", command= save_text)
```

Проверьте работу интерфейса, как показано на рис. 1.

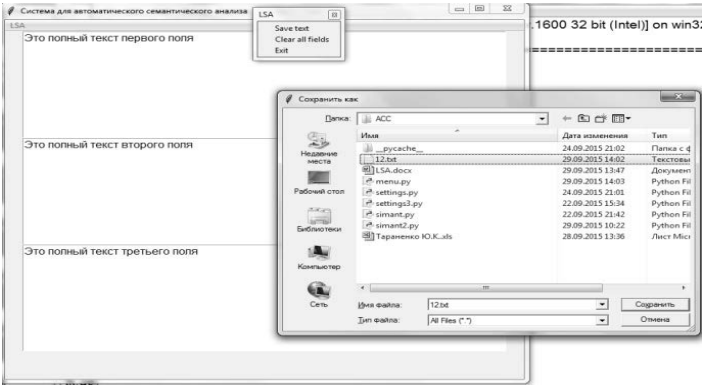

Рис. 1. Сохранение данных всех трёх текстовых полей

Содержание текстового файла при этом должно соответствовать рис. 2.

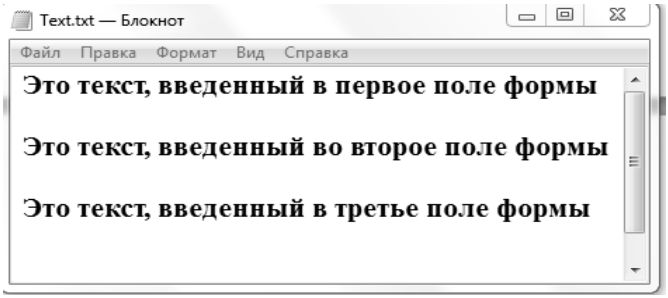

Рис. 2. Содержание текстового файла, сохранённого под именем Test.txt

8. Введите функцию выхода из программы и занесите её в меню:

```
def close_win():
    if askyesno("Exit", "Do you want to quit?"):
        tk.destroy()
file_menu.add_command (label="Exit", command= close_win)
```

Выходу из программы должно предшествовать подтверждение (рис. 3).

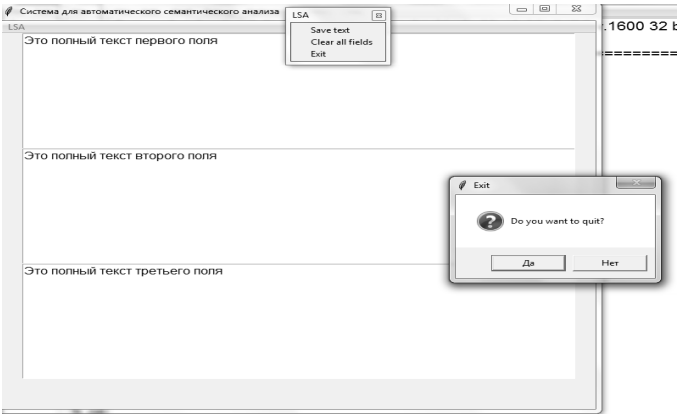

Рис. 3. Выход из программы

9. Добавьте в программу необходимые модули, те, которые, загружали в практической работе № 1:

```
import numpy
from numpy import *
import nltk
import scipy
```

10. Загрузите один из модулей settings.py, выбирая только необходимые переменные: from settings import docs, stem.

11. Загрузите процедуру стемминга (см. пример практической работы № 1) с учётом того, что переменную stem с обозначением выбранного языка мы получили из settings:

```
 from nltk.stem import  SnowballStemmer
stemmer = SnowballStemmer(stem)
doc=docs
```

12. Визуализируйте документы в первом поле формы при помощи процедуры:

```
def STart():
   txt.insert(END,'Исходные документы\n')
```

```
    for k, v in enumerate(docs):
        txt.insert(END,'Ном.док--%u Текст-%s \n'%(k,v))
    #return Word_1()
```

Занесите эту процедуру в программу. Процедура **def STart()** передаёт управление следующей процедуре **return Word_1()** без аргументов. Следующая процедура-функция Word_1() отбирает из общего массива слов во всех документах только те, которые встречаются один раз, приводит все слова к нижнему регистру, отбирает только те, которые содержат больше одной буквы, и отделяет основу слова:

```
def word_1():
    word=nltk.word_tokenize((' ').join(doc))
     n=[stemmer.stem(w).lower() for w in word if len(w) >1 and w.isalpha()]
    fdist=nltk.FreqDist(n)
    t=fdist.hapaxes()
     txt1.insert(END,'Слова которые встричаються только один раз:\n%s'%t)
    txt1.insert(END,'\n')
    # return WordStopDoc(t)
```

Занесите процедуру **def word_1()** в программу. Для запуска **def word_1()** добавьте следующую строку в меню:

```
file_menu.add_command(label="Start", command=  STart)
```

13. Проверьте работу созданной части программы, она должна иметь вид, как показано на рис. 4. (**у Вас свои документы**!)

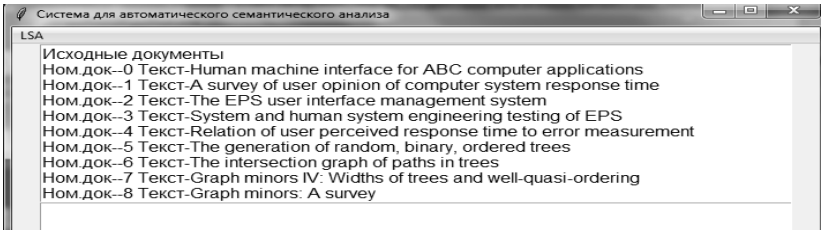

Рис. 4. Исходные документы и основы слов, которые встречаются один раз в первом поле формы

Распечатка содержимого первого поля показана на рис. 5.

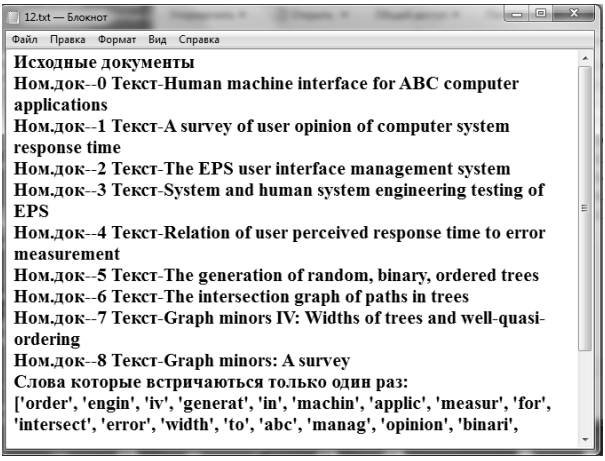

Рис. 5. Исходные документы по номерам и основы слов, которые встречаются один раз, приведенные к нижнему регистру и без учёта цифр и слов, состоящих из одной буквы

Вопросы к практической работе № 2

1. Перечислите, какие этапы LSA, исключая установку модулей, реализованы в данной работе на Python.

2. Какая функция Python позволяет отбирать только слова, состоящие из букв?

3. Какое условие позволяет отбирать слова, содержащие больше одной буквы?

4. Какая функция приводит слова к нижнему регистру?

5. Какая часть и какой процедуры отбирает из списков документов отдельные слова, разделяя их пробелом? Почему эта часть создаёт список слов, а не список букв?

6. В какой части и в какой процедуре-функции осуществляется процедура стемминга?

7. Как в один файл можно записать содержимое трёх различных полей?

8. Как передаётся управление от процедуры def STart() к процедуре def word_1()?

9. Что означают цифры в скобках при чтении данных из поля txt1.get(**1.0, END**, а что означает **END** при записи данных в поле?

Задания к практической работе № 2

1. Выполните все пункты практической работы № 2, создайте свои скриншоты с пояснениями, скопируйте текст основного модуля menu.py, включая последнюю процедуру, занесите их в отчёт для двух setting.py.

2. Ответы на вопросы с подробными пояснениями занесите в отчёт по практической работе № 2.

3. Оформите отчет по схеме, указанной в практической работе № 1.

№ 3. Разработка вспомогательных процедур для латентно-семантического анализа

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки по разработке процедур исключения стоп-слов и определения распределения слов по документам и построению матрицы «слова – документы» в среде программирования **Python.**

Ход выполнения работы:

1. Добавьте к получаемым из модуля setting.py переменным переменную stopwords. Теперь запрос к setting.py должен выглядеть так: from settings import docs, stem, stopwords.

2. Снимите значок # комментарий со строки return WordStopDoc(t).

3. В основной модуль menu.py добавьте следующую процедуру-функцию:

```
def WordStopDoc(t):
    d={}
    c=[]
    for i in range(0,len(doc)):
        word=nltk.word_tokenize(doc[i])
        for w in word:
            if stemmer.stem(w).lower() not in t and len(w) >1 and w.isalpha() and w not in stopwords:
                m=stemmer.stem(w).lower()
                if m not in c:
                    c.append(m)
                    d[m]= [i]
                elif m in c:
                    d[m]= d[m]+[i]
```

