DOI: https://doi.org/10.15688/jvolsu2.2017.2.6

UDC 81'33
LBC 81.1

Submitted: 10.03.2017
Accepted: 11.05.2017

ALGORITHM OF KEY WORDS SEARCH BASED ON GRAPH MODEL OF LINGUISTIC CORPUS

Elena G. Grigoryeva
Volgograd State University, Volgograd, Russian Federation

Vladimir A. Klyachin
Volgograd State University, Volgograd, Russian Federation

Yuriy V. Pomelnikov
Volgograd State University, Volgograd, Russian Federation

Vladimir V. Popov
Volgograd State University, Volgograd, Russian Federation

Abstract. One of the problems of computer corpus linguistics is an automatic determination of keywords inthe text. The solution is a statistical method based on calculation of various frequency characteristics of the text. In this case, the most commonly used model is a "bag of words", which does not take into account the order of words in the text. In this paper, we propose a graph model of the text that allows us to calculate the frequency characteristics of words in the text not only within the framework of the "word bag" model, but with respect to location of pairs of owls in some common part of the text, for example, in one sentence. To work with such a model, a software model is constructed in the form of a database schema intended for storing various statistical text information. Taking into account such a data model, the article proposes an algorithm for determining the keywords of the text, the implementation of which is performed in the Python programming language.

When analyzing a document d of linguistics corpus D, our algorithm creates a list of about 40 words with the largest measure tf-idf, and choise from them 20 words, which are more often used in the document d. We regard these words as vertices of some graph G, and the multiplicity of the edge, connecting the vertices t and t' is equal to the number of sentences in document d, containing both these words. Approximately 10 vertices of the graph with the greatest degree are selected. The words corresponding to these vertices are taken for key words of document d.

Key words: graph, text, word, text split, statistic measure tf-idf, key word, base form of word.

Citation. Grigoryeva E.G., Klyachin V.A., Pomelnikov Yu.V., Popov V.V. Algorithm of Key Words Search based on Graph Model of Linguistic Corpus. *Vestnik Volgogradskogo gosudarstvennogo universiteta. Seriya 2, Yazykoznanie [Science Journal of Volgograd State University. Linguistics]*, 2017, vol. 16, no. 2, pp. 58-67. (in Russian). DOI: https://doi.org/10.15688/jvolsu2.2017.2.6.

УДК 81'33
ББК 81.1

Дата поступления статьи: 10.03.2017
Дата принятия статьи: 11.05.2017

АЛГОРИТМ ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ НА ОСНОВЕ ГРАФОВОЙ МОДЕЛИ ЛИНГВИСТИЧЕСКОГО КОРПУСА

Елена Геннадиевна Григорьева
Волгоградский государственный университет, г. Волгоград, Российская Федерация

Владимир Александрович Клячин
Волгоградский государственный университет, г. Волгоград, Российская Федерация

Юрий Вячеславович Помельников
Волгоградский государственный университет, г. Волгоград, Российская Федерация

Владимир Валентинович Попов
Волгоградский государственный университет, г. Волгоград, Российская Федерация

Аннотация. Одной из задач компьютерной корпусной лингвистики является задача автоматического определения ключевых слов текста. Основные методы решения данной задачи, будучи статистическими, базируются на вычислении различных частотных характеристик текста. При этом чаще других используется модель «мешка слов», которая не учитывает порядок следования слов в тексте. В статье предлагается графовая модель текста, позволяющая вычислять частотные характеристики слов текста не только в рамках модели «мешка слов», но и с учетом расположения пар слов в какой-либо общей части текста, например в одном предложении. Для работы с такой моделью построена программная модель в виде схемы базы данных, предназначенной для хранения различной статистической информации текста. С учетом такой модели данных в статье предложен алгоритм определения ключевых слов текста, реализация которого выполнена на языке программирования Python.

При анализе текста из него сначала выделяется приблизительно 40 слов с наибольшей мерой tf-idf, а из них – 20 слов, которые чаще других употребляются в данном тексте. Эти слова рассматриваются как вершины некоторого графа G, причем кратность ребра, соединяющего вершины t и t', равна числу предложений в тексте d, содержащих оба этих слова. Далее выбираются приблизительно 10 вершин графа наибольшей степени. Слова, соответствующие этим вершинам, и принимаются за ключевые слова данного текста.

Ключевые слова: граф, текст, слово, разбиение текста, статистическая мера tf-idf, ключевое слово, базовая форма слова.

Цитирование. Григорьева Е. Г., Клячин В. А., Помельников Ю. В., Попов В. В. Алгоритм выделения ключевых слов на основе графовой модели лингвистического корпуса // Вестник Волгоградского государственного университета. Серия 2, Языкознание. – 2017. – Т. 16, № 2. – С. 58–67. – DOI: https://doi.org/10.15688/jvolsu2.2017.2.6.

