Сравнение методов и практические рекомендации по сопоставлению текстовых документов

Д.О. Ишанкулиев,
экономист Центра анализа данных, Всероссийская академия внешней торговли (e-mail: Ishankuliev_Dm@vavt.ru)
М.А. Миткина,
экономист Центра анализа данных, Всероссийская академия внешней торговли (e-mail: Mitkina_Mariya@vavt.ru)
Д.И. Гордеев,
канд. филол. наук, руководитель группы машинной лингвистики Центра анализа данных, Всероссийская академия внешней торговли; научный сотрудник лаборатории анализа данных и отраслевой динамики, Российская академия народного хозяйства и государственной службы при Президенте РФ (e-mail: gordeev-di@ranepa.ru)
Д.В. Филиппов,
д.ф.-м.н., директор Центра анализа данных, Всероссийская академия внешней торговли; ведущий научный сотрудник Института контрольно-надзорной деятельности, Российская академия народного хозяйства и государственной службы при Президенте РФ (e-mail: DFilippov@vavt.ru)

Аннотация. *Целью настоящей работы является рассмотрение различных методов сравнения текстов, а также выявление преимуществ и недостатков исследуемых методик. Дан обзор основных принципов сравнения текстовых документов. Описаны наиболее распространенные методы сопоставления различных тестовых документов или различных версий одного документа (векторное пространство с использованием меры TF-IDF, редакционное расстояние Левенштейна, локально-чувствительное хеширование, сравнение с помощью встроенных функций MS Word и сторонних сервисов). Тестирование и сравнение методик было проведено для задачи проверки и анализа актуальности и полноты сведений, представленных в аналитической информационной системе обеспечения открытости деятельности федеральных органов исполнительной власти, размещенной на Портале госпрограмм (www.programs.gov.ru), в рамках осуществления мониторинга разработки и реализации госпрограмм. Вся информация была получена из открытых источников. В результате были определены рекомендации по применению рассмотренных методов и оценены сложности в их использовании.*

Abstract. *The purpose of this work is to consider various methods of comparing texts, as well as to identify the advantages and disadvantages of these methods. An overview of the basic principles of comparing text documents is given. The most common methods of comparing different text documents or different versions of the same document are described (vector space using the TF-IDF measure, Levenshtein edit distance, locality-sensitive hashing, comparison using integrated MS Word functions and outside services). Testing and comparison of methods was carried out for the task of checking and analyzing the relevance and completeness of the information, provided in the analytical information system for ensuring the openness of the activities of federal executive authorities, on the website of state programs (www.programs.gov.ru), within implementation of the development and implementation monitoring of state programs. All information was obtained from open sources. As a result, practical recommendations for using these methods and difficulties in application were identified.*

Ключевые слова: *классификация текстов; анализ текстовой информации; обработка данных; методы сравнения текстов; веб-документы; веб-ресурсы; алгоритмы.*

Keywords: *text classification; analysis of text information; data processing; methods of comparing texts; web documents; web resources; algorithms.*

1 Введение

Нахождение сходства между текстами является важной задачей в области автоматической обработки, возникающей, например, при построении рефератов, отчетов; машинном переводе; определении авторства; обнаружении академического плагиата; в тестах на понимание; при поиске информации; и многих других ситуациях, когда нужно определить и измерить степень подобия между двумя заданными текстовыми документами [23]. Поиск степени схожести текстов был рассмотрен в качестве одной из основных задач в рамках многих российских и международных конференций, чему уделяется значительное внимание в последние годы.

2 Общее описание принципов сравнения текстовых документов

Прямое сравнение текстов – достаточно сложный алгоритмически и ресурсоемкий процесс. Вместо того, чтобы вручную искать различия между документами (или между версиями документа), могут быть использованы автоматизированные методы. Процесс сравнения документов позволяет сравнить любые два документа, включая версии одного конкретного документа, после чего изменения между обоими документами будут показаны как исправления в исходном документе [12; 20]. Решение задач сравнения документов в области автоматической обработки текстов базируется на сопоставлении текстов и оценке их сходства путём сравнения различных признаков.

Современные методы сравнения текстов учитывают следующие критерии при сопоставлении текстов друг с другом:

1. совместная встречаемость, порядок слов и расстояние между словами в текстах документов [7];
2. синтаксические и семантические связи между словами [3];
3. предметная область и тематику текстов [19];
4. отношения между словами текста в тезаурусе или онтология предметной области (синонимические, родовидовые и др.).