```
    txt1.insert(END,'Стоп-слова:\n')
    txt1.insert(END,stopwords)
    txt1.insert(END,'\n')
    txt1.insert(END,'Слова(основа):\n')
    txt1.insert(END,c)
    txt1.insert(END,'\n')
    txt1.insert(END,'Распределение слов по документам:\n')
    txt1.insert(END,d)
    txt1.insert(END,'\n')
  #return Create_Matrix(d,c,t)
```

4. Проверьте работу введенной процедуры, запустив menu. ру, у Вас должны получиться интерфейс (рис. 1) и распечатка (рис. 2).

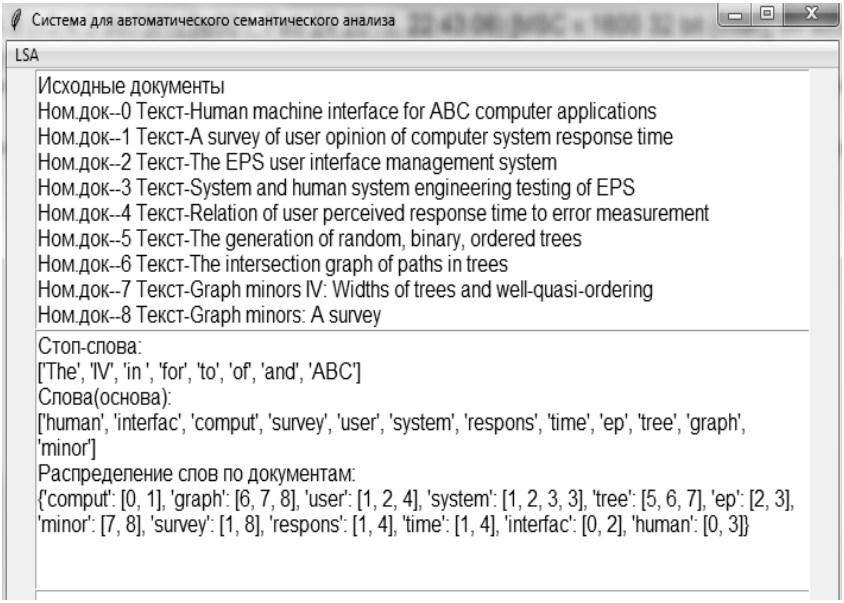

Рис. 1. Основная форма программы с результатами исключения стоп-слов и распределением основ слов в нижнем регистре по документам

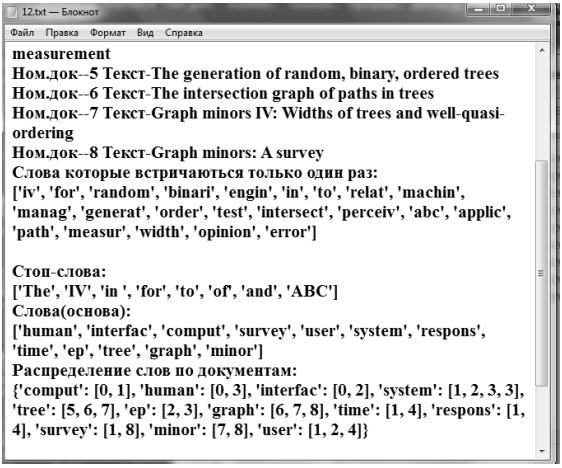

Рис. 2. Распечатка содержимого двух полей формы рис. 1

5. Проанализируйте распределение основ слов по документам рис. 1, 2. Например: слово **system** встречается в документе № 3 два раза, причём наиболее часто (6 раз) встречаются слова из документа № 1.

6. Снимите значок # комментария со строки **return Create_Matrix(d,c,t)** и добавьте процедуру-функцию:

```
def Create_Matrix(d,c,t):
    a=len(c)
    b=len(doc)
    A = numpy.zeros([a,b])
    c.sort()
    for i, k in enumerate(c):
        for j in d[k]:
            A[i,j] += 1
     txt1.insert(END,'Первая матрица для проверки заполнения строк и столбцов:\n')
    txt1.insert(END,A)
   #return Analitik_Matrix(A,c,t)
```

Сделайте распечатку работы программы (рис. 3) и проведите её анализ. Например: 1. Все строки (12) и столбцы (9) заполнены.

2. Подтверждаются выводы, сделанные в п. 5, а именно: в девятой строке четвёртого столбца (документ № 3) стоит цифра 2, что говорит о том, что слово употребляется два раза, причём наиболее часто (6 раз) встречаются слова из документа № 1, это второй столбец.

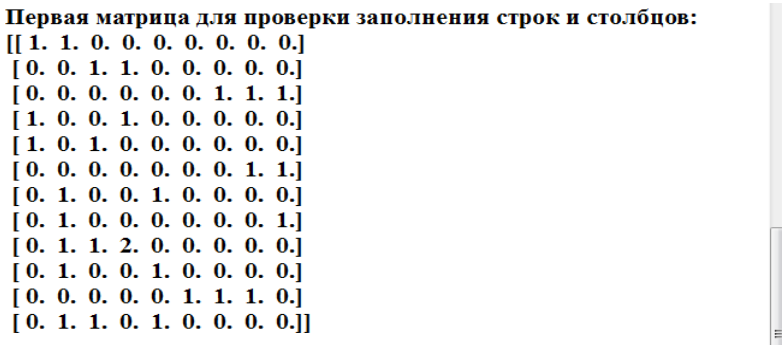

Рис. 3. Первая матрица для проверки заполнения строк и столбцов

Вопросы к практической работе № 3

1. Перечислите, какие этапы LSA реализованы в данной работе на Python.

2. Что сохраняется в списке **c** и зачем?

3. Что сохраняется в словаре **d**? Что в этом словаре относится к ключу, а что к значению?

4. Какое максимальное число значений сохраняет Ваш словарь для settings.py каждого языка?

5. Какую часть процедуры def WordStopDoc(t) нужно изменить, чтобы не учитывать повторений одного и того же слова в документе? Продемонстрируйте изменение и сделайте распечатку. Результаты занесите в отчёт.

6. Зачем в программе сортирую список **c**?

7. Как матрицу заполняют нулями? Приведите в качестве примера строку из программы.

8. На пересечении каких объектов в матрице появляется 1? При каких условиях это число превышает 1?

9. Что означает выражение: A[i,j] += 1.

Задания к практической работе № 3

1. Выполните все пункты практической работы № 3, создайте скриншоты с пояснениями, скопируйте текст основного модуля menu.py, включая последнюю процедуру, занесите их в отчёт, причём приведите скриншоты распечаток для каждого из файлов settings.py.

2. Дайте подробные ответы на вопросы, занесите их в отчёт по практической работе № 3.

3. Оформите отчет по схеме, указанной в практической работе № 1.

№ 4. Разработка процедур для проверки заполнения матрицы «слова – документы» и исключения из неё тех документов, которые не связаны с отфильтрованными словами в среде программирования Python

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки по разработке процедур для проверки заполнения матрицы «слова – документы» и исключения из неё тех документов, которые не связаны с отфильтрованными словами в среде программирования Python.

Ход выполнения работы:

1. Для того чтобы проверить новую функцию программы, которая состоит в исключении из анализа не связанных с отфильтрованными словами документов, нужно добавить в settings.py документы, которые преимущественно содержат стоп-слова и слова, которые встречаются только один раз. Например, такой перечень документов может иметь вид:

0. "Россия отстала от Украины в рейтинге свободных стран мира на 37 позиций",

1. "Самой свободной посткоммунистической страной является Чешская Республика",

2. "Украинские пограничники хотят отгородиться от Приднестровья ‘Европейским валом' ",

3. "Президент страны Александр Лукашенко предложил усилить границы с Востоком.",

4. "Вакцины против полиомиелита распределят по регионам Украины осенью",

5. "Мы знали, что ситуация сложная с обеспечением вакцинами во всех областях, но есть такие, которые эффективно справляются с решением этой ситуации.",

6. "В Красноармейске неизвестные закидали военкомат Коктейлями Молотова",

7. "Работникам военкомата удалось быстро локализовать пожар.",

8. "Минюст требует у властей Италии экстрадиции Маркова",

9. "Министерство юстиции Украины уже передало властям Италии пакет документов по экстрадиции бывшего народного депутата, лидера партии Родина Игоря Маркова.",

10. "В Детройте ловили тигра, сбежавшего с фотосессии.",

11. "Хищника пытались загнать в клетку с помощью газонокосилки.",

12. "Новым директором Департамента массовых коммуникаций ХОГА стала Виктория Аннопольская",

13. "Пресс-секретарь председателя Харьковской облгосадминистрации Виктория Аннопольская назначена директором Департамента массовых коммуникаций ХОГА. До этого Аннопольская занимала должность первого заместителя директора - начальника управления прессы и информации департамента.",

14. "Порошенко встретится с Олландом и Меркель 24 августа",

15. "Среди запланированных для обсуждения вопросов ситуация на Донбассе и состояние выполнения Минских договоренностей, а также вопросы реализации Соглашения об ассоциации.",

16. "Минобразования: В вузах Украины будут учиться более 700 крымчан",

17. "По образовательно-квалификационным уровням младшего специалиста, бакалавра зачислены 207 жителей Крыма, а специалиста и магистра - 554.",

18. "Обвиненный Россией в шпионаже эстонец Кохвер приговорён к 15 годам тюрьмы"

2. Снимите комментарий со строки #return Analitik_ Matrix(A,c,t) и добавьте в программу процедуру-функцию:

```
def Analitik_Matrix(A,c,t):
    wdoc = sum(A, axis=0)
    p=[]
```

```
        q=-1
        for w in wdoc:
            q=q+1
            if w==0:
                p.append(q)
        if len(p)!=0:
            for k in p:
                doc.pop(k)
            word_1()
        elif len(p)==0:
            rows, cols = A.shape
                txt1.insert(END,'Исходная матрица с учётом частоты слова в документе: строки слова -%u столбцы документов--%u \ n'%(rows,cols))
            for i, row in enumerate(A):
                st=(c[i], row)
                txt1.insert(END,st)
                txt1.insert(END,'\n')
            #return TF_IDF(A,c,t)
```

3. При исключении документов программа проводит несколько циклов, каждый раз возвращаясь к процедуре def word_1(), поэтому, чтобы сообщения программы в полях формы не повторялись, в начало процедуры def word_1() нужно ввести:

