УДК 004.021
MSC 68W05

Нахождение наличия заимствований в научных работах на основе марковских цепей

Р. Р. Саакян[1], *И. А. Шпехт*[2], *Г. А. Петросян*[1]

[1] Ванадзорский государственный университет им. О. Туманяна, Республика Армения, 2001, Ванадзор, ул. Тигран Мец, 36
[2] Академия маркетинга и социально-информационных технологий — ИМСИТ, Российская Федерация, 350010, Краснодар, ул. Зиповская, 5

Для цитирования: *Саакян Р. Р., Шпехт И. А., Петросян Г. А.* Нахождение наличия заимствований в научных работах на основе марковских цепей // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2023. Т. 19. Вып. 1. С. 43–50. https://doi.org/10.21638/11701/spbu10.2023.104

Цель исследования заключается в выработке оптимальных подходов для поиска заимствований в научных работах. Рассматриваются этапы нахождения наличия заимствований: предобработка, грубое фильтрование текстов, нахождение схожих текстов, нахождение заимствований. Основное внимание уделяется описанию подходов и техник, которые можно эффективно реализовать на каждом из этапов, например перевод символов текста из заглавных в строчные, удаление знаков препинаний, удаление стоп-слов для этапа предобработки; фильтры по темам и частоте слов для этапа грубого фильтрования текста; подсчет важности слов в контексте текста и представление слова в виде вектора в многомерном пространстве для определения меры близости для этапа нахождения схожих текстов; поиск точного совпадения, перефразов и меры близости выражений для этапа нахождения заимствований. Научная новизна заключается в предлагаемом в статье использовании марковских цепей для нахождения схожести текстов для второго и третьего этапов процесса поиска заимствований. На примере показана методика применения марковских цепей для представления текста, поиска наиболее часто встречающихся слов, построения графа марковской цепи слов и перспективы использования марковских цепей текстов для грубого фильтрования и поиска схожих текстов.

Ключевые слова: поиск заимствований, алгоритмы нахождения заимствований, марковские цепи, программное обеспечение проверки на оригинальность.

1. Введение. В настоящее время для обеспечения самостоятельности соблюдения прав интеллектуальной собственности и публикационной этики обязательной является проверка научных работ на наличие заимствований.

Здесь проблемой видится существование большого количества характеристик, которые будут определять заимствования. Однако они могут игнорироваться как экспертами, так и действующим программным обеспечением, проверяющим публикации на оригинальность. В работе будут рассмотрены подходы поиска заимствований в текстах с целью разработки программного обеспечения для применения в образовательной и научной областях. Подробно рассматривается использование марковских цепей для исследования текстов.

2. Постановка задачи. Пусть имеем базу текстов. В ней есть n текстов T_1, T_2,..., T_n. Нам дан новый текст S. Его нужно сравнить с текстами из базы и найти наиболее вероятные заимствования. При этом нужно понимать, что заимствования бывают правомерными и неправомерными. К правомерным относят *прямое цити-*

рование (дословное воспроизведение фрагмента текста с обязательной ссылкой на первоисточник), *цитирование по вторичным источникам*, так называемое *кросс-цитирование* (дословное воспроизведение фрагмента текста с обязательной ссылкой на вторичный источник), *косвенное цитирование*, или *парафраз* (краткое изложение большого текста при ссылке на первоисточники) и *самоцитирование* (повторное применение собственных текстов со ссылкой на первоисточник).

К неправомерным заимствованиям относят *плагиат* (умышленное использование чужих тезисов без указания ссылки на первоисточник) и *самоплагиат* (повторение собственных текстов без ссылок или в неоправданном объеме) [1].

Для простоты изложения будем сначала рассматривать любое совпадение в текстах.

3. Этапы и подходы нахождения наличия заимствований. Решение будет состоять из следующих этапов:

1) предобработка (preprocessing);
2) грубое фильтрование текстов;
3) нахождение схожих текстов;
4) нахождение заимствований.

Опишем детально эти этапы.

Этап 1 — предобработка.

На этом этапе нужно подготовить текст для будущей обработки. Предложим оставить как можно меньше уникальных слов, при этом не потеряв смысла текста. Для данного этапа можно использовать следующие техники [2, 3].