1. Введение

В работе предложена математическая модель аннотированного лингвистического корпуса на основе графа отношений принадлежности слов семантическим или синтаксическим единицам текста. Из весовых значений вершин и ребер этого графа могут быть извлечены статистические данные, позволяющие решать различные актуальные задачи лингвистики: извлечение ключевых слов, оценка сочетаемости слов, тематическое моделирование лингвистических корпусов, построение конкордансов и т. д. Кроме того, создана программная модель графа отношений принадлежности в виде схемы базы данных.

При решении лингвистических задач обработки текста возможны различные подходы и методы преобразования текстовой информации в наборы числовых данных. Так, текстовую информацию можно преобразовать в наборы векторов в многомерном пространстве (процесс векторизации). Эти векторы нумеруют единицы текста (термы) по отношению к единицам корпуса документов и положению терма в самом документе. В дальнейшем эта информация может быть использована при решении задач классификации, поиска различий и сходств, а также для получения других характеристик.

Значимой во многих алгоритмах обработки текста является мера tf-idf, указывающая на частоту термов в документе и уникальность термов для данного документа. В библиотеке PythonScikitLearn вычисление tf-idf производится в несколько шагов с помощью ScikitLearn'sTfidfVectorizer, что эквивалентно использованию TfidfTransformer вслед за CountVectorizer.

Частота термина tf вычисляется методом fit_transform() класса CountVectorizer; обратная частота idf – методом fit() класса TfidfTransformer; величина tf-idf – методом transform() класса TfidfTransformer.

В результате вычислений можно создать матрицу свойств размерности n*m, где n – количество терминов, а m – размер корпуса, то есть число документов в этом корпусе. В некоторых алгоритмах требуется нормализованная матрица tfidf (тогда можно, например, при-

менить метрику сходства векторов cos(u, v), что отражает расстояние между двумя векторами с концами на единичной сфере). Такую матрицу можно получить путем изменения параметров при обращении к соответствующей процедуре.

В некоторых случаях меру tf-idf целесообразно модифицировать, применив к ней процедуру сглаживания (Smoothidfweights). TfidfVectorizer, после чего веса термов окажутся на интервале числовой прямой с концами 0 и 1. Это соответствует добавлению 1 к частоте документов, как будто существует еще один документ, содержащий каждое слово ровно по одному разу. Таким образом, формула tf · idf для вычисления tf-idf заменяется формулой tf · (idf + 1).

При использовании статистических методов анализа документа для каждого слова, встречающегося в этом документе, определяется базовая, каноническая форма этого слова (лемма), а также набор его грамматических характеристик. Постановку слова в каноническую форму называют нормализацией.

Многие программы по автоматизированной обработке текста составляются на языке Python. Для морфологической обработки русскоязычных текстов часто используется морфологический анализатор Pymorphy2. Например, необходимо проанализировать слово *дождями*. Для этого можно использовать такие операторы:

```
import pymorphy2
morph=pymorphy2.MorphAnalyzer()
sp=morph.parse("дождями")
p=sp[0]
print(p)
```

Первый оператор подключает к программе модуль Pymorphy2, а второй позволяет в дальнейшем обращаться к методу MorphAnalyzer(), используя краткое имя "morph". Третий оператор формирует список sp гипотез, каждая из которых соответствует одному из возможных вариантов разбора данного слова. Число гипотез можно найти с помощью оператора len(sp), а гипотезу с номером i выделяют с помощью оператора p=sp[i]. Четвертый оператор выделяет наиболее вероятную гипотезу p с номером 0 (нумерация гипотез начинается с нуля). Выписанный выше фрагмент программы выдает такой результат:

```
Parse(word='дождями', tag=OpencorporaTag
('NOUN,inan,masc,plur,ablt'), normal_form='дождь',
score=1.0, methods_stack=((<DictionaryAnalyzer>,
'дождями', 190, 10),))
```

Теперь с помощью операторов

```
nw=p.normal_form
sp_g =p.tag
```

можно получить базовую форму (лемму) nw='дождь' слова w='дождями' и тег sp_g, то есть набор его грамматических признаков. В данном случае sp_g – это символьная строка 'NOUN,inan,masc,plur,ablt', в которой указано, что 'дождями' – это существительное (NOUN) неодушевленное (inan), мужского рода (masc) во множественном числе (plur) и творительном падеже (ablt). Более подробно с возможностями модуля Pymorphy2 можно ознакомиться, например, по адресу http://pymorphy2.readthedocs.io/en/latest/user/.

