УДК 004.912

АНАЛИЗ ПРОБЛЕМ АЛГОРИТМОВ АВТОМАТИЧЕСКОГО РЕФЕРИРОВАНИЯ ТЕКСТА

Гриценко Т.Ю.

студент 2 курса магистратуры

Московский государственный технический университет им. Н. Э. Баумана

Россия, г. Москва

Аннотация

В статье рассмотрены существующие алгоритмы автоматического реферирования текста и их разновидности. Проанализированы проблемы алгоритмов экстрактивного реферирования. В результате анализа выделены следующие проблемы: наличие длинных предложений и расчет оценки метрики схожести. В качестве решения этих проблем предложено применение теории риторических структур и использование обученных моделей векторного представления слов.

Ключевые слова: реферирование, Luhn, SumBasic, LSA, TextRank, LexRank, KL Sum, модели векторного представления слов, теория риторических структур

ANALYSIS OF PROBLEMS OF AUTOMATIC SUMMARIZATION TEXT ALGORITHMS

Gritsenko T.Y.

master student

Bauman Moscow State Technical University

Russian Federation, Moscow

Annotation

The article examines existing algorithms for automatic abstracting of text and their varieties. The problems of extractive abstracting algorithms are analyzed. The analysis identified the following problems: the presence of long sentences and the calculation of the similarity metric. As a solution to these problems, the application of the theory of rhetorical structures and the use of trained models of the vector representation of words are proposed.

Keywords: summarization, Luhn, SumBasic, LSA, TextRank, LexRank, KL Sum, vector representation models of words, theory of rhetorical structures

Введение

Человек с легкостью понимает прочитанный им текст и может извлечь из документа характерные особенности и составить его краткое изложение, поскольку у нас есть способность понять смысл текстового документа. Машина с этой задачей справляется гораздо труднее, а на текущий день имеется большая потребность в составлении автоматических рефератов документов, так как в современном мире довольно много текстовой неструктурированной информации. Реферирование документов полезно во многих областях, например, при исследованиях, прочитав краткое изложение материала, можно сразу понять необходимо ли читать статью дальше или она не подходит для исследования.

Виды алгоритмов автоматического реферирования текста

Существует несколько разновидностей автоматического реферирования. В зависимости от вида итогового реферата, алгоритмы автоматического реферирования делятся на следующие виды [12]:

– экстрактивные алгоритмы, результат работы которых представляет собой набор предложений из исходного документа (или документов);

– абстрактные алгоритмы, результирующий реферат которых состоит из новых сформированных предложений на основе исходного документа.

Реализация абстрактных алгоритмов реферирования гораздо сложнее, чем экстрактивных, они дают надежду на более общие решения проблемы автоматического реферирования, но пока более успешными являются экстрактивные алгоритмы.

Алгоритмы экстрактивного реферирования текста

Существуют следующие алгоритмы экстрактивного реферирования: Luhn, SumBasic, LSA, TextRank, LexRank и KL Sum.

Алгоритм Luhn

Алгоритм Луна является самым простым и базовым методом. После предварительной обработки исходного документа определяются ключевые слова путем подсчета наиболее часто встречающихся. Далее отбирается небольшое количество ключевых слов для оценки предложений в исходном документе. Предложениям выставляется оценка в зависимости от того, сколько ключевых слов они содержат. Необходимое количество предложений с самой высокой оценкой выбираются для формирования реферата, при этом порядок вывода этих предложений соответствует порядку их появления в исходном документе [6].

Алгоритм SumBasic

Алгоритм основан на наблюдении, что слова, которые часто встречаются в документе с большей вероятностью встречаются и в реферате. Последовательность алгоритма включает в себя следующие шаги [8]:

1) Вычисляются вероятности появления слов в исходном документе по формуле 1.

$$p(w_i) = \frac{n}{m}, \quad (1)$$

где n – это количество повторений слова, а m – общее количество слов.

2) Каждому предложению в исходном документе назначается вес, равный средней вероятности слов в нем (формула 2).

$$Weight(S_j) = \sum_{W_i \in S_j} \frac{p(w_i)}{|\{w_i | w_i \in S_j\}|}, \quad (2)$$

3) Выбирается предложение, которое имеет наибольший вес.

4) В выбранном предложении на шаге 3 для каждого слова обновляется вероятность по формуле 3.

$$p_{new}(w_i) = p_{old}(w_i) * p_{old}(w_i) \quad (3)$$

5) Если количество предложений в итоговом реферате меньше необходимого, то происходит переход к шагу 2.

Алгоритм LSA

Алгоритм основан на скрытом семантическом анализе. В него входят следующие шаги [11]:

1) В начале процесса формируется матрица A — матрица терм-предложение. Размер данной матрицы равен $n \times m$, где n — количество термов, m — количество предложений. Если терм i встречается в предложении j, то элемент a_{ij} матрицы A равен частоте встречаемости терма i в тексте.