Перевод букв из заглавных в строчные (lowercasing).

Лемматизация — перевод слов в их словарные формы.

Примеры:

— собакой собака;
— сидела сидеть.

Удаление цифр и картинок. Тексты, отличающиеся только числами и картинками, можно считать одинаковыми, поэтому ими можно пренебречь.

Удаление стоп-слов: это часто употребляемые слова, которые имеют мало значения, и без контекста они бессмысленны.

Примеры:

— местоимения: он, она, его;
— предлоги: в, на, у, к;
— глагол «есть»: был, есть, были.

Этап 2 — грубое фильтрование текстов.

В реальной жизни база текстов может состоять из большого количества текстов. Но использовать на этих текстах сложные алгоритмы очень трудозатратно. Даже если обрабатывать каждый текст в течение долей секунды, на обработку всей базы текстов могут уйти месяцы.

Поэтому нужно уметь быстро отфильтровывать большое количество текстов. На этом этапе должна отфильтроваться большая часть всех текстов из базы.

Для того чтобы проводить такое фильтрование текстов, нужно применять очень грубые техники, которые помогут отсеять такое большое количество текстов. Рассмотрим возможные варианты.

Фильтр по темам. Для этого определяют, к каким темам относится текст S, и отсеивают все тексты, в тематике которых нет ни одной темы из текста S. Это очень

просто делать, если у текстов уже есть размеченные темы. Если же таких тем нет, то могут помочь алгоритмы глубокого обучения [4].

Фильтр по частоте слов. Для каждого текста подсчитаем количество наиболее встречаемых слов. Выберем список из 100 самых встречаемых слов в тексте (этот параметр можно варьировать).

Пусть для текста T это слова $w_1(T), w_2(T)$,..., $w_{100}(T)$, и они встречаются соответственно $k_{w1}(T), k_{w2}(T)$,..., $k_{w100}(T)$ раз в тексте.

Можно интерпретировать эти числа как вероятности нахождения в тексте.

Пусть $k = k_{w1}(T) + k_{w2}(T) + ... + k_{w100}(T)$. Тогда $p_{w1}(T) = k_{w1}(T)/k$, $p_{w2}(T) = k_{w2}(T)/k$, ..., $p_{w100}(T) = k_{w100}(T)/k$. Исходя из этого, чтобы определить схожесть текстов S и T, можно применить формулу

$$\text{similarity}(S,T) = \sum_{i=1}^{100} (I(w_i(T), S)|p_{wi}(T) - p_{wi}(S)|,$$

где $I(w_i(T), S)$ — это индикатор присутствия слова $w_i(T)$ в топ-100 слов из текста S.

Также вместо данной метрики можно применять косинусное расстояние векторов для текстов S и T. После этого можно выбрать 100 самых похожих статей по такой метрике и использовать их на следующем этапе.

Фильтр по N-граммам (последовательности из n элементов). Принцип этого метода такой же, как и в предыдущем пункте, только теперь вместо одного слова берут N подряд идущих слов, которые и называются N-грамма. Также подсчитывают топ-100 N-грамм в текстах и применяют одну из метрик расстояния для этих векторов текста.

Этап 3 — нахождение схожих текстов.

На этом этапе ожидается несколько тысяч текстов, оставшихся после грубой фильтрации. Сейчас можно воспользоваться более сложными алгоритмами для нахождения схожих текстов. Для этого можно использовать следующие техники.

TF-IDF. Это статистическая мера для подсчета важности слова в контексте текста (TF — term frequency, IDF — inverse document frequency). Можно взять топ-100 самых популярных слов из текста S и подсчитать *tf-idf* для них в тексте T. Тогда метрику для данных текстов можно будет рассчитать по формуле

$$\text{similarity}(S,T) = \sum_{i=1}^{100} p_{w1}(S) * tf - idf(w_i(S), T).$$

Затем можно выбрать 10 самых схожих текстов и оставить их только в качестве текстов заимствований. Также можно определить минимальный барьер метрики схожести и выбрать только те тексты, которые его проходят.

Word2Vec. Это способ представления слова в виде вектора в многомерном пространстве [5]. Будем определять меру близости слов схожих текстов и использовать также взвешенную сумму, как и в TF-IDF. Однако такой алгоритм представляется достаточно времязатратным.