Если текст документа хранится в строке с именем s, то для получения списка слов sp_word в этой строке, составленных из букв кириллического алфавита, можно использовать, например, оператор

```
sp_word=re.findall('[А-Яа-яё]+',s).
```

В этом операторе используется регулярное выражение (шаблон) '[А-Яа-яё]+'. Для работы с регулярными выражениями необходимо подключить к программе модуль re, что достигается в результате выполнения оператора

```
import re.
```

2. Графовая модель лингвистического корпуса

Алфавитом будем называть произвольное конечное множество Σ. Элементы множества Σ будем называть символами. Упорядоченный набор символов назовем словом или цепочкой символов. Множество всех цепочек символов алфавита Σ обозначим через Σ^*. Текстом будем называть упорядоченный набор символов, записывать его будем в виде

$$T = a_1 a_2 \ldots a_n,\ a_i \in \Sigma,\ n = |T|.$$

Если T_1, T_2 – два текста, то через $T_1 \cdot T_2$ будем обозначать текст, склеенный из двух данных текстов (конкатенация текстов). Здесь и далее через $|X|$ обозначим мощность множества X – количество элементов множества X. Через 2^X будем обозначать множество всех подмножеств множества X. Некоторую совокупность текстов будем называть корпусом. Для построения графовой модели нам необходимо понятие разбиения текста. Под разбиением текста будем понимать произвольное подмножество

$$P \subset 2^N, N = \{1,2,\ldots,|T|\}$$

упорядоченных подмножеств множества N. Каждое такое подмножество определяется набором номеров $(i_1, i_2, \ldots, i_k) \in P$ входящих в него символов. Примерами таких разбиений могут быть разбиение текста на слова, разбиение текста на предложения и разбиение текста на m-граммы.

Пусть $I \in P$. Построим отображение $\omega : P \to \Sigma^*$,

$$\omega(I) = a_{i_1} \ldots a_{i_k},$$

где $i_1 < i_2 < \Lambda < i_k$, $i_j \in I$, $j = \overline{1,k}$.

Это отображение сопоставляет набору номеров символов соответствующую часть текста в виде цепочки символов.

Пример 1. Зафиксируем произвольный символ $s \in \Sigma$. Для заданного текста T пусть найдутся номера $1 = i_0 \le i_1 < i_2 < \Lambda < i_k < i_{k+1} = |T|$ такие, что $a_{i_j} = s$, $j = \overline{1,k}$. Тогда

$$P_s = \{(i_0,\ldots,i_1-1),(i_1+1,\ldots,i_2-1), \ldots,(i_k+1,\ldots,i_{k+1})\}$$

представляет собой разбиение текста с помощью разделителя $S \in \Sigma$.

Пример 2. Для заданного разбиения вида P_s и произвольного натурального m можно построить разбиение вида

$$P_{s,m} = \{\omega_1 \cup \ldots \cup \omega_m, \omega_2 \cup \ldots \cup \omega_{m+1}, \ldots, \omega_{k+m-2} \cup \ldots \cup \omega_{k+1}\},$$

где $\omega_j = (i_{j-1}+1,\ldots,i_j-1), j = 1, \ldots, k,$ $\omega_{k+1} = (i_{k+1}+1,\ldots,i_{k+1})$. Данное разбиение представляет собой разбиение m-грамм.

По определению будем считать, что разбиение P' является более мелким, чем разбиение P'', и записывать $P' \subset P''$, если для всякого $I' \in P'$ найдется такое $I'' \in P''$, что $I' \subset I''$. Для двух заданных текстов T_1, T_2 с соответствующими разбиениями P_1, P_2 определим разбиение $P_1 \cdot P_2$ для текста $T_1 \cdot T_2$ следующим образом:

$$P_1 \cdot P_2 = \{I \subset \{1,2,\ldots,|T_1|+|T_2|\} : I = I_1 \in P_1. \text{ или } I = \{i+|T_1|, i \in I_2 \in P_2\}\}.$$

Несложно проверить, что совокупность всех текстов и их разбиений (T, P) образует полугруппу с групповой операцией

$$(T_1, P_1) + (T_2, P_2) = (T_1 \cdot T_2, P_1 \circ P_2).$$

Для заданного текста T и его разбиения P обозначим через $U(T, P) = \{\omega(I): I \in P\}$ совокупность уникальных частей разбиения. Например, в случае разбиения текста на слова это множество представляет собой совокупность уникальных слов текста. Рассмотрим текст T и два его произвольных разбиения $P' \subset P''$. Построим граф $G = G(T, P', P'')$ со множеством вершин $VG = U(T, P')$ и множеством ребер

$$EG = \{(a,b): a,b \in U(T,P'), \text{ если для } I_a, I_b \in P', \omega(I_a) = a, \omega(I_b) = b \text{ существует } I \in P'', \text{ такое, что } I_a, I_b \subset I\}$$

Этот граф представляет отношение принадлежности пары частей текста одного разбиения одной части другого разбиения. Например, ребру графа с вершинами в виде уникальных слов текста соответствует пара слов, которые встречаются в каком-либо одном предложении. Непосредственно можно проверить следующую ключевую формулу для построения графа при склеивании двух текстов. Пусть имеются тексты T_1, T_2 с соответствующими разбиениями $P'_i \subset P''_i$. Тогда имеет место равенство

$$G(T_1 \cdot T_2, P_1' \circ P_2', P_1'' \circ P_2'') = G(T_1, P_1', P_1'') \cup G(T_2, P_2', P_2'').$$

Ниже при описании алгоритма выделения ключевых слов текста мы покажем, каким образом будет использован построенный граф в этой задаче.