Первая группа критериев не учитывает семантические связи в тексте и относится к нелингвистическим. Вторая группа опирается на результаты синтаксического и семантического анализа. Третья группа критериев подразумевает анализ текстов с последующей кластеризации на множество тематических рубрик. Критерии четвертой группы подразумевают использование специализированных словарей, тезаурусов или лингвистических онтологий [18]. Исследования по сопоставлению текстов можно разделить на две основные группы: первая группа находится на уровне словаря, а другая – на семантическом уровне. Исследования на уровне словаря в основном касаются решения проблемы соответствия на уровне слов, их порядка, орфографии и пунктуации [13]. В данной работе будут разобраны методы, относящиеся ко второй группе исследований.

Существует множество различных методов сравнения текстов: векторное пространство (с использованием TF-IDF), редакционное расстояние (edit distance), локально-чувствительное хеширование (TLSH), Soundex – сравнение документов по их звучанию, сравнение с помощью встроенных функций MS Word и иных сервисов доступных в сети Интернет (например, сервис «Embedika Compare»). Одним из первых появившихся методов по сравнению текстов был метод SoundEx – алгоритм, разработанный в 1922, который позволял сравнивать набор слов по их звучанию, однако смог обрести свою популярность лишь во второй половине XX века благодаря развитию ЭВМ. В то же время начали набирать популярность методы сравнения текстовых документов по ключевым словам. Работа классических алгоритмов по сопоставлению текстовых документов заключается в разделении текста на ключевые слова (термины), которые затем проходят процедуру упорядочения в зависимость от выбранного метода. В качестве метода по оценке значимости слов может служить модель Ципфа-Мандельброта. Благодаря своей универсальности и простоте она стала особенно популярна в конце XX. Модель Ципфа-Мандельброта нашла свое применение в методе взвешивания слов в текстовом документе TF-IDF.

3 Методики сравнения текстов

Векторная модель

Алгоритмы с применением векторных представлений набора слов или текста популярны. Суть метода заключается в представлении текста в качестве элемента векторного пространства [10]. Компоненты вектора отражают частоту терминов в документах, каждое слово является измерением, а документы являются векторами. В случае если точки в векторном пространстве лежат близко, то тексты считаются схожими. Основное различие между методами сравнения текстов состоит в выборе метрики, на основе которой будет определяться расстояние (близость) между векторами.

Один из вариантов выбора метрики – определение «веса» каждого слова. Наиболее распространенным является метод TF-IDF (term frequency – inverse document frequency) [9]. TF – обозначает частотность слова в тексте (локальные веса). IDF – обозначает обратную частотность документа в коллекции (глобальные веса). Для более точного определения весов значения «TF» и «IDF» перемножают [8]. Это обычно называют взвешиванием TF-IDF. В основе метода TF-IDF лежит следующая логика – тексты, в которых определенное слово употребляется с близкой частотой, близки [9]. Контекст употребления слов TF-IDF не учитывает. Вектора текстов строятся, исходя из гипотезы о «мешке слов». Данная мера вычисляется как произведение частотности слов в тексте и обратной частотности слова в наборе документов.

TF некоторого слова ***i*** – это отношение количества раз, которое слово ***i*** встречается в тексте к количеству всех слов в тексте. Ниже приведен пример реализации алгоритма на языке программирования Python (рис. 1).

```
import collections

def compute_tf(text):
#На вход берем текст в виде списка (list) слов
    #Считаем частотность всех терминов во входном массиве с помощью
    #метода Counter библиотеки collections
    tf_text = collections.Counter(text)
    for i in tf_text:
        #для каждого слова в tf_text считаем TF путём деления
        #встречаемости слова на общее количество слов в тексте
        tf_text[i] = tf_text[i]/float(len(text))
    #возвращаем объект типа Counter с TF всех слов текста
    return tf_text

text = ['hasta', 'la', 'vista', 'baby', 'la', 'vista', 'la']
print compute_tf(text)
Out: Counter({'la': 0.42857142857142855, 'vista': 0.2857142857142857,
'hasta': 0.14285714285714285, 'baby': 0.14285714285714285})
```

Рис. 1. **Пример кода (Python)**
Источник: составлено автором.

IDF – обозначает обратную частотность документа в коллекции. Она измеряет значимость определенного слова. Известно, что, например, предлоги встречаются очень часто, хотя практически не влияют на смысл текста. IDF позволяет учесть слова, которые встречаются часто в большинстве текстах. IDF рассчитывается как логарифм от общего числа документов, деленного на количество документов, в которых встречается слово ***i***.