Paragraph2Vec. Можно пройтись по параграфам каждого из текстов и представить их в виде векторов в многомерном пространстве. И уже эти вектора сравнивать друг с другом косинусным расстоянием [6].

Этап 4 — нахождение заимствований.

Это последний этап нахождения плагиата в текстах, на котором уже имеем не более десятка текстов и хотим найти их использование в тексте S.

Укажем возможные техники.

Точное совпадение. Нахождение точных совпадений участков из нескольких слов в текстах.

Нахождение перефразов. Для этого можно использовать поиск их смысловых эквивалентов, для чего можно применять методики этапа 3, например такие, как Word2Vec.

Paragraph2Vec. Можно пройтись по параграфам каждого из текстов и представить их в виде векторов в многомерном пространстве. И уже эти векторы сравнивать друг с другом косинусным расстоянием [6].

4. Использование *марковских цепей* для нахождения заимствований. Для нахождения схожести текстов на этапах 2 и 3 предложим разработанный нами метод использования марковских цепей (марковская цепь — последовательность случайных событий с конечным или счетным числом исходов, где вероятность наступления каждого события зависит только от состояния, достигнутого в предыдущем событии).

4.1. Представление текста в виде марковской цепи. Рассмотрим некоторый текст T после предобработки на первом этапе в виде марковской цепи. Каждое уникальное слово в тексте будет представлено в виде вершины. Для простоты будем обозначать вершину, отвечающую за слово w_i при помощи v_i. Изучим все биграммы (w_i, w_j). Тогда количество биграмм (w_i, w_j) в тексте обозначим за n_{ij}, и для каждой пары слов (i, j) подсчитаем значение переменной p_{ij} по формуле

$$p_{ij} = n_{ij} / \sum_k n_{ik}.$$

При этом очевидно, что

$$\sum_j p_{ij} = 1.$$

Между вершинами v_i и v_j можно провести ребро с весом p_{ij}. Таким образом получим марковскую цепь для текста T.

В целом можно составить топ-100 самых частых слов.

Пример.

Исходные тексты:

«Математика — наука про числа».
«Все числа мне нравятся».
«Математика мне нравится тоже».
«Математика как наука мне нравится».
«Числа в математике важны».

Тексты после предобработки:

математика наука число
число нравиться
математика нравиться
математика наука нравиться
число математика важно

Считаем биграммы:

Математика:
(математика, наука) — 2,
(математика, важно) — 1,
(математика, нравиться) — 1.
Итого: 4.

Наука:
(наука, число) — 1,
(наука, нравиться) — 1.
Итого: 2.
Число:
(число, нравиться) — 1,
(число, математика) — 1.
Итого: 2.
Нравиться:
нет пар
Важно:
нет пар

Превращаем в вероятностные переходы:

Математика:
(математика, наука) — 0.5,
(математика, важно) — 0.25,
(математика, нравиться) — 0.25.
Наука:
(наука, число) — 0.5,
(наука, нравиться) — 0.5.
Число:
(число, нравиться) — 0.5,
(число, математика) — 0.5.
По полученным данным построим граф марковской цепи (рисунок).

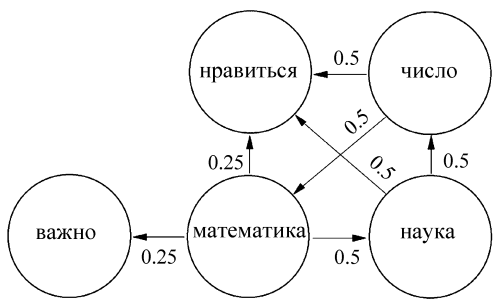

Рисунок. Пример графа марковской цепи для сравниваемых текстов

4.2. Использование марковских цепей текстов. Как было указано выше, марковские цепи хорошо могут пригодиться в грубом фильтровании и при нахождении схожих текстов. Укажем возможные техники.