Рассмотрим некоторую аддитивную полугруппу W. Вес для графа G – это отображение, сопоставляющее каждой вершине графа или каждому ребру графа значения из полугруппы W

$$w: VG \to W \text{ или } w: EG \to W.$$

Введение весовой функции для построенного графа объясняется тем, что в корпусной лингвистике распространены статистические методы решения разнообразных задач. Весовая функция и призвана производить вычисление соответствующих статистических величин.

Пример 3. Пусть $W = \mathrm{N}$ – множество натуральных чисел. Для $v \in VG, G = G(T, P', P'')$ положим

$$w'(v) = |\omega'^{-1}(v)|,$$

$$w''(v) = |\omega''^{-1}(v)|.$$

Первая функция вычисляет количество вхождений уникальной части текста v разбиения P' в тексте T, а вторая функция вычисляет количество частей текста разбиения P'', содержащих v.

На множестве подмножеств множества z_m введем операцию +

$$I_1 + I_2 = sort(I_1 \cup I_2), I_1, I_2 \in 2^m$$

как результат композиции объединения и сортировки. Тогда можно положить

$$w^m(v) = \omega''^{-1}(v).$$

Данная весовая функция строит упорядоченные списки номеров частей текста разбиения P'', содержащие уникальную часть текста v разбиения P'. Для двух текстов T_1, T_2 и соответствующих весовых функций $w_i : VG(T_i, P_i', P_i'') \to W$ определим весовую функцию

$$w(v) = \begin{cases} w_1(v), \text{если } v \in U(T_1, P_1') \setminus U(T_2, P_2'), \\ w_2(v), \text{если } v \in U(T_2, P_2') \setminus U(T_1, P_1'), \\ w_1(v) + w_2(v), \text{если } v \in U(T_1, P_1') \cap U(T_2, P'_2). \end{cases}$$

Эта формула позволяет вычислять весовую функцию для результата склейки двух текстов при условии, что значения весовых функций двух текстов принадлежат одному множеству. Получим также формулу для случая, когда эти множества разные.

Пусть m_1, m_2 – два натуральных числа. Ясно, что $2^{Z_{m_1}} \subset 2^{Z_{m_1}+Z_{m_2}}$. Пусть $I_1 \in 2^{Z_{m_1}}, I_2 \in 2^{Z_{m2}}$. Положим

$$I_1 \oplus I_2 = \{(i_1, \ldots, i_k, j_{1+m_1}, \ldots, j_{l+m_1}) : \\ : (i_1, \ldots, i_k) = I_1, (j_1, \ldots, j_l) = I_2\}.$$

Непосредственным вычислением проверяется свойство ассоциативности

$$I_1 \oplus (I_2 \oplus I_3) = (I_1 \oplus I_2) \oplus I_3).$$

Приведем пример еще одной весовой функции, которая может иметь определенное значение при исследовании лингвистического корпуса. Предположим, что имеются два текста T_1, T_2 и две весовые функции

$$w_i : VG(T_i, P_i', P_i'') \to W_i,$$

где $W_i = 2^{m_i}$, $m_i = |T_i|$. Требуемую функцию $w: VG(T_1 \cdot T_2) \to W = 2^{m_1+m_2}$ построим следующим образом:

$$w(v) = \begin{cases} w_1(v), \text{если } v \in U(T_1, P_1') \setminus U(T_2, P_2'), \\ \varnothing \oplus w_2(v), \text{если } v \in U(T_2, P_2') \setminus U(T_1, P_1'), \\ w_1(v) \oplus w_2(v), \text{если } v \in U(T_1, P_1') \cap U(T_2, P_2'). \end{cases}$$

Наконец приведем пример весовой функции для ребер графа $G = G(T, P', P'')$. Обозначим $m = |T|$. Пусть $e = (v_1, v_2) \in EG$. Примерами весовых функций могут быть функции

$$w(e) = |w^m(v_1) \cap w^m(v_2)|,$$

или

$$w(e) = w^m(v_1) \cap w^m(v_2).$$

Первая функция вычисляет количество вхождений пары (v_1, v_2) в одну часть разбиения P'', вторая функция перечисляет все такие части разбиения P''. Заметим, что эти функции могут быть вычислены через соответствующие весовые функции вершин графа.