IDF некоторого слова ***i*** – это логарифм отношения общего количества документов к количеству документов, в которых встречается данное слово ***i***. Ниже приведен пример реализации алгоритма на языке программирования Python (рис. 2).

TF-IDF слова-***i*** = (TF слова-***i***) × (IDF слова-***i***)

```
import math

def compute_idf(word, corpus):
#на вход берется слово, для которого считаем IDF
#и корпус документов в виде списка списков слов
        #количество документов, где встречается искомый термин
        #считается как генератор списков
        return math.log10(len(corpus)/sum([1.0 for i in corpus if word
in i]))

texts = [['pasta', 'la', 'vista', 'baby', 'la', 'vista'],
['hasta', 'siempre', 'comandante', 'baby', 'la', 'siempre'],
['siempre', 'comandante', 'baby', 'la', 'siempre']]
print compute_idf('pasta', texts)
Out: 0.47712125472
```

Рис. 2. **Пример кода (Python)**
Источник: составлено автором.

Однако такой метод имеет несколько недостатков:

- Не учитывает порядок слов в тексте
- Не дает возможности сравнить фрагмент с целым текстом

TF-IDF – это очень интересный способ преобразования текстового представления информации в модель векторного пространства (VSM), которая фактически является вектором частотности терминов в документе.

На следующем этапе производится нахождение *коэффициента косинусного сходства* – метрики, которая показывает, насколько связаны два документа. Предположим, у нас есть два вектора: $\vec{a} = (a_1, a_2, \dots, a_n)$ и $\vec{b} =$

$(b_1, b_2, \dots, b_n)$, где a_k и b_k являются компонентами вектора (значения TF-IDF для каждого слова в документе), а *n* – размерность векторов. Косинус между двумя векторами (или двумя документами в векторном пространстве) – это мера, которая вычисляет косинус угла между векторами $\vec{a}$ и $\vec{b}$. Если два документа совершенно разные (вектора ортогональны), то $\cos(\vec{a}, \vec{b}) = 0$, если полностью совпадают (коллинеарны), то $\cos(\vec{a}, \vec{b}) = 1$. Косинусное сходство двух документов изменяется в диапазоне от 0 до 1, поскольку частота терма (вес TF-IDF) не может быть отрицательной, то угол между двумя векторами частоты терма не может быть больше, чем 90° [16].

TF-IDF не учитывает семантику текста в разных документах, т.е. если фрагменты текста имеют одинаковый смысл, но при этом перефразированные, TF-IDF полезен только как функция лексического уровня, что, однако, не является недостатком в контексте текущей задачи по сопоставлению двух документов, в которой важны четкие формулировки и совпадения. Другим недостатком является невозможность просмотра конкретных изменений и различий, внесенных в документы. Данный метод позволяет лишь получить общее представление о совпадении документов.

Помимо векторных представлений на основе TF-IDF также используются методы на основе сжатых представлений Word2Vec. Отличительной особенностью векторов слов в Word2Vec является то, близость употреблённых слов рассматривается с учётом использования слова в схожем контексте (семантическая близость). Один из простейших вариантов – считать для каждого слова частоты его появления в текстах вместе с другими словами. Такие представления имеют право на существование и могут давать приемлемые результаты, однако лучшие результаты дают векторные представления, которые обучаются, т.е. настраивают элементы векторов на некоторых корпусах текстов.

В обзоре [5] представлено описание и применение векторных представлений текстов. В настоящее время развитие этого подхода осуществляется по следующим направлениям:

- дополнение векторных представлений слов другими представлениями, например, представлениями именованных сущностей, частей речи, моделями языка и т.п.;
- создание представлений, которые позволяют учитывать несколько различных смыслов слов (или сочетаний) – омонимы;
- замена векторных представлений функциями распределения плотности вероятности – переход от дискретного пространства к непрерывному;
- введение векторных представлений устойчивых словосочетаний;
- создание представлений, учитывающих специализацию предметной области;
- векторные представления для нескольких языков одновременно; и другие.