Схожесть марковских цепей (грубое фильтрование текстов, 2-й этап). Можно попробовать каким-то образом определять схожесть марковских цепей. Например, вычислять PageRank для всех вершин и найти косинусное расстояние между PageRank метриками всех вершин двух цепей для текстов S и T. PageRank представляет собой алгоритм для ранжирования веб-страниц в результатах поиска, используемый механизмом поиска Google. Также можно исследовать и другие методы схожести марковских цепей.

Генерирование текстов в виде марковских цепей (нахождение схожих текстов, 3-й этап). Генерирование текстов в виде марковских цепей показано в работах [7, 8]. В контексте поставленной в статье проблемы на этапе нахождения схожих текстов можно сгенерировать несколько параграфов при помощи марковских цепей и затем сравнить их при помощи метода Paragraph2Vec [9].

5. Заключение. Очевидно, что задача сравнения текстов на предмет схожести остается актуальной и требует привлечения современных методов глубокого обучения. Был приведен алгоритм поэтапного подхода при сравнении текстов и предложены подходящие методы исследования на каждом этапе. Особенный интерес представляет использование на различных этапах исследования инструмента марковских цепей.

Литература

1. Заимствования в научных публикациях и рекомендации по оформлению цитирований. М.: Рос. эконом. ун-т им. Г. В. Плеханова, 2022.
URL: https://www.rea.ru/ru/org/managements/orgnirupr/Pages/Заимствования.aspx (дата обращения: 1 сентября 2022 г.).

2. *Agrawal R.* Must known techniques for text preprocessing in NLP // Analytics Vidhya. 2022. URL: https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/ (дата обращения: 1 сентября 2022 г.).

3. *Camacho-Collados J., Pilehvar M. T.* On the role of text preprocessing in neural network architectures // An evaluation study on text categorization and sentiment analysis. 2018.
URL: https://arxiv.org/pdf/1707.01780.pdf (дата обращения: 1 сентября 2022 г.).

4. *Minaee Sh., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M., Gao J.* Deep learning based text classification: a comprehensive review. Cornell: Cornell University, 2020.
URL: https://arxiv.org/pdf/2004.03705.pdf (дата обращения: 1 сентября 2022 г.).

5. *Mikolov T., Chen K., Corrado G., Dean J.* Efficient estimation of word representations in vector space. Cornell: Cornell University, 2013. URL: https://arxiv.org/pdf/1301.3781.pdf (дата обращения: 1 сентября 2022 г.).

6. *Le V., Mikolov T.* Distributed representations of sentences and documents. Cornell: Cornell University, 2014. URL: https://arxiv.org/pdf/1405.4053.pdf (дата обращения: 1 сентября 2022 г.).

7. *Yang Zh., Jin Sh., Huang Y., Zhang Y., Li H.* Automatically generate steganographic text based on Markov model and Huffman coding. Cornell: Cornell University, 2018. URL:
https://arxiv.org/ftp/arxiv/papers/1811/1811.04720.pdf (дата обращения: 1 сентября 2022 г.).

8. *Thelin R.* Build a deep learning text generator project with Markov chains // Educative, 2022. URL: https://www.educative.io/blog/deep-learning-text-generation-markov-chains (дата обращения: 1 сентября 2022 г.).

9. *Papadopoulos A., Roy P., Pachet F.* Avoiding plagiarism in Markov sequence generation // Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. July 27–31, 2014. P. 2731–2737. URL: https://www.francoispachet.fr/wp-content/uploads/2021/01/papadopoulos-14a.pdf (дата обращения: 1 сентября 2022 г.).

Статья поступила в редакцию 13 ноября 2022 г.
Статья принята к печати 19 января 2023 г.

Контактная информация:

Саакян Рустам Рафикович — ректор, д-р техн. наук, проф.; rsahakyan@yahoo.com

Шпехт Ирина Александровна — канд. техн. наук, доц.; shpekht@mail.ru

Петросян Геворг Арменович — магистр; gapetrosyan14@gmail.com

Finding the presence of borrowings in scientific works based on Markov chains

R. R. Saakyan[1], *I. A. Shpekht*[2], *G. A. Petrosyan*[1]

[1] Vanadzor State University named after H. Tumanyan, 36, ul. Tigran Mets, Vanadzor, 2001, Republic of Armenia

[2] Academy of Marketing and Social Information Technologies — IMSIT, 5, Zipovskaya ul., Krasnodar, 350010, Russian Federation