2) К полученной матрице A применяется разложение:

$$A = U\Sigma V^T, \quad (4)$$

где $U = [u_{ij}]$ – ортонормированная матрица $n \times m$, $\Sigma = diag(\sigma_1, \dots, \sigma_m)$ – диагональная матрица, $V = [v_{ij}]$ – ортонормированная матрица $m \times m$. Если $rank(A) = r$, то выполняется условие:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \sigma_m = 0 \quad (5)$$

3) Далее для каждого предложения s_k изначального текста рассчитывается вес по формуле 6.

$$s_k = \sqrt{\sum_{i=1}^{m} v_{ik}^2 * \sigma_i^2} \quad (6)$$

4) Далее отбирается необходимое количество предложений для генерации реферата путем отбора предложений с наибольшим весом.

Алгоритм TextRank

Данный алгоритм используется и для извлечения ключевых слов и для автоматического реферирования текста. Применительно к реферированию текста он имеет следующую последовательность шагов [7]:

1) По тексту строится взвешенный неориентированный граф, вершины в котором обозначают предложения текста. Весом ребра между двумя вершинами является степень схожести двух предложений, соответствующих вершинам. Она вычисляется, как количество совпадающих слов в предложениях, нормированное суммарной длиной этих предложений.

2) Далее каждой вершине итерационно присваивается вес по формуле 7.

$$W(V_i) = (1 - d) + d * \sum_{V_j \in Inc(V_i)} \frac{w_{ij}}{\sum_{V_j \in Inc(V_i)} W(V_j)}, \quad (7)$$

где V_i, V_j – вершины графа, $Inc(V_i)$ – множество вершин, смежных вершиной V_i, w_{ij} – вес ребра между вершинами V_i, V_j, d – коэффициент затухания равный 0.85.

3) В реферат отбираются предложения с наибольшим весом.

Алгоритм LexRank

Основная идея данного алгоритма состоим в том, что предложения «рекомендуют» другие похожие предложения. Таким образом, если одно предложение будет похоже на многие другие, оно, вероятно, будет важным. Важность этого предложения также вытекает из важности предложений, «рекомендующих» его. Последовательность алгоритма включает в себя следующие шаги [4]:

1) Каждое предложение обрабатывается как узел в графе.

2) Вычисляется похожесть двух предложений в тексте по формуле 8. Для каждого слова, встречающегося в предложении, значение соответствующего измерения в векторном представлении предложения – это число вхождений слова в предложении, умноженное на idf слова.

$$idf\_modified\_cos(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}}, \quad (8)$$

где $tf_{w,s}$ – это число вхождений слова w в предложение s.

3) Набор предложений характеризуется матрицей схожести, каждый элемент которой соответствует вычисленной величине схожести между парой предложений. Эту матрицу можно интерпретировать как матрицу связности взвешенного графа, содержащую веса ребер.

4) Веса упорядочиваются по убыванию, и в реферат включаются N предложений с наибольшим весом.

Алгоритм KL Sum

Данный алгоритм отбирает предложения на основе того, насколько они имеют такое же распределение слов, что и исходный текст. KL Sum вводит следующий критерий для выбора предложений в реферат S для документа D (формула 9) [5]:

$$S^* = \min_{S:words(S)\leq L} KL(P_D||P_S)\,, \quad (9)$$

где P_S – это эмпирическое распределение униграмм кандидатов на включение в реферат, KL(P||Q) – дивергенция Кульбака-Лейблера, которая определяется по формуле 10.

$$\mathrm{KL(P||Q)} = \sum_w P(w) \log \frac{P(w)}{Q(w)}\,, \quad (10)$$

где P – истинное распределение (исходный документ), Q – аппроксимирующее распределение (реферат).

Проблемы алгоритмов экстрактивного реферирования текста и их решения

Проблема метрики схожести

Графовые алгоритмы экстрактивного реферирования текста отбирают предложения в итоговый реферат на основе рассчитанной схожести между ними. Например, в алгоритме TextRank схожесть вычисляется, как количество совпадающих слов в предложениях, нормированное суммарной длиной этих предложений. Такая метрика схожести не дает качественного результата, потому что при подсчете учитываются только одинаковые слова, но в предложениях также могут использоваться семантически близкие слова для исключения лексических повторов в тексте. Например, имеются два предложения:

«Ствол у березы белый, покрыт тонкой берестой, а нижняя часть как бы прочерчена черными полосками.»

«Это дерево привлекает своим внешним чарующим видом.»

В этих предложениях употребляются слова «береза» и «дерево». Согласно, алгоритму метрика схожести этих предложений будет равна нулю, так как они не имеют повторяющихся слов. Но при корректном подсчете схожести предложений, слова «дерево» и «береза» должны повлиять на данную метрику и увеличить ее.