Редакционное расстояние

Другой метод определения метрики – редакционное расстояние (расстояние Левенштейна, «Edit distance»). Это метод, который позволяет определить схожесть (различие) между двумя последовательностями символов. Суть данного метода заключается в нахождении минимально возможного числа, так называемых «элементарных операций» последовательность которых преобразовывает один текст в другой. Под элементарными операциями понимаются такие операции как:

- вставка/удаление символа
- замена одного символа на другой

Алгоритм производит подсчет операций, необходимых для преобразования одного набора символов (текста, документа) в другой. Данный метод назван в честь советского математика Владимира Левенштейна, который рассматривал это расстояние в 1965 году [1]. Пусть S_1 и S_2 – две строки (длиной *M* и *N* соответственно), тогда редакционное расстояние (расстояние Левенштейна) $d(S_1; S_2)$ можно подсчитать по следующей формуле: $d(S_1; S_2) = D(M; N)$, где

$$D(0;0) = 0;\ D(0;i) = i;\ D(j;0) = j;$$

$$D(i;j) = \min\begin{Bmatrix} D(i;j-1)+1 \\ D(i-1;j)+1 \\ D(i-1;j-1) + \mathrm{m}(S_1[i]; S_2[j]) \end{Bmatrix}$$

при $i > 0$ и $j > 0$, где $\mathrm{m}(a;b) = 0$, если $a = b$ и единице в противном случае [1]. Здесь шаг по *i* символизирует удаление из первой строки, по *j* – вставку в первую строку, а шаг по обоим индексам символизирует замену символа (или отсутствие изменений). Ниже приведен пример реализации алгоритма на языке программирования Python (рис. 3):

```
def my_dist(a, b):
    def recursive(i, j):
        if i == 0 or j == 0:
            # если одна из строк пустая, то расстояние до другой строки - ее длина
            # т.е. n вставок
            return max(i, j)
        elif a[i - 1] == b[j - 1]:
            # если оба последних символов одинаковые, то съедаем их оба, не меняя
расстояние
            return recursive(i - 1, j - 1)
        else:
            # иначе выбираем минимальный вариант из трех
            return 1 + min(
                recursive(i, j - 1),  # удаление
                recursive(i - 1, j),   # вставка
                recursive(i - 1, j - 1)  # замена
            )
    return recursive(len(a), len(b))
```

Рис. 3. **Пример кода (Python)**
Источник: составлено автором.

Расстояние Левенштейна широко используется для:

- нахождения расхождений текстовых файлов;
- исправления ошибок в слове (в поисковых системах, базах данных, при вводе текста, при автоматическом распознавании отсканированного текста или речи);
- для сравнения генов, хромосом и белков в биоинформатике [15];
- аналог рассматриваемой метрики был использован Филипповым Д. В. в ядерной физике при построении модели низкоэнергетических ядерных реакций – расстояние между двумя ансамблями ядер определялось через количество элементарных преобразований, последовательность которых переводила один ансамбль в другой [4].

Метод определения расстояния Левенштейна имеет ряд недостатков:

1. Даже обычная перестановка слов, не меняющая общего содержания текста способна вызвать получение больших расстояний;

2. В случае, если мы сравниваем расстояния между двумя совершенно разными словами, но имеющими малое число символом редакционное расстояние получится маленьким. Напротив, имея схождения слова с большим набором символов редакционное расстояние получится большим, что говорит о низкой точности данного метода [1].

3. Скорость вычисления расстояния Левенштейна достаточно низка. В случае большого объема текста требуются значительные вычислительные мощности и большой объем памяти [6].

4. Полученный результат может дать только общее представление о сходстве или различии двух документов (текстов) [2].

Локально-чувствительное хеширование (TLSH)

Хеш – это уникальный набор символов (код), рассчитанный по определенному алгоритму и соответствующий определенному набору данных. Существует ряд алгоритмов (хеш-функций), используемых для расчета хеш-суммы: CRC32, SHA256, MD5. В случае, если файл подвергается изменению, изменяется и хеш-сумма этого файла. Для полностью идентичных файлов хеш-сумма будет равной [21].

Для сравнения документов большого объема, имеющих большое количество разделов может быть применено локально-чувствительное хеширование (TLSH). Алгоритмы локально-чувствительного хеширования помещают схожие данные в одну корзину с высокой вероятностью [11]. Однако у такого подхода есть недостаток: минимальные изменения в разделе полностью меняют хеш-файла и два раздела будут определены, как разные, если в одном из них есть пробел, а в другом - нет. Возможно, что два файла Excel содержат одни и те же значения в тех же строках и столбцах, но хеш будет полностью

отличался из-за различного форматирования, стилей, настроек и т. д.