For citation: Saakyan R. R., Shpekht I. A., Petrosyan G. A. Finding the presence of borrowings in scientific works based on Markov chains. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2023, vol. 19, iss. 1, pp. 43–50. https://doi.org/10.21638/11701/spbu10.2023.104 (In Russian)

The study aims to develop optimal approaches to the search for borrowings in scientific works. The article discusses the stages of searching for the presence of borrowings, such as preprocessing, rough filtering of texts, searching for similar texts, and searching for borrowings. The main focus is on the description of approaches and techniques that can be effectively implemented at each stage. For example, for the preprocessing stage, it may be converting text characters from uppercase to lowercase, removing punctuation marks, and removing stop words. For the stage of rough text filtering, it is filters by topic and word frequency. It may be calculating the importance of words in the context of the text and representing the word as a vector in multidimensional space to determine the proximity measure for the stage of finding similar texts. Finally, it is a search for an exact match, paraphrases and a measure of similarity of expressions for the stage of finding borrowings. The scientific novelty lies in using Markov chains to find the similarity of texts for the second and third stages of the search for borrowings proposed by authors. As a result, the example shows the technique of using Markov chains for text representation, searching for the most frequently occurring words, building a graph of a Markov chain of words, and the prospects for using Markov chains of texts for rough filtering and searching for similar texts.

Keywords: search for borrowings, algorithms for finding borrowings, Markov chains, originality checker software.

References

1. *Borrowings in scientific publications and recommendations for citations.* Moscow, Plekhanov Russian University of Economics Press, 2022. Avalaible at: https://www.rea.ru/ru/org/managements/orgnirupr/Pages/Заимствования.aspx (accessed: September 1, 2022).

2. Agrawal R. Must known techniques for text preprocessing in NLP. *Analytics Vidhya*, 2022. Avalaible at: https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/ (accessed: September 1, 2022).

3. Camacho-Collados J., Pilehvar M. T. On the role of text preprocessing in neural network architectures. *An evaluation study on text categorization and sentiment analysis*, 2018. Avalaible at: https://arxiv.org/pdf/1707.01780.pdf (accessed: September 1, 2022).

4. Minaee Sh., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M., Gao J. *Deep learning based text classification: a comprehensive review.* Cornell, Cornell University Press, 2020. Avalaible at: https://arxiv.org/pdf/2004.03705.pdf (accessed: September 1, 2022).

5. Mikolov T., Chen K., Corrado G., Dean J. *Efficient estimation of word representations in vector space.* Cornell, Cornell University Press, 2013. Avalaible at: https://arxiv.org/pdf/1301.3781.pdf (accessed: September 1, 2022).

6. Le V., Mikolov T. *Distributed representations of sentences and documents.* Cornell, Cornell University Press, 2014. Avalaible at: https://arxiv.org/pdf/1405.4053.pdf (accessed: September 1, 2022).

7. Yang Zh., Jin Sh., Huang Y., , Zhang Y., Li H. *Automatically generate steganographic text based on Markov model and Huffman coding.* Cornell, Cornell University Press, 2018. Avalaible at: https://arxiv.org/ftp/arxiv/papers/1811/1811.04720.pdf (accessed: September 1, 2022).

8. Thelin R. Build a deep learning text generator project with Markov chains. *Educative*, 2022. Avalaible at: https://www.educative.io/blog/deep-learning-text-generation-markov-chains (accessed: September 1, 2022).

9. Papadopoulos A., Roy P., Pachet F. Avoiding plagiarism in Markov sequence generation. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, July 27–31, 2014, pp. 2731–2737. Avalaible at: https://www.francoispachet.fr/wp-content/uploads/2021/01/papadopoulos-14a.pdf (accessed: September 1, 2022).

Received: November 13, 2022.
Accepted: January 19, 2023.

Authors' information:

Rustam R. Saakyan — Rector, Dr. Sci. in Technical Sciences, Professor; rsahakyan@yahoo.com

Irina A. Shpekht — PhD in Technical Sciences, Associate Professor; shpekht@mail.ru

Gevorg A. Petrosyan — Master; gapetrosyan14@gmail.com