www.volsu.ru

DOI: https://doi.org/10.15688/mpcm.jvolsu.2025.3.3

УДК 519.176+004.8
ББК 22.176

Дата поступления статьи: 22.04.2025
Дата принятия статьи: 06.05.2025

О ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ИНДЕКСА ВИНЕРА ДЛЯ ВЫЧИСЛЕНИЯ ПРИЗНАКОВ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Владимир Александрович Клячин

Доктор физико-математических наук, заведующий кафедрой компьютерных наук и экспериментальной математики,
Волгоградский государственный университет
klchnv@mail.ru, klyachin.va@volsu.ru
https://orcid.org/0000-0003-1922-7849
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Екатерина Владимировна Хижнякова

Старший преподаватель кафедры компьютерных наук и экспериментальной математики,
Волгоградский государственный университет
kate1995yakovleva@gmail.com, yakovleva.e.v@volsu.ru
https://orcid.org/0000-0002-7914-9988
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Аннотация. В статье показано применение индекса Винера к решению одной из задач обработки текстов на естественном языке. Индекс Винера определяется как сумма всех кратчайших расстояний во взвешенном связном графе. Эта величина характеризует сложность графа. В настоящей работе вводятся две нормализации этого индекса. В первом варианте обычный индекс Винера N вершинного связного графа делится на $(N-1)^2$. Во втором варианте индекс Винера евклидова графа делится на сумму расстояний между любой парой не совпадающих вершин. Для применения к задачам обработки текста в статье вводится граф предложений текста: ребро образует пара слов, которые встречаются в тексте в каком-либо предложении. Чтобы вычислять величину индекса Винера для евклидова графа, применяется вложение слов. В статье вкратце описан алгоритм обучения вложению слов Т. Миколова. Дополнительно приводится алгоритм приближенного вычисления остовного дерева с минимальным индексом Винера. Алгоритм основан на минимизации

нового слагаемого при добавлении ребра к построенной части дерева. С целью идентификации неинформативного текста вычисляются 4 признака на основе индекса Винера и его модификаций. Классификация осуществляется стандартными методами машинного обучения.

Ключевые слова: граф, индекс Винера, остовное дерево, вложение слов, машинное обучение.

1. Индекс Винера

Пусть $G = (V, E)$ – связный граф с множеством вершин V и множеством ребер E. Обозначим через $N = |V|$ число вершин графа. Предположим, что каждому ребру $e \in E$ приписан некоторый неотрицательный вес $w(e)$. Если γ – некоторый путь в графе G, то его длина $|\gamma|$ равна сумме весов ребер, из которых этот путь состоит. Обозначим через

$$|pq|_G = \inf_{\gamma} |\gamma|$$

минимальную длину путей, соединяющих вершины $p, q \in V$. Тогда индекс Винера $W(G)$ [16] определяется по формуле

$$W(G) = \frac{1}{2} \sum_{p,q \in V} |pq|_G.$$

Наряду с индексом Кирхгофа, индекс Винера используется для вычисления сложности графа [3]. Если граф G представляет собой простую цепь $G = P_N$ и все веса равны 1, то

$$W(G) = W(P_N) = C_{N+1}^3 = \frac{(N+1)N(N-1)}{6}.$$

Если граф G представляет собой дерево в виде «звезды» S_N, то есть когда ребра соединяют все вершины графа с некоторой одной выделенной, то

$$W(G) = W(S_N) = (N-1)^2.$$

Для полного графа $G = K_N$ очевидно

$$W(G) = W(K_N) = C_N^2 = \frac{N(N-1)}{2}.$$

Наряду с величиной $W(G)$ будем рассматривать приведенный индекс Винера

$$W_n(G) = \frac{W(G)}{(N-1)^2}.$$

Известно [14], что для единичных весов и произвольного дерева T с N вершинами имеет место неравенство

$$W(S_N) \le W(T) \le W(P_N), \tag{1}$$

а для произвольного связного графа с единичными весами – неравенство

$$W(K_N) \le W(G) \le W(P_N).$$

Нормирующий множитель для W_n выбран исходя из того, что $W_n(T) \geq 1$ для дерева T с единичными весами ребер, причем равенство достигается только если это дерево представляет собой «звезду» S_N. Так что нормированная величина W_n меряет отклонение дерева от графа вида S_N. Одной из задач, связанных с введенной величиной, является задача вычисления остовного дерева T графа G с минимальным индексом Винера. Пусть $S(G)$ – множество всех остовных деревьев графа G. Тогда в этой задаче требуется найти $T^* \in S(G)$

$$T^* = \operatorname{argmin}_{T \in S(G)} W(T). \tag{2}$$

Заметим, что эта задача на минимум может иметь не единственное решение. В таком случае предполагается найти хотя бы одно остовное дерево, минимизирующее $W(T)$. В работах [6; 11] доказана NP-полнота этой и подобной задачи для поиска подграфа минимального индекса Винера. Отметим, что в работе [11] задача поиска дерева с минимальным индексом Винера решена для евклидовых деревьев конечного множества точек в выпуклом положении, то есть для вершин выпуклого многоугольника на плоскости. Для евклидовых пространств задача ставится с весами, равными евклидовым расстояниям между вершинами для полного графа. В частности, в [11] доказано, что евклидово дерево с минимальным индексом Винера планарно, то есть ребра дерева попарно не пересекаются.