SoundEx

SoundEx – алгоритм, разработанный в 1922, который позволяет сравнивать набор слов по их звучанию [14]. Изначально данный алгоритм был ориентирован на фонетику американского варианта английского языка. В настоящий момент существует ряд модификаций, позволяющих применять и другие языки, включая русский. Алгоритм SoundEx для английского языка состоит из пяти шагов [17]:

Шаг 1. Сохранить первую букву слова.

Шаг 2. Удалить из слова буквы A, E, I, O, U, Y, W и H.

Шаг 3. Заменить оставшиеся буквы на цифры (способ замены – см. [17]).

Шаг 4. Если в коде имеются группа из одинаковых цифр, то заменить эту группу первой цифрой, за исключением цифр, разделяющих буквы W и H в исходном слове.

Шаг 5. Сформировать результирующий код из первой буквы слова и трех первых цифр, полученный на предыдущих шагах. Если в коде менее трех цифр, то дополнить код нулями.

Иногда этот алгоритм не способен обнаружить сходство между очень близкими фамилиями: например, «*Levinson*» получит код L152, а «*Lewinson*» – код L525. Кроме того, Soundex плохо работает в ситуациях, когда произношение сильно расходится с написанием, что в английском языке бывает нередко. По прямому назначению использовать его в русском языке нельзя так, как он дает слишком большие погрешности для единичного слова [22].

Встроенные функции MS Word / Сторонние сервисы

Функцией сравнения также обладает программа Microsoft Word. Встроенный инструмент дает возможность сравнить два документа и выводит различия между двумя файлами. Изменений документа при этом не производится. В качестве изменений инструмент также определяет регистр, пробелы, абзацы. Таким образом, для корректного осуществления сравнений с помощью MS Word документы должны иметь полностью идентичный формат, конструкцию вплоть до пробелов.

Существует также ряд сторонних сервисов и сайтов по сравнению документов и поиску различий. Например, «Embedika Compare». Данный сервис поддерживает работу с большим набором форматов, в число которых входят PDF, DOC и DOCX, не изменяет исходное форматирование документов, не разбивает строки и абзацы. Кроме того, сервис поддерживает сравнение таблиц внутри текстов. Огромным плюсом данного метода является простота его использования и доступность широкому кругу лиц. Готовые решения не требуют дополнительных знаний и навыков, что существенно упрощает их использование. Однако следует отметить недостаток – отсутствие общего понимания в сходстве/различии документов. Вышеуказанные сервисы не дают общую оценку идентичности двух документов, лишь показывая места, в которых были обнаружены изменения.

4. Практические рекомендации по сравнению текстовых документов

Сравнение различных методов было проведено на примере сопоставлении различных редакций текстов Государственных программ. Каждый из методов, обсуждаемых в настоящей работе, превосходит остальные в зависимости от поставленной задачи: размера набора данных, свойств документов (содержания в них таблица, графиков и т.д.) и ограниченности вычислительных ресурсов. При наличии больших наборов однородных данных, без ограничений по памяти и при отсутствии в тексте таблиц и графиков метод локально-чувствительного хеширования (KLSH) обеспечивает наилучший результат.

В случае, если необходимо получить общее понимание совпадения двух документов, без семантического анализа и с минимальными затратами вычислительных ресурсов, полезна векторная модель с весами, рассчитанными методом TF-IDF, при этом необходимо принимать во внимание его особенность – он не позволяет достоверно сравнивать фрагменты с целым. Также, в случае его использования, стоит учитывать, что он принимает в расчет в большей степени количество часто используемых терминов и «занижает» число редких терминов, которые более информативны по своей сути.

В чистом виде метод нахождения редакционного расстояния слабо применим в качестве инструмента нахождения различий в двух документах в виду больших вычислительных затрат,

необходимых для работы алгоритма, в случае большого массива данных и низкой точности полученных результатов. Например, в случае если слова поменяны местами, что можно встретить достаточно часто.

Сравнение с помощью встроенных функций MS Word или сторонних сервисов является самым простым в использовании, поскольку это не требует глубоких знаний алгоритмов, а также опыта работы с языками программирования. Данный метод идеально подходит в случае наличия документов, которые заведомо должны быть заведомо идентичны как, например, законодательные акты, разные версии одного и того же документа и т.п.