```
txt1.delete(1.0, END)
                    txt2.delete(1.0, END)
```

Это позволит сохранить в полях txt1, txt2 формы только последнюю запись, когда исключение документов уже заканчивается.

4. Для проверки работы процедуры запустите программу и сделайте распечатку:

Исходные документы:

Ном.док--0 Текст-Россия отстала от Украины в рейтинге свободных стран мира на 37 позиций

Ном.док--1 Текст-Самой свободной посткоммунистической страной является Чешская Республика

Ном.док--2 Текст-Украинские пограничники хотят отгородиться от Приднестровья 'Европейским валом'

Ном.док--3 Текст-Президент страны Александр Лукашенко предложил усилить границы с Востоком.

Ном.док--4 Текст-Вакцины против полиомиелита распределят по регионам Украины осенью

Ном.док--5 Текст-Мы знали, что ситуация сложная с обеспечением вакцинами во всех областях,но есть такие,которые эффективно справляются с решением этой ситуации.

Ном.док--6 Текст-В Красноармейске неизвестные закидали военкомат Коктейлями Молотова

Ном.док--7 Текст-Работникам военкомата удалось быстро локализовать пожар.

Ном.док--8 Текст-Минюст требует у властей Италии экстрадиции Маркова

Ном.док--9 Текст-Министерство юстиции Украины уже передало властям Италии пакет документов по экстрадиции бывшего народного депутата, лидера партии Родина Игоря Маркова.

Ном.док--10 Текст-В Детройте ловили тигра, сбежавшего с фотосессии.

Ном.док--11 Текст-Хищника пытались загнать в клетку с помощью газонокосилки.

Ном.док--12 Текст-Новым директором Департамента массовых коммуникаций ХОГА стала Виктория Аннопольская

Ном. Док—13Текст-Пресс-секретарь председателя Харьковской облгосадминистрации Виктория Аннопольская назначена директором Департамента массовых коммуникаций ХОГА. До этого Аннопольская занимала должность первого заместителя директора - начальника управления прессы и информации департамента.

Ном.док--14 Текст-Порошенко встретится с Олландом и Меркель 24 августа

Ном.док--15 Текст-Среди запланированных для обсуждения вопросов ситуация на Донбассе и состояние выполнения Минских договоренностей, а также вопросы реализации Соглашения об ассоциации.

Ном.док--16 Текст-Минобразования: В вузах Украины будут учиться более 700 крымчан

Ном.док--17 Текст-По образовательно-квалификационным уровням младшего специалиста, бакалавра зачислены 207 жителей Крыма, а специалиста и магистра - 554.

Ном.док--18 Текст-Обвиненный Россией в шпионаже эстонец Кохвер приговорили к 15 годам тюрьмы

Слова, которые встречаются только один раз:

['родин', 'справля', 'народн', 'быстр', 'позиц', 'для', 'выполнен', 'запланирова', 'полиомиелит', 'регион', 'шпионаж', 'юстиц', 'пакет', 'тюрьм', 'сред', 'обсужден', 'коктейл', 'депутат', 'эстонец', 'приговор', 'всех', 'котор', 'красноармейск', 'явля', 'об', 'распредел', 'предлож', 'решен', 'игор', 'треб', 'министерств', 'донбасс', 'знал', 'посткоммунистическ', 'но', 'усил', 'локализова', 'обвинен', 'уж', 'закида', 'такж', 'границ', 'ест', 'во', 'лукашенк', 'работник', 'молотов', 'реализац', 'уда', 'переда', 'состоян', 'парт', 'эт', 'александр', 'бывш', 'от', 'прот', 'чешск', 'мир', 'эффективн', 'что', 'област', 'кохвер', 'отста', 'рейтинг', 'ассоциац', 'соглашен', 'республик', 'договорен', 'пожар', 'осен', 'лидер', 'минск', 'минюст', 'документ', 'сложн', 'год', 'сам', 'мы', 'обеспечен', 'неизвестн', 'восток', 'президент', 'так']

Стоп-слова:

['-', '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']

Слова (основа):

['росс', 'украин', 'свободн', 'стран', 'вакцин', 'ситуац', 'военкомат', 'власт', 'итал', 'экстрадиц', 'марков', 'вопрос']

Распределение слов по документам:

{'власт': [7, 8], 'стран': [0, 1, 2], 'росс': [0, 10], 'марков': [7, 8], 'итал': [7, 8], 'вопрос': [9, 9], 'вакцин': [3, 4], 'ситуац': [4, 4, 9], 'военкомат': [5, 6], 'свободн': [0, 1], 'украин': [0, 3, 8], 'экстрадиц': [7, 8]}

Первая матрица для проверки заполнения строк и столбцов:

```
[[ 0.  0.  0.  1.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  2.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 1.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  2.  0.  0.  0.  0.  1.  0.]
 [ 1.  1.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  1.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  1.  0.  0.]]
```

Исходная матрица с учётом частоты слова в документе: строки слов – 12, столбцы документов – 11:

Вакцин	{[0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0.]}
Власт	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
Военкомат	{[0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]}
Вопрос	{[0. 0. 0. 0. 0. 0. 0. 0. 0. 2. 0.]}
Итал	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
марков	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
Росс	{[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]}
свободн	{[1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]}
ситуац	{[0. 0. 0. 0. 2. 0. 0. 0. 0. 1. 0.]}
Стран	{[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]}
украин	{[1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.]}
экстрадиц	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}

5. Проведите анализ полученной распечатки. Например: исходных документов было 19, осталось 11, 8 документов исключено, что существенно упрощает анализ. После исключения все строки и столбцы исходной матрицы заполнены. Осталось 12 ключевых слов, которые могут характеризовать документы по группам. Наибольшее количество ключевых слов (5) содержит документ № 8.

6. Пока в программу не введены новые элементы обработки целесообразно сравнить полученные результаты с известными, например, популярной Интернет-публикацией [29].

7. Берём документы, приведенные в этой статье, и устанавливаем их в setting.py нашей программы.

0. Британская полиция знает о местонахождении основателя WikiLeaks

1. В суде США начинается процесс против россиянина, рассылавшего спам

2. Церемонию вручения Нобелевской премии мира бойкотируют 19 стран

3. В Великобритании арестован основатель сайта Wikileaks Джулиан Ассандж

4. Украина игнорирует церемонию вручения Нобелевской премии

5. Шведский суд отказался рассматривать апелляцию основателя Wikileaks

6. НАТО и США разработали планы обороны стран Балтии против России

7. Полиция Великобритании нашла основателя WikiLeaks, но, не арестовала

8. В Стокгольме и Осло сегодня состоится вручение Нобелевских премий

Запускаем нашу программу и копируем:

Исходная матрица с учётом частоты слова в документе: строки слов – 13, столбцы документов – 9.

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. 0. 0. 1.]}
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.]}

Сравним с матрицей, полученной вручную в работе [29] (рис.1).

	T1	T2	T3	T4	T5	T6	T7	T8	T9
wililеaks	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	0	0	1
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

Рис. 1. Матрица «основа слова – документ», приведенная в работе [29]

Вывод: получены одинаковые матрицы, причём в нашей программе матрица получена автоматически.

Вопросы к практической работе № 4

1. Перечислите, какие этапы LSA реализованы в данной работе на Python.

2. В какой строке процедуры-функции **def Analitik_ Matrix(A,c,t)** реализована проверка матрицы на наличие пустых столбцов или строк?

3. В какой строке процедуры-функции **def Analitik_ Matrix(A,c,t)** реализовано исключение документов из общего списка и при каких условиях?

4. Что нужно дописать в процедуре **word_1(),** чтобы при повторном анализе уменьшающейся группы документов сообщения программы, выводимые в текстовые поля, не повторялись?

5. При каких условиях в LSA возникает проблема пустых столбцов и строк?

6. Какие преимущества имеет наша программа в сравнении с методикой в работе [29].

Задания к практической работе № 4

1. Выполните все пункты практической работы № 4, создайте скриншоты или распечатки результатов с пояснениями, скопируйте текст основного модуля menu.py, включая последнюю процедуру, занесите их в отчёт, причём сделайте скриншоты распечаток для каждого из файлов settings.py.

2. Дайте подробные ответы на вопросы и занесите их в отчёт по практической работе № 4?

3. Внесите изменения в программу, чтобы в шапке матрицы выводились номера документов, как показано на рис. 1.

3. Оформите отчет по схеме, указанной в практической работе № 1.

№ 5. Разработка процедур для нормализации частотной матрицы и ее разложение

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки по разработке процедур для нормализации полученной частотной матрицы методом TF-IDF с последующим её сингулярным разложением и определением семантических координат ключевых слов и документов, а также отдельного ключевого слова, от которого можно отсчитывать семантическое расстояние до остальных слов и документов в среде программирования Python.

Ход выполнения работы:

1. Введите в программу процедуру-функцию **def TF_IDF (A,c, t**), со строки #return TF_IDF(A,c,t) снимите комментарий, используйте setting.py, полученный Вами в практической работе № 4 с документами, часть которых исключает программа из-за пустых строк или столбцов частотной матрицы:

```
def TF_IDF(A,c,t):
    wpd = sum(A, axis=0)
    dpw= sum(asarray(A > 0,'i'), axis=1)
    rows, cols = A.shape
   txt1.insert(END,'Нормализованная по методу TF-IDF матрица: строки слов -%u столбцы документов--%u \n'%(rows,cols))
    for i in range(rows):
       for j in range(cols):
          m=float(A[i,j])/wpd[j]
          n=log(float(cols) /dpw[i])
          A[i,j] =round(n*m,2)
    for i, row in enumerate(A):
       st=(c[i], row)
```