Для организации хранения текстов и соответствующей статистической информации, вычисляемой по вышеприведенным формулам, предлагается схема базы данных. Ниже мы даем ее краткое описание и приводим соответствующую ER-диаграмму (Entity Relation Diagram) (см. рис. 1). Сама база данных может быть развернута на базе промышленного сервера PostgreSQL либо в локальной настольной версии на базе легковесной СУБД SQLite. Структура базы данных – это 6 основных таблиц, связанных между собой по типу родительско-дочерней связи. Опишем кратко поля этих таблиц. Таблица Corpus предназначена для хранения информации о корпусе: идентификатор корпуса (id_corpus), создатель корпуса (id_user), название корпуса (title) и его описание (description). По мере необходимости возможно добавление дополнительных полей (например, дата создания корпуса и др.). Таблица Users содержит информацию о пользователях системы: идентификатор (id_user), логин (login_user), пароль (password_user) и поле для дополнительной информации (information). Таблица Documents является основной таблицей для хранения собственно текстов (content), входящих в корпус, ключевых слов (key_words). Здесь же задается тип текста (публицистика, документ и пр.) как идентификатор-ссылка (id_type) на таблицу с типами Content_type. В этой же таблице сохраняется путь к файлу с текстом – поле file_name. Таблица Authors необходима для получения информации об авторах текстов. Последняя из таблиц – Edges – хранит сведения о словосочетаниях word1+word2 в заданном тексте (id_text), причем хранится не само слово, а его номер в предложении (sentence). Построение базы данных на основе реляционной СУБД позволит использовать встроенный полнотекстовый поиск, реализовывать некоторые лингвистические алгоритмы с помощью хранимых процедур и функций, а также упростить реализацию бизнес-логики системы.

3. Алгоритм выделения ключевых слов

В этой части статьи мы описываем алгоритм выделения ключевых слов текста. Данный алгоритм реализован на языке программирования Python. В дальнейшем планируется интегрировать реализацию этого алгоритма в общую информационную систему по обработке данных, схема которых была охарактеризована выше.

Описанные в разделе 2 структуры данных позволяют решать многие лингвистические задачи. В частности, с помощью этих структур можно извлекать ключевые слова из текстов. Задаче выделения ключевых слов посвящены многие публикации российских и иностранных авторов. Обзор результатов по этому направлению пред-

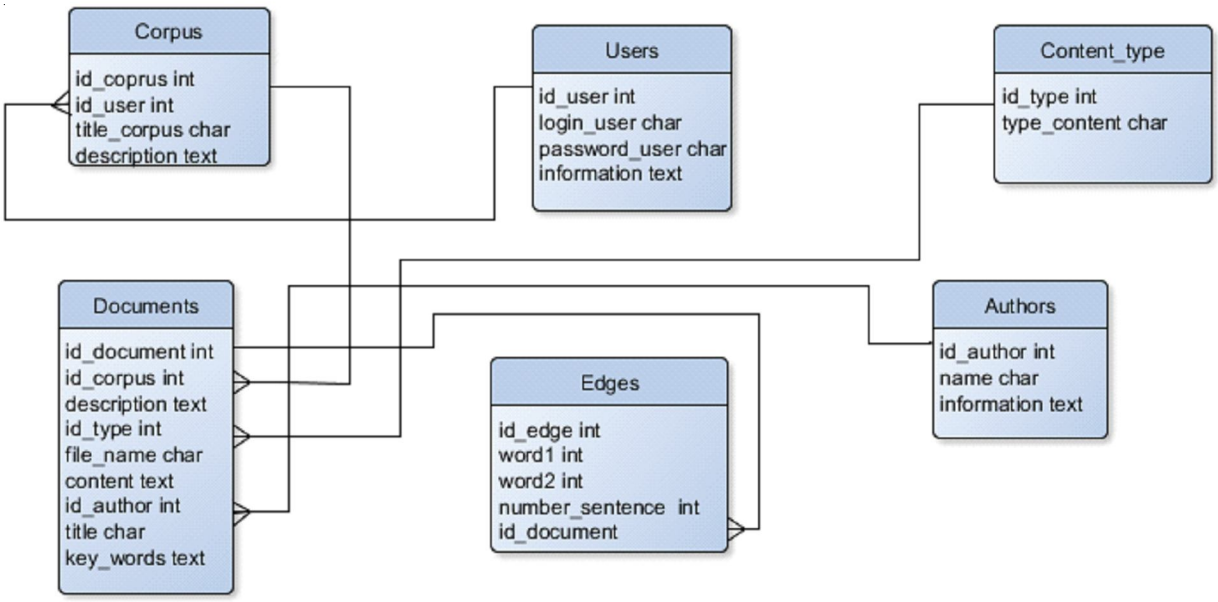

Рис. 1. ER-диаграмма базы данных

ставлен в работах Н.Б. Баканова [2013], А.С. Ванюшкина, Л.А. Гращенко [2016]. Предлагаемый ниже алгоритм выделения ключевых слов учитывает слова русского языка, удовлетворяющие двум условиям:

а) слово состоит не менее чем из 4 букв;

б) слово распознается морфологическим анализатором как существительное.