5. Заключение

В настоящей работе исследована применимость методов: Векторное пространство (с использованием меры TF-IDF), редакционное расстояние (Расстояние Левенштейна, edit distance), локально-чувствительное хеширование (TLSH), Soundex, сравнение с помощью встроенных функций MS Word/сторонних сервисов. Последний метод подходит для всех уверенных компьютерных пользователей, он выполняет все необходимые функции по сравнению текстов, в случае если не требуется лингвистический анализ. Остальные методы требуют навыков программирования и используются для решения более сложных задач, связанных с обработкой текстов.

Библиографический список:

1.Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР, 1965, Т. 163, №4, С. 845.

2.Лиманова Н. И., Седов М. Н. Алгоритм нечеткого поиска в базах данных и его практическая реализация // Информационные технологии и нанотехнологии, 2017, С. 1885.

3.Соченков И. В. Метод сравнения текстов для решения поисково-аналитических задач // Искусственный Интеллект и принятие решений, 2013, Т. 2, С. 32.

4.Уруцкоев Л. И., Филиппов Д. В. Возможна ли трансформация ядер в низкотемпературной плазме с точки зрения законов сохранения? // Прикладная физика, 2004, №2, С. 30.

5.Харламов А. А., Гордеев Д. И. Дистрибутивная VS сетевая семантика в диалоговых системах // Проблемы искусственного интеллекта, 2019, №2 (13), С. 93.

6.Чувилин К. В. Эффективный алгоритм сравнения документов в формате LATEX // Компьютерные исследования и моделирование, 2015, Т. 7, №2, С. 329.

7.Чугреев В. Л. Модель структурного представления текстовой информации и метод ее тематического анализа на основе частотно-контекстной классификации, дисс. к.т.н. – С.-П., 2003.

8.Chi Tian, Tezuka T., Oyama S., Tajima K., Tanaka K. Web Search Improvement Based on Proximity and Density of Miltiple Keywords // Data Engineering Workshops, 2006, P. 5.

9.Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. An Introduction to Information Retrieval // Cambridge University Press, 2009, P. 19.

10. Joaquin Perez-Iglesias – Integrating the Probabilistic Model BM25: BM25F into Lucene // Cornell University free distribution platform[arXiv.org], 2009, P. 7.

11. Loïc Paulevé, Hervé Jégou, Laurent Amsaleg – Locality sensitive hashing: A comparison of hash function types and querying mechanisms // HAL, 2011, P. 11.

12. Montemurro M. A. Beyond the Zipf-Mandelbrot law in quantitative linguistics // Cornell University free distribution platform[arXiv.org], 2001, P. 578.

13. Robertson S. E., Walker S., Jones S., Hancock-Beaulieu M. and Gatford M. Okapi at TREC-3 // Proceedings of the Third Text Retrieval Conference TREC, 1994, P. 567.

14. Russell R. C, Margaret K. O. US Patent 1262167, 1435663, // USPTO, 1922.

15. Shama Rani, Jaiteg Singh – Enhancing Levenshtein's Edit Distance Algorithm for Evaluating Document Similarity // Communications in Computer and Information Science, 2017, P. 75.

16. Umadevi M., Document Comparison based on TF-IDF metric // International Research Journal of Engineering and Technology (IRJET), 2020, P. 9.

17. Vykhovanets V., Du J., Sakulin S., Overview of Phonetic Encoding Algorithms // Pleiades Publishing, Ltd, 2018, P. 6.

18. Yin, W., and Schutze, H. Task-specific attentive pooling of phrase alignments contributes to sentence matching // Cornell University free distribution platform[arXiv.org], 2017, P. 38.

19. Yue Wang, Hongsong Li, Haixun Wang, Kenny Q. Zhu. Toward Topic Search on the Web // JMIR Publications, 2012, P. 28.

20. Zampieri M., Amorim R. C. Between Sound and Spelling: Combining Phonetics and Clustering Algorithms to Improve Target Word Recovery // Springer International Publishing, 2014, P. 438.

21. Zaragoza H., Craswell N., Taylor M., Saria S. and Robertson S. Microsoft Cambridge at TREC-13: Web and HARD tracks // Conference: Proceedings of the Thirteenth Text Retrieval Conference, TREC, 2004, P. 57.

22. Zeeshan Bhatti, Ahmad Waqas, Imdad Ali Ismaili, Dil Nawaz Hakro, Waseem Javaid Soomr // Phonetic based SoundEx & ShapeEx algorithm for Sindhi Spell Checker System, 2014, P. 7.

23. Zipf, G. K. Selected studies of the principle of relative frequencies of language // Harvard University Press, 1932, P. 67.