```
        txt1.insert(END,st)
        txt1.insert(END,'\n')
    #return U_S_Vt(A,c,t)
```

2. Для проверки работы программы сделайте распечатку и выберите только нормализованную матрицу:

Нормализованная по методу TF-IDF матрица: строки слов – 12, столбцы документов – 11.

вакцин	{[0. 0. 0. 0.85 0.57 0. 0. 0. 0. 0. 0.]}
власт	{[0. 0. 0. 0. 0. 0. 0. 0.43 0.34 0. 0.]}
военкомат	{[0. 0. 0. 0. 0. 1.7 1.7 0. 0. 0. 0.]}
вопрос	{[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.6 0.]}
итал	{[0. 0. 0. 0. 0. 0. 0. 0.43 0.34 0. 0.]}
марков	{[0. 0. 0. 0. 0. 0. 0. 0.43 0.34 0. 0.]}
росс	{[0.43 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.7]}
свободн	{[0.43 0.85 0. 0. 0. 0. 0. 0. 0. 0. 0.]}
ситуац	{[0. 0. 0. 0. 1.14 0. 0. 0. 0. 0.57 0.]}
стран	{[0.32 0.65 1.3 0. 0. 0. 0. 0. 0. 0. 0.]}
украин	{[0.32 0. 0. 0.65 0. 0. 0. 0. 0.26 0. 0.]}
экстрадиц	{[0. 0. 0. 0. 0. 0. 0. 0.43 0.34 0. 0.]}

3. Сравните полученную в п. 2 нормализованную матрицу с исходной частотной матрицей, полученной в практической работе № 4:

вакцин	{[0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0.]}
власт	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
военкомат	{[0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]}
вопрос	{[0. 0. 0. 0. 0. 0. 0. 0. 0. 2. 0.]}
итал	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
марков	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}
росс	{[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]}
свободн	{[1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]}
ситуац	{[0. 0. 0. 0. 2. 0. 0. 0. 0. 1. 0.]}
стран	{[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0.]}
украин	{[1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.]}
экстрадиц	{[0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]}

Например, большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа – **военкомат, во-**

прос, росс, ситуац, и с низкой частотой употреблений в других документах.

4. Введите в программу процедуру-функцию **def U_S_ Vt(A,c,t)**, со строки #return **U_S_Vt(A,c,t)** снимите комментарий, используйте тот же setting.py, что и в п. 1:

```
def U_S_Vt(A,c,t):
    U, S,Vt = numpy.linalg.svd(A)
    rows, cols = U.shape
    for j in range(0,cols):
        for i  in range(0,rows):
            U[i,j]=round(U[i,j],3)
    txt1.insert(END,' Первые 2 столбца ортогональной матрицы U слов, сингулярного преобразования нормализованной матрицы: строки слов -%u\n'%rows)
    for i, row in enumerate(U):
        st=(c[i], row[0:2])
        txt1.insert(END,st)
        txt1.insert(END,'\n')
        for i in range(0,len(c)):
            if c[i]==stemmer.stem(wordd).lower():
                kt=i
    res1=-1*U[:,0:1]
    wx=res1[kt]
    res2=-1*U[:,1:2]
    wy=res2[kt]
    txt1.insert(END,' Координаты x --%f и y--%f опорного слова --%s, от которого отсчитываются все расстояния \n'%(wx,wy,wordd))
    txt1.insert(END,'Диагональная матрица S \n')
    txt1.insert(END,S)
    txt1.insert(END,'\n')
    rows, cols = Vt.shape
    for j in range(0,cols):
        for i  in range(0,rows):
            Vt[i,j]=round(Vt[i,j],3)
    txt1.insert(END,' Первые 2 строки ортогональной матрицы Vt документов сингулярного преобразования нормализованной матрицы: столбцы документов -%u\n'%cols)
    st=(-1*Vt[0:2, :])
```

```
    txt1.insert(END,st)
    txt1.insert(END,'\n')
    res3=(-1*Vt[0:1, :])
    res4=(-1*Vt[1:2, :])
     #return Word_Distance_Document(res1,wx,res2,wy,res3,re
s4,Vt,t,c)
```

5. Для работы процедуры **def U_S_Vt(A,c,t)** выберите из нормализованной матрицы наиболее весомое слово, например **“Украина”**, и введите его для значения переменной word в Ваш файл setting.py.

6. Проверьте работу процедуры **def U_S_Vt(A,c,t)**, для этого распечатайте только результаты роботы этой процедуры сингулярного разложения нормализованной частотной матрицы:

Первые 2 столбца ортогональной матрицы U слов, сингулярного преобразования нормализованной матрицы: строки слов – 12

вакцин	{[0. 0.1765]}
власт	{[0. 0.0026]}
военкомат	{[-1. 0.0000]}
вопрос	{[0. 0.7799]}
итал	{[0. 0.0026]}
марков	{[0. 0.0026]}
росс	{[0. 0.2595]}
свободн	{[0. 0.0481]}
ситуац	{[0. 0.5308]}
стран	{[0. 0.0774]}
украин	{[0. 0.0575]}
экстрадиц	{[0. 0.0026]}

Координаты x---0.0000 и y---0.0575 опорного слова **Украина**, от которого отсчитываются все расстояния.

Диагональная матрица S

2.4042
1.7834
1.7761
1.5668
1.3105
1.1128
0.9032

0.7742
0.2502
0.0
0.0

Первые 2 строки ортогональной матрицы Vt документов сингулярного преобразования нормализованной матрицы: столбцы документов – 11

0.0000 0.0000 0.0000 0.0000 0.0000 0.7071 0.7071 0.0000 0.000 0.0 0.0

-0.0984 -0.0511 -0.0564 -0.1051 -0.3957 0.0 0.0 -0.0025 -0.0103 -0.8693 -0.2473

Для проверки сингулярного разложения частотной матрицы воспользуемся самой известной математической программой MATLAB 7.11.0(R2010b) [31]. В окно Command Windows MATLAB введём нормализованную по методу TF-IDF матрицу, видоизменённую по правилам MATLAB:

```
A= [ 0   0   0  0.85  0.57  0   0   0   0   0   0 ;
 0   0   0   0   0   0   0  0.43  0.34  0   0 ;
 0  0   0 0   0   1.7  1.7  0   0   0   0 ;
 0  0   0  0   0   0   0   0   0   1.6  0 ;
 0  0   0   0   0   0   0   0.43  0.34  0  0 ;
 0  0   0   0   0   0   0   0.43  0.34  0  0 ;
 0.43  0   0   0   0   0   0   0   0   0   1.7 ;
 0.43  0.85  0   0   0   0   0   0   0   0   0;
 0  0   0   0 1.14  0   0   0   0   0.57  0  ;
 0.32  0.65  1.3   0   0   0   0   0   0   0   0;
 0.32  0   0   0.65  0   0   0   0   0.26  0  0;
 0  0   0   0   0   0   0  0.43  0.34  0   0]
```

Полное сингулярное разложение запишем в виде:

```
[U,S,V]=svd(A)
A=[ 0   0   0  0.85  0.57  0   0   0   0   0   0 ;
 0   0   0   0   0   0   0   0.43  0.34  0   0 ;
 0  0   0 0   0   1.7  1.7  0   0   0   0 ;
 0  0   0  0   0   0   0   0   0   1.6  0 ;
 0  0   0   0   0   0   0   0.43  0.34  0  0 ;
```

```
 0  0   0    0    0   0    0    0.43  0.34  0  0 ;
 0.43  0    0    0    0    0    0    0    0  0   1.7 ;
 0.43  0.85  0    0    0    0    0    0    0    0    0;
 0  0   0   0 1.14  0   0   0   0    0.57  0  ;
 0.32  0.65  1.3   0   0   0   0    0    0   0    0;
 0.32  0   0   0.65  0    0    0    0    0.26  0  0;
 0  0   0   0    0    0   0  0.43  0.34  0   0]
>> [U,S,V]=svd(A)
U =

 Columns 1 through 8
   0.0000  -0.1765  -0.0307   0.0241   0.6932   0.0700   0.3055  -0.1502
  -0.0000  -0.0026   0.0028   0.0042   0.0600  -0.4754  -0.1316   0.0102
  -1.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
   0.0000  -0.7799  -0.2324  -0.0156  -0.4514  -0.1291   0.3127  -0.0755
  -0.0000  -0.0026   0.0028   0.0042   0.0600  -0.4754  -0.1316   0.0102
  -0.0000  -0.0026   0.0028   0.0042   0.0600  -0.4754  -0.1316   0.0102
  -0.0000  -0.2595   0.9102  -0.3146  -0.0290   0.0140  -0.0421  -0.0421
   0.0000  -0.0481   0.1570   0.3585  -0.0008  -0.0141   0.1816   0.8937
  -0.0000  -0.5308  -0.1515   0.0018   0.4171   0.1871  -0.5980   0.1624
  -0.0000  -0.0774   0.2553   0.8764  -0.0629   0.0285  -0.1275  -0.3738
   0.0000  -0.0575   0.0613   0.0594   0.3500  -0.1958   0.5752  -0.0692
  -0.0000  -0.0026   0.0028   0.0042   0.0600  -0.4754  -0.1316   0.0102