Алгоритм использует статистическую меру важности слова tf-idf в контексте документа d [Jones, 2004]. Пусть имеется коллекция документов D. Пусть t – слово и d – документ коллекции D. Тогда частота tf(t,d) слова t в документе d – это отношение числа употреблений этого слова в документе d к общему числу слов этого документа, а документальная частота df(t,D) – это число документов коллекции D, в которых встречается слово t. Обратная частота idf(t,D) слова t в коллекции D определяется формулой

$$idf(t, D) = \log \frac{|D|}{df(t, D)},$$

где |D| – число документов в коллекции D. Обычно логарифм берется по основанию 2, хотя выбор другого основания соответствует умножению чисел idf(t,D) на одно и то же число, что не меняет их взаимного расположения. Знаменатель дроби всегда отличен от нуля, поскольку мера tf-idf вычисляется только для слов t, встречающихся в документе d, а для таких слов $df(t,D) \geq 1$. Наконец, мера tf-idf(t,d,D) определяется формулой

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D).$$

Алгоритм требует предварительной обработки коллекции текстов D – исключения элементов форматирования, удаления стоп-слов и т. д.; кроме того, необходимо формирование словаря DICT, состоящего из лемм (базовых форм) слов t, встречающихся в документах коллекции D и удовлетворяющих условиям а и б.

Пусть d – документ коллекции D. Для выделения из него ключевых слов производим следующие действия:

1. Отбираем кандидатов в ключевые слова. С этой целью выделяем из документа d все слова t, входящие в словарь DICT, находим меры tf - idf(t,d,D) этих слов и формируем список List1, состоящий из C_1=40 слов с наибольшей мерой tf-idf.

2. Производим фильтрацию кандидатов в ключевые слова. С этой целью документ d разбиваем на предложения и для каждого слова t словаря списка List1 запоминаем список Sp(t) номеров предложений, в которых это слово употреблено. Число элементов списка Sp(t) пропорционально частоте tf(t,d) слова t в документе d. Выделяем из списка List1 приблизительно C_2=20 слова с наибольшей частотой tf(t,d) и формируем из них список List2.

3. Рассматриваем слова списка List2 как вершины некоторого графа G. Вершины t и t' этого графа соединены ребром, если есть предложение в документе d, содержащее оба этих слова. Кратность ребра, соединяющего вершины t и t', равна числу таких предложений. Выбираем из списка List2 C_3=10 слов, которые соответствуют вершинам графа G в наибольшей степени. Эти слова и принимаем за ключевые слова данного документа d.

Описанный алгоритм допускает модификации: вместо предложений можно использовать другие структурные единицы (например, m-граммы), а также изменять параметры C_1, C_2, C_3.

Алгоритм был опробован на модельном примере. В качестве коллекции документов был выбран набор из 20 текстов по различным научным направлениям (математический анализ, дифференциальные уравнения, теория графов, лингвистика). В таблице 1 указаны списки ключевых слов, полученные для одного текста по каждому из этих направлений.

Если вместо предложений рассматривать m-граммы, то списки ключевых слов могут измениться, хотя и незначительно. В таблице 2 приводятся списки ключевых слов для тех же текстов при m=5.

Наконец можно менять параметры C_1, C_2, C_3. Если выбрать $C_1 = 60$, $C_2 = 30$, $C_3 = 10$ и рассматривать n-граммы при m = 5, то получим списки, представленные в таблице 3.

Наблюдается определенная стабильность конечного списка слов при изменении параметров алгоритма.

Ниже приведены графы (см. рис. 2, 3), соответствующие документу по математическому анализу из таблицы 3.

Таблица 1

Список найденных слов при разбиении текста на предложения

Научное направление	Набор найденных ключевых слов
Математический анализ	*интеграл*, *функция*, *лебег*, *риман*, *площадь*, *мера*, *смысл*, *понятие*, *сумма*, *трапеция*
Дифференциальные уравнения	*уравнение*, *порядок*, *частное*, *функция*, *решение*, *теория*, *коэффициент*, *квадратура*, *простейшее*
Теория графов	*граф*, *вершина*, *ребро*, *метка*, *дуга*, *матрица*, *компонент*, *смежность*, *инцидентность*
Лингвистика	*предложение*, *знак*, *контекст*, *граница*, *препинание*, *анализ*, *алгоритм*, *документ*, *псевдослово*, *эвристика*

Таблица 2

Список найденных слов при разбиении текста на m-граммы при m = 5

Научное направление	Набор найденных ключевых слов
Математический анализ	*интеграл*, *функция*, *лебег*, *риман*, *смысл*, *площадь*, *мера*, *объем*, *трапеция*, *отрезок*
Дифференциальные уравнения	*уравнение*, *порядок*, *частное*, *решение*, *функция*, *теория*, *простейшее*, *коэффициент*, *входящая*
Теория графов	*вершина*, *граф*, *ребро*, *число*, *степень*, *маршрут*, *путь*, *метка*, *цикл*, *дуга*
Лингвистика	*предложение*, *знак*, *граница*, *препинание*, *контекст*, *алгоритм*, *документ*, *анализ*, *эксперимент*