Для решения этой проблемы при расчете схожести предложений необходимо использовать инструменты, позволяющие узнать семантическую схожесть слов. Семантическую схожесть слов можно определить с использованием обученных семантических моделей, представленных на ресурсах RusVectores и WebVectores.

Ресурс RusVectores предоставляет обученные семантические модели для русского языка [10], аналогичный ресурс WebVectors дает возможность пользоваться обученными моделями для английского языка [13]. Модели работают с лексическими векторами слов. Вектор представляет значение слова, автоматически извлеченное из статистики совместной встречаемости слов в корпусах (больших коллекциях текстовых данных).

Обучение дистрибутивных моделей на основе больших корпусов требует существенных вычислительных мощностей, поэтому RusVectores и WebVectors предоставляют доступ к уже предобученным моделям.

В дистрибутивной семантике слова обычно представляются в виде векторов в многомерном пространстве их контекстов. Семантическое сходство вычисляется как косинусная близость между векторами двух слов и может принимать значения в промежутке [-1...1] (на практике часто используются только значения выше 0). Значение 0 означает, что у этих слов нет похожих

контекстов и их значения не связаны друг с другом. Значение 1, напротив, свидетельствует о близком значении [10].

Пример использования обученных моделей для получения оценки семантической близости, представленных на ресурсе RusVectores показан на рис.1.

```
models = ['ruscorpora_upos_cbow_300_20_2019',
          'ruwikiruscorpora_upos_skipgram_300_2_2019',
          'tayga_upos_skipgram_300_2_2019']

words = [['береза_NOUN', 'дерево_NOUN'],
         ['береза_NOUN', 'ствол_NOUN'],
         ['береза_NOUN', 'небо_NOUN']]

count = 0
for model in models:
    for word in words:
        similarity_ru[count].append(api_similarity_ru(model, word[0], word[1])[:6])
    count += 1

print_table(models, words, similarity_ru)
```

	ruscorpora_upos_cbow_300_20_2019	ruwikiruscorpora_upos_skipgram_300_2_2019	tayga_upos_skipgram_300_2_2019
['береза_NOUN', 'дерево_NOUN']	0.7142	0.7223	0.7678
['береза_NOUN', 'ствол_NOUN']	0.5546	0.4676	0.5858
['береза_NOUN', 'небо_NOUN']	0.3972	0.3783	0.3002

Рис. 1 – Пример использования моделей RusVectores [Составлено автором]

С использованием моделей векторного представления слов при подсчете метрики схожести можно учитывать синонимы, выставляя пороговое значение, при котором слова необходимо считать семантически близкими.

Проблема длинных предложений

Существует еще одна проблема общая для всех алгоритмов реферирования – длинные предложения. Длинные предложения могут иметь водные части, которые не несут существенной информации, но при подсчете совпадений слов, они получат высокий вес, потому что вероятность того, что слово из длинного предложения встречается в других предложениях повышается из-за их количества. Поэтому алгоритм будет выделять такие предложения в качестве ключевых, что увеличивает общий размер полученного реферата и добавляет в него несущественную информацию.

Для решения этой проблемы необходимо перед обработкой текста каким-либо алгоритмом реферирования выделять главную часть предложения, если в

нем есть несущественная информация, таким образом сокращая предложение. Это можно реализовать с применением теории риторических структур. Исследование применимости теории риторических структур для обработки текста приведено в статье [1], пример использования в работе [2].

Теория риторических структур моделирует содержательное строение текста посредством выделения отношений. Она устанавливает два разных типа единиц: спутники и ядра. Ядра считаются наиболее важными частями текста, тогда как спутники вносят вклад в ядра и являются вторичными. Ядро содержит основную информацию, а спутник содержит несущественную информацию [3].

Существуют одноядерные и многоядерные риторические отношения. В многоядерных отношениях оба элемента являются главными, а значит оба содержат значимую информацию. При одноядерном риторическом отношении в предложении обязательно присутствует та часть, которая не несет существенной информации. Такие части (спутники) можно выделять с использованием маркеров языка. Пример одноядерного риторического отношения с маркером «поскольку» представлен на рис. 2.