 Columns 9 through 12
  -0.6050   -0.0000    0.0000   -0.0000
  -0.0545    0.7789   -0.2223   -0.3066
        0    0.0000   -0.0000    0.0000
  -0.1168    0.0000   -0.0000   -0.0000
  -0.0545   -0.1309    0.8461   -0.1305
  -0.0545   -0.6124   -0.4536   -0.4114
  -0.0254   -0.0000    0.0000    0.0000
  -0.1122    0.0000   -0.0000   -0.0000
   0.3199    0.0000   -0.0000    0.0000
   0.0042   -0.0000    0.0000    0.0000
   0.7021    0.0000    0.0000   -0.0000
  -0.0545   -0.0356   -0.1701    0.8484
S =
 Columns 1 through 8
```

```
    2.4042         0         0         0         0         0         0         0
         0    1.7834         0         0         0         0         0         0
         0         0    1.7761         0         0         0         0         0
         0         0         0    1.5668         0         0         0         0
         0         0         0         0    1.3105         0         0         0
         0         0         0         0         0    1.1128         0         0
         0         0         0         0         0         0    0.9032         0
         0         0         0         0         0         0         0    0.7742
         -         -         -         -         -         -         -         -    -
```

Columns 9 through 11

```
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    0.2502         0         0
         0    0.0000         0
         0         0    0.0000
         0         0         0
```

V =

Columns 1 through 8

```
    0.0000   -0.0984    0.3154    0.2032    0.0603   -0.0482    0.2250    0.2899
    0.0000   -0.0511    0.1686    0.5581   -0.0317    0.0059    0.0792    0.6674
   -0.0000   -0.0564    0.1868    0.7272   -0.0624    0.0333   -0.1836   -0.6276
    0.0000   -0.1051    0.0078    0.0377    0.6232   -0.0609    0.7015   -0.2230
   -0.0000   -0.3957   -0.1071    0.0101    0.6644    0.2275   -0.5620    0.1286
   -0.7071   -0.0000   -0.0000   -0.0000    0.0000    0.0000    0.0000    0.0000
   -0.7071   -0.0000   -0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0000
   -0.0000   -0.0025    0.0027    0.0046    0.0788   -0.7347   -0.2507    0.0226
   -0.0000   -0.0103    0.0111    0.0135    0.1317   -0.6267   -0.0326   -0.0054
    0.0000   -0.8693   -0.2580   -0.0152   -0.3697   -0.0898    0.1765   -0.0366
   -0.0000   -0.2473    0.8712   -0.3413   -0.0376    0.0213   -0.0793   -0.0925
```

Columns 9 through 11

```
 0.6667   0.2129  -0.4681
-0.3703  -0.1077   0.2368
 0.0219   0.0014  -0.0032
-0.2313  -0.0000   0.0000
 0.0794   0.0000   0.0000
 0.0000  -0.6437  -0.2927
-0.0000   0.6437   0.2927
-0.3744   0.2071  -0.4555
 0.4334  -0.2620   0.5761
-0.0183   0.0000   0.0000
-0.1724  -0.0538   0.1184
```

Сравнивая первые два столбца полученной нами ортогональной матрицы U в п. 5, с первыми двумя столбцами сингулярного разложения матицы A, видим их полное совпадение в цифрах. Разница состоит только в знаке, что объясняется особенностью дальнейшего использования. Для дальнейшего анализа полученных результатов воспользуйтесь данными табл. 1–3.

Таблица 1

Сравнение результатов сингулярного разложения в MATLAB и Python для матрицы U

Ортогональная матрица U 2 столбца Python		Ортогональная матрица U 2 столбца MATLAB	
Координата-X	Координата-Y	Координата-X	Координата-Y
0. 0000	0.1765	0.0000	-0.1765
0. 0000	0.0026	-0.0000	-0.0026
-1. 0000	0.0000	-1.0000	0.0000
0. 0000	0.7799	0.0000	-0.7799
0. 0000	0.0026	-0.0000	-0.0026
0. 0000	0.0026	-0.0000	-0.0026
0. 0000	0.2595	-0.0000	-0.2595
0. 0000	0.0481	0.0000	-0.0481
0. 0000	0.5308	-0.0000	-0.5308
0. 0000	0.0774	-0.0000	-0.0774
0. 0000	0.0575	0.0000	-0.0575
0. 0000	0.0026	-0.0000	-0.0026

Самый простой способ сравнить матрицу V сингулярного разложения матрицы A, полученной в MATLAB, с транспонированной матрицей Vt, полученной в Python, – транспонирование V непосредственно в MATLAB. Для этого после вывода результатов сингулярного разложения нужно записать V'. Получим:

```
V'
ans =
  Columns 1 through 6
    0.0000    0.0000   -0.0000    0.0000   -0.0000   -0.7071
   -0.0984   -0.0511   -0.0564   -0.1051   -0.3957   -0.0000
    0.3154    0.1686    0.1868    0.0078   -0.1071   -0.0000
    0.2032    0.5581    0.7272    0.0377    0.0101   -0.0000
    0.0603   -0.0317   -0.0624    0.6232    0.6644    0.0000
   -0.0482    0.0059    0.0333   -0.0609    0.2275    0.0000
    0.2250    0.0792   -0.1836    0.7015   -0.5620    0.0000
    0.2899    0.6674   -0.6276   -0.2230    0.1286    0.0000
    0.6667   -0.3703    0.0219   -0.2313    0.0794    0.0000
    0.2129   -0.1077    0.0014   -0.0000    0.0000   -0.6437
   -0.4681    0.2368   -0.0032    0.0000    0.0000   -0.2927


  Columns 7 through 11
   -0.7071   -0.0000   -0.0000    0.0000   -0.0000
   -0.0000   -0.0025   -0.0103   -0.8693   -0.2473
   -0.0000    0.0027    0.0111   -0.2580    0.8712
   -0.0000    0.0046    0.0135   -0.0152   -0.3413
    0.0000    0.0788    0.1317   -0.3697   -0.0376
   -0.0000   -0.7347   -0.6267   -0.0898    0.0213
    0.0000   -0.2507   -0.0326    0.1765   -0.0793
    0.0000    0.0226   -0.0054   -0.0366   -0.0925
   -0.0000   -0.3744    0.4334   -0.0183   -0.1724
    0.6437    0.2071   -0.2620    0.0000   -0.0538
    0.2927   -0.4555    0.5761    0.0000    0.1184
```

Таблица 2

Сравнение результатов сингулярного разложения в MATLAB и Python для матрицы Vt

Ортогональная трансформированная матрица **V** – две строки **MATLAB**					
0.0000	0.0000	0.0000	0.0000	0.0000	0.7071
-0.098	-0.0511	-0.0564	-0.1051	-0.3957	0.0000
-0.7071	0.0000	0.0000	0.0000	0.0000	X
0.0000	-0.0025	-0.0103	-0.8693	-0.2473	**Y**
Ортогональная матрица **Vt** – две строки **Python**					
0.0000	**0.0000**	0.0000	0.0000	0.0000	0.7071
-0.0984	-0.0511	-0.0564	-0.1051	-0.3957	-0.0000
0.7071	0.0000	0.0000	0.0000	0.0000	X
0.0000	-0.0025	-0.0103	-0.8693	-0.2473	Y

Таблица 3

Сравнение результатов сингулярного разложения в MATLAB и Python для диагональной матрицы S

Диагональная матрица S MATLAB	**Диагональная матрица S Python**
2.4042	2.4042
1.7834	1.7834
1.7761	1.7761
1.5668	1.5668
1.3105	1.3105
1.1128	1.1128
0.9032	0.9032
0.7742	0.7742
0.2502	0.2502
0.0000	0.0000
0.0000	0.0000

Сделайте вывод на основании проведенного по Вашим данным анализа. Например, табл. № 1–3 свидетельствуют о том, что сингулярное разложение нормализованной матрицы «слова – документы» в Python полностью идентично разложению в специали-

зированной на матричных преобразованиях программе MATLAB, а следовательно, полученным Вами результатам можно доверять.

7. Введите в программу процедуру-функцию **Word_Distance_Document(res1,wx,res2,wy,res3,res4,Vt,t,c)**, со строки #return Word_Distance_Document(res1,wx,res2,wy,res3,res4,Vt,t,c) снимите комментарий. (**Внимание**! Используйте тот же setting.py, что и в предыдущем п. 6):