Таблица 3

Список найденных слов при разбиении текста на m-граммы при m = 5 после изменения констант C_1, C_2, C_3

Научное направление	Набор найденных ключевых слов
Математический анализ	*интеграл*, *текст*, *лебег*, *площадь*, *функция*, *риман*, *смысл*, *статья*, *понятие*, *точка*
Дифференциальные уравнения	*уравнение*, *порядок*, *первое*, *частное*, *решение*, *функция*, *теория*, *простейшее*, *коэффициент*
Теория графов	*граф*, *вершина*, *ребро*, *число*, *степень*, *маршрут*, *путь*, *метка*, *множество*, *цикл*
Лингвистика	*предложение*, *знак*, *граница*, *препинание*, *контекст*, *точность*, *алгоритм*, *документ*, *полнота*, *эксперимент*

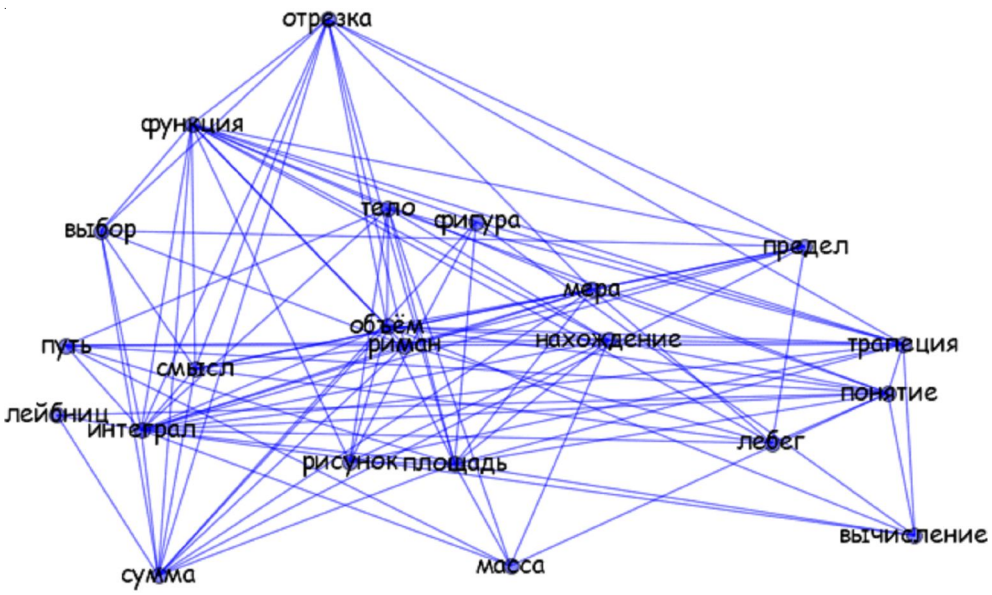

Рис. 2. Представление графа G из алгоритма выделений ключевых слов с 22 вершинами

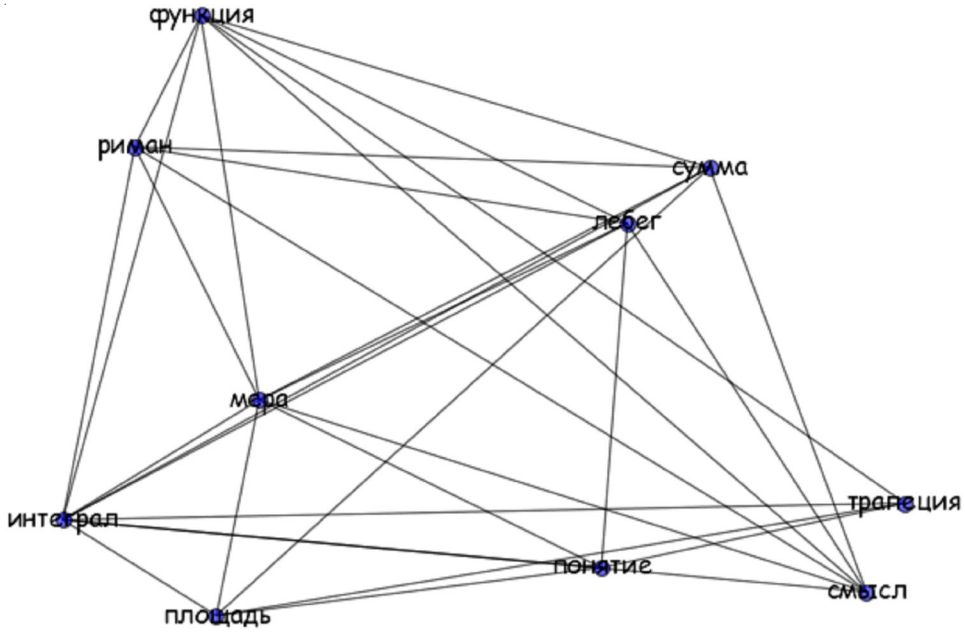

Рис. 3. Представление графа G из алгоритма выделений ключевых слов с 10 вершинами