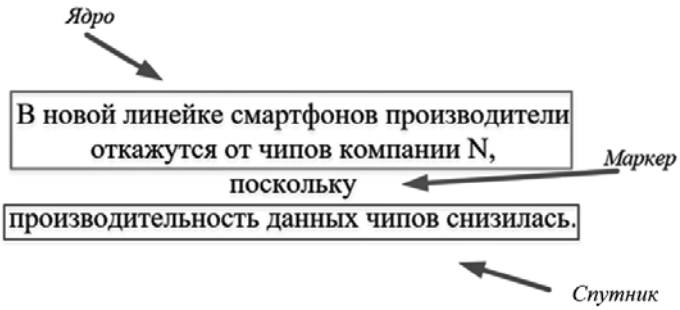

Рис. 2 – Пример одноядерного риторического отношения [Составлено автором]

Список маркеров для русского языка можно найти на ресурсе Ru-RSTreebank. Ru-RSTreebank – это русскоязычный дискурсивный корпус, предназначенный для исследователей, заинтересованных в изучении письменного дискурса [9].

С применением теории риторических структур можно выделять ядра из предложений перед сжатием текста алгоритмами реферирования для решения проблемы длинных предложений.

Заключение

В рамках статьи были рассмотрены решения проблем экстрактивных алгоритмов реферирования текста. Для решения проблемы метрики схожести было предложено использовать обученные модели векторного представления слов, представленные на ресурсах RusVectores и WebVectores, однако для реферирования текстов, касающихся определенной нестандартной темы, необходимо самостоятельно обучать векторные модели на документах соответствующей тематики, так как модели могут «не знать» некоторых слов. Для решения проблемы длинных предложений было предложено применять теорию риторических структур для удаления частей предложений, не несущих существенную информацию, перед работой алгоритмов экстрактивного реферирования.

Библиографический список

1 Бакиева Т. В. Исследование применимости теории риторических структур для автоматической обработки научно-технических текстов / Бакиева Т. В., Батура А.М // Cloud of Science. – 2017. – №3 [Электронный ресурс]. — Режим доступа — URL: https://cyberleninka.ru/article/n/issledovanie-primenimosti-teorii-ritoricheskih-struktur-dlya-avtomaticheskoy-obrabotki-nauchno-tehnicheskih-tekstov (Дата обращения 27.04.2020)

2 Батура Т. В. Создание системы автоматического реферирования научных текстов / Бакиева Т. В., Батура А.М // Вестник НГУ. – 2018. – №3 [Электронный ресурс]. — Режим доступа — URL: https://cyberleninka.ru/article/n/sozdanie-sistemy-avtomaticheskogo-referirovaniya-nauchnyh-tekstov (Дата обращения 27.04.2020)

3 Теория риторических структур [Электронный ресурс]. — Режим доступа — URL: https://ru.wikipedia.org/wiki/Теория_риторических_структур (Дата обращения 20.04.2020)

4 Erkan G. LexRank: Graph-based Lexical Centrality as Salience / G. Erkan, D.R. Radev [Электронный ресурс]. — Режим доступа — URL: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html (Дата обращения 15.04.2020)

5 Haghighi A. Exploring Content Models for Multi-Document Summarization / A. Haghighi, L. Vanderwende [Электронный ресурс]. – URL: https://www.aclweb.org/anthology/N09-1041.pdf (Дата обращения 21.04.2020)

6 Luhn H.P. The Automatic Creation of Literature Abstracts / H.P. Luhn // IBM Journal. – 1958 [Электронный ресурс]. — Режим доступа — URL: http://courses.ischool.berkeley.edu/i256/f06/papers/luhn58.pdf (Дата обращения 21.11.2014)

7 Mihalcea R. TextRank: Bringing Order into Texts / R. Mihalcea, Tarau P. [Электронный ресурс]. — Режим доступа — URL: https://www.aclweb.org/anthology/W04-3252.pdf (Дата обращения 10.04.2020)

8 Nenkova A. The Impact of Frequency on Summarization / A. Nenkova, L. Vanderwende / [Электронный ресурс]. — Режим доступа — URL: https://www.cs.bgu.ac.il/~elhadad/nlp09/sumbasic.pdf (Дата обращения 11.04.2020)

9 Ru-RSTreebank. Русскоязычный дискурсивный корпус [Электронный ресурс]. — Режим доступа — URL: https://rstreebank.ru/ (Дата обращения 25.04.2020)

10 RusVectores: семантические модели для русского языка [Электронный ресурс]. — Режим доступа — URL: https://rusvectores.org/ru/ (Дата обращения 21.04.2020)

11 Steinberger J. Using Latent Semantic Analysis in Text Summarization / J. Steinberger, K. Ježek [Электронный ресурс]. — Режим доступа — URL:

http://www.kiv.zcu.cz/~jstein/publikace/isim2004.pdf (Дата обращения 13.04.2020)

12 Unsupervised Text Summarization using Sentence Embeddings [Электронный ресурс]. — Режим доступа — URL: https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1 (Дата обращения 03.04.2020)

13 WebVectors: word embeddings online [Электронный ресурс]. — Режим доступа — URL: http://vectors.nlpl.eu/explore/embeddings/en/ (Дата обращения 23.04.2020)

Оригинальность 87%