```
def Word_Distance_Document(res1,wx,res2,wy,res3,res4,Vt,t,c):
    xx, yy = -1 * Vt[0:2, :]
    arts = []
    p={}
    txt2.insert(END,'Результаты анализа: Всего документов:%u. Осталось документов после исключения не связанных:%u\n'%(ddd,len(doc)))
    for k, v in enumerate(doc):
        word=nltk.word_tokenize(v)
        txt2.insert(END,'Ном.док--%u Текст-%s \n'%(k,v))
          p[k]=[stemmer.stem(w).lower()  for w in word if stemmer.stem(w).lower() not in t and len(w) >1 and w.isalpha() and w not in stopwords]
        ax, ay = xx[k], yy[k]
        dx, dy = float(wx - ax), float(wy - ay)
        arts.append((k,p[k],round(float(sqrt(dx * dx + dy * dy)),3)))
    q=(sorted(arts,key = lambda a: a[2]))
    for i in range(1,len(doc)):
        if len([w for w in (q[i-1])[1] if w in (q[i])[1]])!=0:
            zz=[w for w in (q[i-1])[1] if w in (q[i])[1]]
        else:
            zz=['Нет общих слов']
        qq=round(float((q[i])[2]-(q[i-1])[2]),3)
          tr='№№ Док %u,%u.Общие слова - '%((q[i-1])[0],(q[i])[0]),zz,'Расстояние между документами-',qq,'от ключевого слова -',wordd
        txt2.insert(END, tr)
        txt2.insert(END,'\n')
    #return Grafics_End(res1,res2,res3,res4,c)
```

8. Проверьте работу процедуры **Word_Distance_Document (res1,wx,res2,wy, res3,res4,Vt,t,c))**, для этого распечатайте только результаты работы этой процедуры:

Результаты анализа. Всего документов: 19. Осталось документов после исключения не связанных: 11

Ном.док--0 Текст-Россия отстала от Украины в рейтинге свободных стран мира на 37 позиций

Ном.док--1 Текст-Самой свободной посткоммунистической страной является Чешская Республика

Ном.док--2 Текст-Президент страны Александр Лукашенко предложил усилить границы с Востоком.

Ном.док--3 Текст-Вакцины против полиомиелита распределят по регионам Украины осенью

Ном.док--4 Текст-Мы знали, что ситуация сложная с обеспечением вакцинами во всех областях,но есть такие,которые эффективно справляются с решением этой ситуации.

Ном.док--5 Текст-В Красноармейске неизвестные закидали военкомат Коктейлями Молотова

Ном.док--6 Текст-Работникам военкомата удалось быстро локализовать пожар.

Ном.док--7 Текст-Минюст требует у властей Италии экстрадиции Маркова

Ном.док--8 Текст-Министерство юстиции Украины уже передало властям Италии пакет документов по экстрадиции бывшего народного депутата, лидера партии Родина Игоря Маркова.

Ном.док--9 Текст-Среди запланированных для обсуждения вопросов ситуация на Донбассе и состояние выполнения Минских договоренностей, а также вопросы реализации Соглашения об ассоциации.

Ном.док--10 Текст-Обвиненный Россией в шпионаже эстонец Кохвер приговорили к 15 годам тюрьмы

{№№ Док 2,1.Общие слова - } {['стран']} {Расстояние между документами-} 0.005 {от ключевого слова -} Украина

{№№ Док 1,0.Общие слова - } {['свободн', 'стран']} {Расстояние между документами-} 0.035 {от ключевого слова -} Украина

{№№ Док 0,8.Общие слова - } {['украин']} {Расстояние между документами-} 0.006 {от ключевого слова -} Украина

{№№ Док 8,3.Общие слова - } {['украин']} {Расстояние между документами-} 0.001 {от ключевого слова -} Украина

{№№ Док 3,7.Общие слова-}{['Нет общих слов']} {Расстояние между документами-} 0.007 {от ключевого слова -} Украина

{№№ Док 7,10.Общие слова - } {['Нет общих слов']} {Расстояние между документами-} 0.135 {от ключевого слова -} Украина

{№№ Док 10,4.Общие слова - } {['Нет общих слов']} {Расстояние между документами-} 0.148 {от ключевого слова -} Украина

{№№ Док 4,5.Общие слова - } {['Нет общих слов']} {Расстояние между документами-} 0.371 {от ключевого слова -} Украина

{№№ Док 5,6.Общие слова - } {['военкомат']} {Расстояние между документами-} 0.0 {от ключевого слова -} Украина

{№№ Док 6,9.Общие слова - } {['Нет общих слов']} {Расстояние между документами-} 0.103 {от ключевого слова -} Украина

9. Проведите анализ работы процедуры **Word_Distance_Document** для Вашего setting.py. Например: в нашем примере п. 8 расстояние между документами № 5, 6 равно 0, что свидетельствует об их смысловой близости. Учитывая то, что графический анализ семантического пространства даже в двухмерном варианте является более точным и будет рассмотрен в следующей работе, алогизм некоторых расстояний не должен вызывать недоверие к методу, тем более что при правильном выборе точки отсчёта – ключевого слова – логика в целом сохраняется.

Вопросы к практической работе № 5

1. Перечислите, какие этапы LSA реализованы в данной работе на Python.

2. В каких строках процедуры-функции **def U_S_Vt(A,c,t)** реализовано определение координат опорного слова?

3. Почему координаты опорного слова берутся из первых двух столбцов матрицы U?

4. Как к строкам матрицы U добавить соответствующие ключевые слова? Приведите часть кода.

5. Как выбрать из диагональной матрицы S числовые значения главной диагонали с заданным числом знаков после запятой?

6. Как в процедуре-функции **Word_Distance_Document** получают координаты документов? Приведите в качестве примера строку кода.

7. Какие преимущества для анализа имеет матрица, получаемая в процедуре def TF_IDF(A,c,t)?

8. В процедуре-функции **Word_Distance_Document** есть строка **q=(sorted(arts, key = lambda a: a[2]))**. Поясните, что означает в этой строке выражение **key = lambda a: a[2].**

9. В процедуре-функции **Word_Distance_Document** есть строка: **zz=[w for w in (q[i-1])[1] if w in (q[i])[1]].** Поясните, какое условие реализует эта строка.

10. Какие данные хранятся в списке **arts** строки **q** вопроса 8. Для ответа на этот вопрос выведите содержание списка arts командой print, результаты занесите в отчет.

11. Какие данные выводит строка **qq=round(float((q[i])[2]-(q[i-1])[2]),3)** процедуры **Word_Distance_Document**? Для ответа на этот вопрос выведите содержание строки **qq** командой **print,** результаты занесите в отчет.

12. Какие значения выводит строка tr='№№ Док %u,%u. Общие слова - '%((q[i-1])[0],(q[i])[0]),zz,'Расстояние между документами-',qq,'от ключевого слова -',wordd? Воспользуйтесь командой **print,** результаты занесите в отчет.

Задания к практической работе № 5

1. Выполните все пункты практической работы № 5, создайте скриншоты или распечатки результатов с пояснениями, скопируйте текст основного модуля menu.py, включая последнюю процедуру, занесите их в отчёт, причём сделайте скриншоты распечаток для каждого из файлов settings.py.

2. Дайте подробные ответы на вопросы и занесите их в отчёт по практической работе № 5.

3. Оформите отчет по схеме, указанной в практической работе № 1.

№ 6. Разработка процедур для графического анализа

Цель работы: получить необходимые для автоматизации семантического анализа знания и практические навыки по разработке процедур для графического анализа положения ключевых слов и документов с последующим их выделением в группы средствами графического пакета matplotlib в среде программирования Python.

Ход выполнения работы:

1. Подготовьте среду программирования **Python 3.41** для работы с графическим пакетом matplotlib, для этого:

а) скачайте с сайтов разработчиков (при работе в компьютерном классе предоставляются преподавателем) и установите самораспаковывающиеся модули **matplotlib-1.4.3.win32-py3.4.exe** и **pyparsing-2.0.2.win32-py3.4.exe**

б) скачайте с сайтов разработчиков, разархивируйте и вручную установите пакеты модулей: папка с именем **python-dateutil-2.4.2** и папка с именем **six-1.10.0** (при работе в компьютерном классе предоставляются преподавателем). Для этого папки размещаются в директориях по путям **Path_1, Path_2** с доступом для установки программ. После этого через командную строку последовательно войдите в каждую папку командой **cd Path_1** или **cd Path_2** введите в командную строку команду **python setup.py install,** далее ожидайте полной установки соответствующего модуля и после окончания установок введите команду **exit**.

в) откройте интерфейс **IDLE Python 3.4 GUI** и последовательно в строку приглашения “>>” введите:

>> import **matplotlib-**

>> import **pyparsing**

>> import **dateutil**

>> import **six**

Убедившись, что среда не выдала ошибки и ввод модулей закончился новым приглашением “>>”, можно приступить к выполнению лабораторной работы.

Примечание: программа может выдать ошибку и при наличии модуля из-за несоблюдения правила ввода, например, лишний пробел от приглашения выдаст следующую ошибку:

>>> import matplotlib

SyntaxError: unexpected indent

Необходимо ввести всё заново с минимальным пробелом от строки приглашения.

2. В область загрузки модулей Вашей программы menu.py нужно добавить следующие загрузки:

а) самого графического пакета:

import matplotlib.pyplot as plt

import matplotlib as mpl

б) установки шрифтов для правильного отображения кириллических подписей на графиках:

mpl.rcParams['font.family'] = 'fantasy'

mpl.rcParams['font.fantasy'] = 'Comic Sans MS, Arial'

в) проверьте, установлены ли следующие модули:

import numpy

from numpy import

При необходимости установите их дополнительно.

3. Снимите комментарий со строки #return и установите последнюю процедуру-функцию:

```
def Grafics_End(res1,res2,res3,res4,c):
    plt.title('Semantic space', size=14)
    plt.xlabel('x-axis', size=14)
    plt.ylabel('y-axis', size=14)
    e1=(max(res1)-min(res1))/len(c)
    e2=(max(res2)-min(res2))/len(c)
    e3=(max(res3[0])-min(res3[0]))/len(doc)
    e4=(max(res4[0])-min(res4[0]))/len(doc)
```