4. Выводы

Итак, в работе предложена графовая модель представления текста, которая строится на основе разбиения текста на отдельные фрагменты. Вершинами графа являются слова (в базисной форме), причем два слова связаны ребром, если они находятся в одном фрагменте. Проведены компьютерные эксперименты для случая, когда фрагментами являются предложения, а также для случая, когда фрагментами являются N-граммы. По мнению авторов, в первом случае можно получать более значимую информацию, поскольку тот факт, что два слова встречаются в одном предложении более важен, чем то, что эти слова расположены близко друг к другу. В дальнейшем предполагается проведение компьютерных экспериментов с целью обоснования этого предположения.

СПИСОК ЛИТЕРАТУРЫ

Баканова, Н. Б. Обзор программных средств автоматизированного поиска и анализа ключевых слов документов / Н. Б. Баканова // Проблемы современной науки. – 2013. – Вып. 7–3. – С. 40–45.

Ванюшкин, А. С. Методы и алгоритмы извлечения ключевых слов / А. С. Ванюшкин, Л. А. Гращенко // Новые информационные технологии в автоматизированных системах. – 2016. – № 19. – С. 85–93.

Jones, K. S. A statistical interpretation of term specificity and its application in retrieval / K. S. Jones // Journal of Documentation. – 2004. – Vol. 60, iss. 5. – P. 493–502.

REFERENCES

Bakanova N.B. Obzor programmnykh sredstv avtomatizirovannogo poiska i analiza klyuchevykh slov dokumentov [Review of Software of Computer-Assisted Retrieval and Analysis of Documents Keywords]. *Problemy sovremennoy nauki*, 2013, iss. 7-3, pp. 40-45.

Vanyushkin A.S., Grashchenko L.A. Metody i algoritmy izvlecheniya klyuchevykh slov [Methods and Algorithms for Extracting Keywords]. *Novye informatsionnye tekhnologii v avtomatizirovannykh sistemakh* [New Information Technologies in Automated Systems], 2016, no. 19, pp. 85-93.

Jones K.S. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 2004, vol. 60, iss. 5, pp. 493-502.

Information About the Authors

Elena G. Grigoryeva, Candidate of Sciences (Physics and Mathematics), Associate Professor, Department of Computer Science and Experimental Mathematics, Volgograd State University, Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation, e_grigoreva@volsu.ru, kiem@volsu.ru, http://orcid.org/0000-0001-8303-262X.

Vladimir A. Klyachin, Doctor of Sciences (Physics and Mathematics), Associate Professor, Head of Department of Computer Science and Experimental Mathematics, Volgograd State University, Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation, kiem@volsu.ru, http://orcid.org/0000-0003-1922-7847.

Yuriy V. Pomelnikov, Candidate of Sciences (Physics and Mathematics), Associate Professor, Department of Computer Science and Experimental Mathematics, Volgograd State University, Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation, kiem@volsu.ru, http://orcid.org/0000-0001-7311-2941.

Vladimir V. Popov, Candidate of Sciences (Physics and Mathematics), Associate Professor, Department of Computer Science and Experimental Mathematics, Volgograd State University, Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation, popov.vlaval@volsu.ru, kiem@volsu.ru, http://orcid.org/0000-0003-0419-2874.

Информация об авторах

Елена Геннадиевна Григорьева, кандидат физико-математических наук, доцент кафедры компьютерных наук и экспериментальной математики, Волгоградский государственный университет, просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация, e_grigoreva@volsu.ru, kiem@volsu.ru, http://orcid.org/0000-0001-8303-262X.

Владимир Александрович Клячин, доктор физико-математических наук, доцент, заведующий кафедрой компьютерных наук и экспериментальной математики, Волгоградский государственный университет, просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация, kiem@volsu.ru, http://orcid.org/0000-0003-1922-7847.

Юрий Вячеславович Помельников, кандидат физико-математических наук, доцент кафедры компьютерных наук и экспериментальной математики, Волгоградский государственный университет, просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация, kiem@volsu.ru, http://orcid.org/0000-0001-7311-2941.

Владимир Валентинович Попов, кандидат физико-математических наук, доцент кафедры компьютерных наук и экспериментальной математики, Волгоградский государственный университет, просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация, popov.vlaval@volsu.ru, kiem@volsu.ru, http://orcid.org/0000-0003-0419-2874.