```
                plt.axis([min(res1)-e1,  max(res1)+e1,  min(res2)-e2,
max(res2)+e2])
        plt.plot(res1, res2, color='r', linestyle=' ', marker='s',ms=10)
        k={}
        for i in range(0,len(res1)):
           xv=float(res1[i])
           yv=float(res2[i])
           if (xv,yv) not in k.keys():
              k[xv,yv]=c[i]
           elif (xv,yv) in k.keys():
              k[xv,yv]= k[xv,yv]+','+c[i]
        plt.annotate(k[xv,yv], xy=(res1[i], res2[i]), xytext=(res1[i]+0.01,
res2[i]+0.01),arrowprops=dict(facecolor='red', shrink=0.1),)
          plt.axis([min(res3[0])-e3, max(res3[0])+e3, min(res4[0])-e4,
max(res4[0])+e4])
                  plt.plot(res3[0],  res4[0],  color='b',  linestyle='  ',
marker='o',ms=10)
        k={}
        for i in range(0,len(doc)):
           xv=float((res3[0])[i])
           yv=float((res4[0])[i])
           if (xv,yv) not in k.keys():
              k[xv,yv]=i
           elif (xv,yv) in k.keys():
              k[xv,yv]= k[xv,yv]+','+str(i)
                  plt.annotate(k[xv,yv],  xy=((res3[0])[i],  (res4[0])[i]),
xytext=((res3[0])[i]+0.015, (res4[0])[i]+0.015),arrowprops=dict(faceco
lor='blue', shrink=0.1),)
      plt.grid()
          plt.show()
```

4. Сравните графические результаты усовершенствованной программы с результатами, полученными в работе [29], как это уже делалось при сравнении исходных матриц (см. п. 7 практической работы № 4). Для этого используйте setting.py, приведенный в п. 7 практической работы № 4. Далее отключите введенные усовершенствования, которые только предлагаются в работе [29], –

это TF-IDF преобразование исходной матрицы, причём не нужно отключать исключение документов, не содержащих ключевых слов, так как в приведенном setting.py нет документов, которые не содержат ключевых слов. Чтобы осуществить отключение нормализации исходной матрицы в процедуре-функции **def Analitik_Matrix(A,c,t)**, нужно закомментировать одну строку и добавить другую:

```
#return TF_IDF(A,c,t)
return U_S_Vt(A,c,t)
```

Дополнительно для исключения влияния процедуры-функции **def TF_IDF(A,c,t)** закомментируйте её последнюю строку:

```
#return U_S_Vt(A,c,t)
```

Для сравнения сначала рассмотрим результаты графического анализа, приведенные в статье [29] (рис. 1).

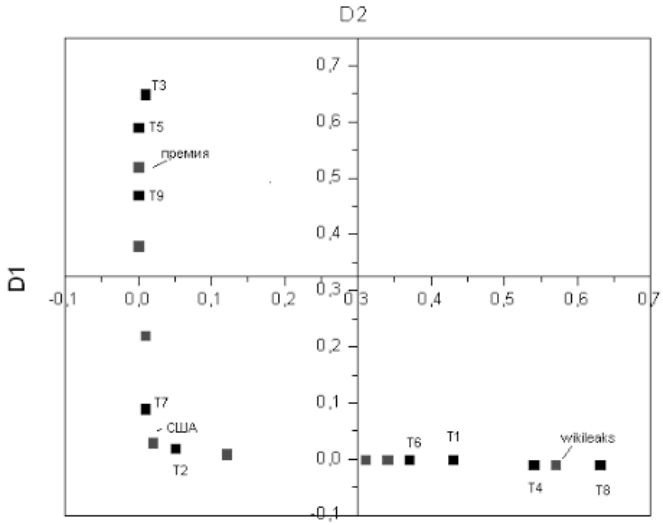

Рис. 1. Семантическое пространство слов и статей из источника [29]

Согласно утверждению автора упомянутой статьи, на основе графического анализа делается вывод о том, что документы (или статьи, как в [29]) образуют **три независимые группы**. Первая группа статей располагается рядом со словом **«wikileaks»**. И действительно, если мы посмотрим названия этих статей, становится понятно, что они имеют

отношение к **wikileaks**. Другая группа статей объединяется вокруг слова **«премия»**. И действительно, в них идет обсуждение Нобелевской премии. Хотя о третьей группе статей и слов автор ничего не сообщает, но, судя по графику, эта группа объединяется вокруг сокращённого названия страны – **«США»**.

После запуска нашей программы получим управляемый графический объект с изображением семантического пространства в двухмерном представлении (рис. 2).

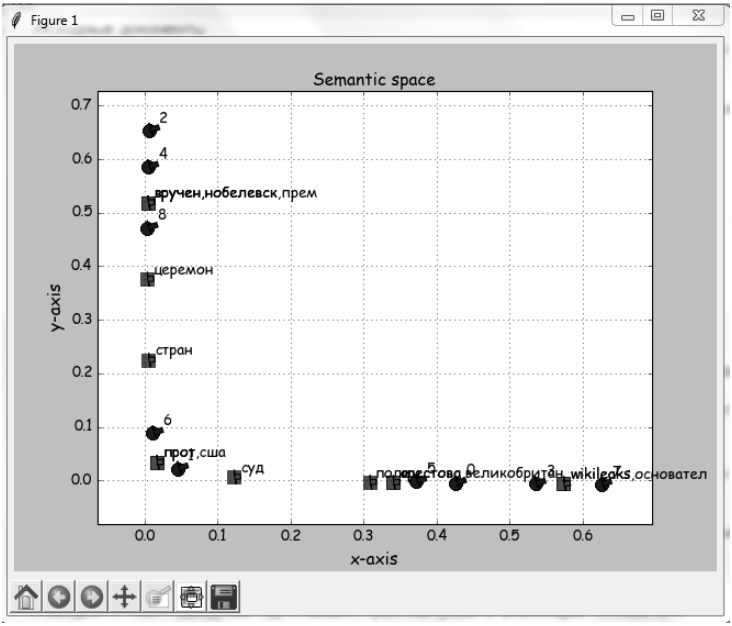

Рис. 2. Ссемантическое пространство слов и статей, построенное нашей программой по исходным данным источника [29]

Прежде чем проводить сопоставительный анализ, рассмотрим возможности управления графиком рис. 2. Палитра инструментов в левом нижнем углу справа налево: 1 – save the figure, 2 – Cofigure subplots, 3 – Zoom to restangle, 4 – Pan axes with left mouse, zoom with right, 5 – Format to next view, 6 – Back to previous view, 7 – Reset original view.

Сопоставляя графики на рис. 1 и 2, видим, что они идентичны, с той лишь разницей, что на рис. 2 приведены все слова, и нумерация статей начинается не с нуля, а с единицы. Однако чтение

некоторых слов и номеров документов первой группы затруднено. Этот недостаток устраняет инструмент **3 – Zoom to restangle**, который позволяет проводить детальный просмотр каждой группы в отдельности, например, для первой группы (рис. 3).

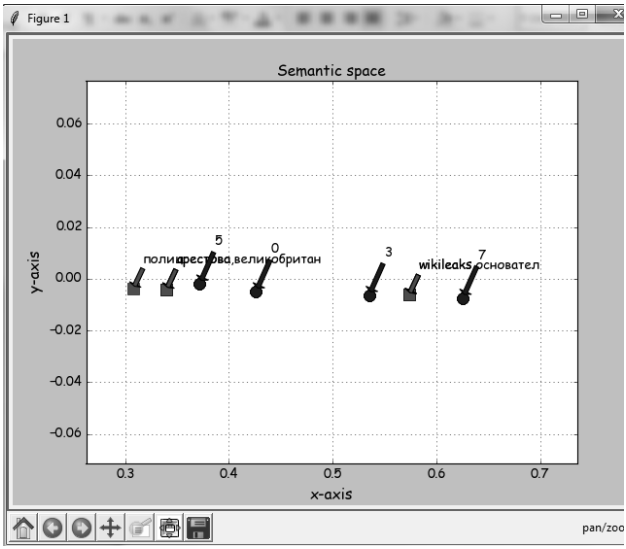

Рис. 3. Ссемантическое пространство слов и статей первой группы, построенное нашей программой по исходным данным источника [29]

Из рис. 3 следует, что статьи № 0, 3, 5, 7 семантически близки и, действительно, имеют общий смысл, все они имеют отношение к **wikileaks.**

5. Применим TF-IDF преобразование исходной матрицы, задекларированное автором статьи [29; 31], с целью совершенствования алгоритма. Для этого отменим сделанные ранее изменения, получим график (рис. 4). Для удобства анализа приведём распечатку из нашей программы состава преобразованных документов, количество которых совпадает с исходными:

Результаты анализа: Всего документов: 9. Осталось документов после исключения не связанных: 9

Ном.док--0 Текст-Британская полиция знает о местонахождении основателя WikiLeaks

Ном.док--1 Текст-В суде США начинается процесс против россиянина, рассылавшего спам

Ном.док--2 Текст-Церемонию вручения Нобелевской премии мира бойкотируют 19 стран

Ном.док--3 Текст-В Великобритании арестован основатель сайта Wikileaks Джулиан Ассандж

Ном.док--4 Текст-Украина игнорирует церемонию вручения Нобелевской премии

Ном.док--5 Текст-Шведский суд отказался рассматривать апелляцию основателя Wikileaks

Ном.док--6 Текст-НАТО и США разработали планы обороны стран Балтии против России

Ном.док--7 Текст-Полиция Великобритании нашла основателя WikiLeaks, но, не арестовала

Ном.док--8 Текст-В Стокгольме и Осло сегодня состоится вручение Нобелевских премий

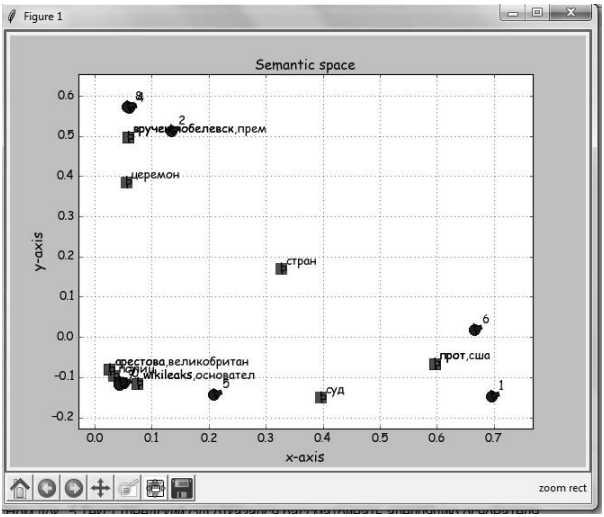

Рис. 4. Семантическое пространство слов и статей, построенное нашей программой по исходным данным источника [29] после нормализации исходной частотной матрицы методом TF-IDF

На основе анализа рис. 4, в сравнении с рис. 1 и 2, можно сделать очень важный вывод – **количество групп сохранилось**, но они переформатировались и **локализовались в семантическом**

пространстве отделив слово **«стран»** и собрав вместе **статьи № 6, 1** и слова **«против», «США».** Действительно, анализ смысла документов **№ 6, 1** (см. распечатку) показывает, что метод TF-IDF нормализации позволяет выделить новые логические связи слов и документов.

6. Программная среда Python при установке её на персональном компьютере может использоваться для эксплуатации написанного на ней программного обеспечения без компиляции, это придает таким программам дополнительную гибкость. Одним из примеров такой гибкости является адаптация к исходным данным и обработке согласно [31], приведенная в п. 4. Гибкость также обеспечивается лёгкой сменой набора анализируемых документов. Для этого в той же папке, что и основная программа menu.py, можно разместить несколько файлов конфигурации, например setting.py, setting_1.py…….. setting_n.py. Краткое описание всех файлов конфигурации с их именами можно поместить в текстовом файле setting.txt, который можно открывать в одном из полей основной программы menu.py.

Вопросы к практической работе № 6

1. Перечислите, какие этапы LSA реализованы в данной работе на Python.

2. Как в процедуре-функции **def Grafics_End(res1,res2,res3,res4,c)** реализована возможность подписать на графике одну точку несколькими надписями? Приведите только данный код процедуры.

3. В функции **def Grafics_End(res1,res2,res3,res4,c)** укажите переменные, отвечающие за: размер отметки на графике; цвет отметки на графике; форму отметки на графике.

4. Что и из какой процедуры передаётся в функцию **def Grafics_End**? Опишите каждую переменную.

5. Для чего в процедуре **def Grafics_End** используются переменные e1, e2, e3, e3, приведите их числовые значения для полученного Вами графика семантического пространства.

6. Какая функция наносит на график сетку? Напишите функцию.

7. В чём состоит гибкость программного обеспечения, работающего в среде Python без компиляции? Приведите примеры изменения кода.

8. Как изменятся графические результаты автоматизированного семантического анализа при смене частотной матрицы с ненормализованной на нормализованную? Приведите пример со своими данными файла конфигурации с выделением и анализом групп документов и слов.

9. На основе какого принципа выделяют на графике семантического пространства группы слов и документов?

10. Опишите функции панели инструментов графического объекта.

11. Какие части программного кода процедуры **def Grafics_ End** отвечают за вывод **стрелок сносок на графике** при использовании инструмента **Zoom to restangle**? Приведите части кода.

12. Какие модули графического пакета **matplotlib** предназначены для записи кириллических символов?

Задания к практической работе № 6

1. Выполните все пункты практической работы № 6, создайте скриншоты или распечатки результатов с пояснениями, скопируйте текст основного модуля menu.py, включая последнюю процедуру, занесите их в отчёт, причём сделайте скриншоты распечаток для каждого из файлов settings.py.

2. Дайте подробные ответы на вопросы и занесите их в отчёт по практической работе № 6.

3. Доработайте программу menu.py для загрузки содержимого файла setting.txt в первое текстовое поле формы menu.py. Функцию загрузки занести в меню.

4. Оформите отчет по схеме, указанной в практической работе № 1.

ЛИТЕРАТУРА

1. Гальперин И.Р. Текст как объект лингвистического исследования / И.Р. Гальперин. – М.: Наука, 1981. – 139 с.

2. Тураева З.Я. Лингвистика текста: (структура и семантика) / З.Я. Тураева. – М.: Просвещения, 1986. – 127 с.

3. Кухаренко В.А. Интерпретация текста / В.А. Кухаренко. – М.: Просвещение, 1986. – 205 с.

4. Столярова Л.П. Базовый словарь лингвистический терминов / Л.П. Столярова, Т.С. Пристайко, Л.П. Попко. – К.: изд-во Государственной академии руководящих кадров культуры и искусств, 2003. – 192 с.

5. Качалова К.Н. Практическая грамматика английского языка с упражнениями и ключами / К.Н. Качалова, Е.Е. Израилевич. – М.: ЮНВЕС ЛИСТ, 1997. – 717 с.

6. Верба Л.Г. Граматика сучасної англійської мови: довідник / Л.Г. Верба, Г.В. Верба. – К.: ТОВ «ВП ЛОГОС», 2002. – 352 с.

7. Hurford J.R. (1994). Grammar. A student`s guide – Cambridge University Press, 1994. – 271 p.

8. Murphy R. English Grammar in Use. – Cambridge University Press, 1995. – 350 p.

9. Рудяков Н.А. Основы стилистического анализа художественного произведения / Н.А. Рудяков. – Кишинёв: Штиинца, 1972. – 179 с.

10. Бабенко Л.Г. Лингвистический анализ художественного текста. Теория и практика: учебник; практикум / Л.Г. Бабенко, Ю.В. Казарин. – М.: Флинта: Наука, 2004. – 496 с.

11. Новиков Л.А. Художественный текст и его анализ: учеб. пособие / Л.А. Новиков – М.: ЛКИ, 2007. – 304 с.

12. Автоматический анализ текстов на естественном языке [Электронный ресурс] / Е.В. Щуревич, Е.Н. Крючкова; кафедра прикладной математики, Алтайский государственный технический университет. – Режим доступа: \www/ URL: http:// http://math.nsc.ru/conference/zont09/reports/43Schurevich-Kryuchkova.pdf/

13. Автоматическая обработка текста [Электронный ресурс]. – 2003. – Режим доступа : \www/ URL: http://www.aot.ru

14. Кашткин Н.Н. Численные методы / Н.Н. Кашткин. – М.: Наука, 1978. – 512 с.

15. Лебедев И.С. Построение семантически связанных информационных объектов текста / И.С. Лебедев // Прикладная информатика. – 2007. – № 5 (11). – С. 54–61.

16. Bird S. Introduction to Natural Language Processing / S. Bird, E. Klein, E. Loper, University of Pennsylvania. – 2001–2007.

17. Язык программирования Python [Электронный ресурс]: учебник / Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач [и др.]. – Режим доступа: https://rtfm.co.ua/uploads/python_rossum.pdf/

18. Язык программирования Python [Электронный ресурс]: курс лекций / Р.А. Сузи. – Режим доступа: \www/ URL: http://www.docme.ru/doc/7427/r.-a.-suzi---yazyk-programmirovaniya-python/ - Загл. с экрана.

19. Mertz D. Text Processing in Python / D. Mertz, A. Wesley. – 2003. – 544 p.

20. О программировании, алгоритмах и не только. – Стоп-символы русского языка [Электронный ресурс]. – Режим доступа: http://www.algorithmist.ru/2010/12/stop-symbols-in-russian.html

21. О программировании, алгоритмах и не только. – Стеммер Портера для русского языка [Электронный ресурс]. – Режим доступа: \www/ URL: http://www.algorithmist.ru/2010/12/porter-stemmer-russian.html

22. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/ wiki/TF-IDF

23. Jones K.S. (2004). A Statistical interpretation of term specificity and its application in retrieval / K.S. Jones // Journal of Documentation. – University: MCB University Press, 2004. – Vol. 60, № 5. – P. 493–502.

24. William H. Singular Value Decomposition / William H. // Numerical Recipes in C. – 2nd edition. – Cambridge: Cambridge University Press.

25. Code, Manage, Collaborate [Электронный ресурс]. – Режим доступа: https://bitbucket.org/pcarbonn/pipwin/downloads

26. NLTK 3.0 documentation, Source code for nltk.stem.snowball [Электронный ресурс]. – Режим доступа: http://www.nltk.org/_modules/nltk/stem/snowball.html

27. Модуль tkinter. Создание графического интерфейса [Электронный ресурс]. – Режим доступа: http://kabinet-vplaksina.narod.ru/olderfiles/5/Modul_tkinter.pdf

28. Хабрахабл, Edunov Латентно-семантический анализ [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/110078/—

29. Потемкин В.Г. Справочник по MATLAB. Линейная алгебра. [Электронный ресурс] / В.Г. Потемкин – Режим доступа: http:\www.matlab.exponenta.ru/ml/bock2/chapter7/svd/php

30. Taranenko Y.K Development of a computer system for generating semantic template of a group of documents by using latent semantic analysis. / Y.K. Taranenko, M.R. Kabanova // Eastern-Europeean Journal of Enterpise Technologies. – 2016. – Vol. 4, № 2, (82).

Навчальне видання

Тараненко Юрій Карлович
Кабанова Марина Романівна
Черняк Наталія Олександрівна

КОМП'ЮТЕРНА ЛІНГВІСТИКА

Навчальний посібник

(російською мовою)

Редактор М.С. Велес
Комп'ютерна верстка А.Ю. Такій

Підписано до друку 3.11.2016. Формат 60×84/16.
Ум. друк. арк. 6,51. Тираж 100 пр. Зам. № .

ПВНЗ «Дніпропетровський університет імені Альфреда Нобеля».
49000, м. Дніпро, вул. Січеславська Набережна, 18.
Тел. (056) 778-58-66, e-mail: rio@duep.edu
Свідоцтво ДК № 4611 від 05.09.2013 р.

Віддруковано у ТОВ «Роял Принт».
49052, м. Дніпро, вул. В. Ларіонова, 145.
Тел. (056) 794-61-05, 04
Свідоцтво ДК № 4765 від 04.09.2014 р.