Пусть $G = (V, E)$ – связный граф, вершинами которого являются точки евклидова пространства $\mathbb{R}^n, n > 1$. В качестве весов ребер выбираем расстояние между соответствующими вершинами ребра графа. Определим величину

$$W_e(G) = \frac{\sum\limits_{p,q \in V} |pq|_G}{\sum\limits_{p,q \in V} |pq|},$$

где $|pq| = |p - q|$ обозначено расстояние между точками $p, q \in V \subset \mathbb{R}^n$. Величина $W_e(G)$ характеризует извилистость кратчайших путей в графе G и всегда $W_e(G) \geq 1$.

Поиск экстремалей для индекса Винера – одна из сложных задач теории графов. Отметим недавнюю работу [15], в которой решается задача поиска графа с минимальным индексом Винера в классе графов Халина [12]. В работе [8] рассматривается задача минимизации произведения $W(G) \cdot H(G)$, где

$$H(G) = \sum_{p,q \in V, p \neq q} \frac{1}{|pq|_G} \quad -$$

так называемый индекс Харари. Там доказано неравенство

$$W(G) \cdot H(G) \geq \frac{1}{4} \left([(N-1)N - M][(N-1)N + 2M] \right),$$

где M – число ребер в графе. При этом равенство имеет место тогда и только тогда, когда диаметр графа G не превосходит 2.

Основная цель настоящей работы состоит в том, чтобы показать возможность использования индекса Винера и его модификаций для вычисления признаков текстов на естественном языке. Мы выделяем четыре таких признака и применяем их для решения задачи определения информативности текста. Поясним последний термин подробнее. В последнее время в медийном пространстве увеличивается количество текстов,

обладающих признаками недостоверного содержания и служащих средством манипуляции общественным сознанием. Деструктивные тексты разнообразны, имеют различную природу, часто характеризуются пересекающимися признаками и с трудом поддаются классификации. К медийным текстам, образующим деструктивные медийные практики, исследователи относят инфодемические сообщения, фейковые тексты и тексты, использующие техники «кликбейтинга» [2]. Разновидностью недостоверных медийных сообщений являются тексты, созданные с целью привлечения внимания читателя и увеличения количества переходов на сайт. Эти тексты строятся по модели информационного сообщения, но их содержание не соответствует заявленному в заголовке тезису (то есть используется технология «кликбейтинга»). Таким образом, текстовые сообщения не соответствующие основной цели медийного дискурса – информированию, являются достаточно разнородными, не имеют четко выраженных критериев выделения, имитируют достоверные сообщения, используя определенную структуру и языковые средства. Кроме того, приемы создания таких текстов постоянно модифицируются, отражая новые практики создания деструктивного контента. Соответственно, важной задачей исследователей является составление параметрических моделей деструктивных текстов, что позволит в дальнейшем решить задачу поиска и идентификации деструктивной информации в разножанровых текстах автоматизированным путем.

Цель нашего исследования – построение обученной модели естественного языка медийных сообщений и анализ точности автоматического распознавания информативных и недостоверных медийных сообщений на ее основе. Существуют различные подходы к автоматическому определению текстов, содержащих признаки дезинформации. В работе [4] авторы отмечают, что статистическая обработка множества поддельных статей позволяет выделить наборы ключевых слов, которые с определенной долей вероятности сигнализируют о возможности, что статья является поддельной. Автоматическое определение «кликбейтинга» на материале английского языка проведено в работах [7; 9; 13]. В них предложена гибридная техника категоризации кликбейтных и некликбейтных статей путем интеграции различных функций, структуры предложений и кластеризации, после чего к набору данных применяются модели машинного обучения, чтобы с помощью соответствующих метрик получить оценки точности распознавания деструктивного текста.

2. Графовая модель текста

Здесь мы вкратце опишем графовую модель текста [1], используемую нами в дальнейшем для вычисления признака текста на основе индекса Винера. Мы предполагаем, что текст написан с использованием конечного алфавита Σ и представляет собой конечную последовательность U символов алфавита Σ. Текст U разбивается на отдельные слова и предложения посредством множества разделителей $D \subset \Sigma$. Пусть V – это список уникальных слов текста – словарь. Построим для пары $u, v \in V$ ребро e, если в тексте найдется предложение, включающее в себя эти два слова. В результате получим граф $G_U = (V, E)$ заданного текста. Надо заметить, что этот граф не обязательно связен. Можно даже сказать, что этот граф, скорее всего, для реальных текстов будет несвязным. Это обстоятельство можно объяснить одним из статистических законов Ципфа – в множестве V подавляющее число слов, которые в тексте встречаются один раз. В указанной выше работе [1] были найдены характеристики графа текста, которые чувствительны к перемешиванию слов текста. При перемешивании текста теряется его

осмысленность, хотя статистически он тот же – частоты слов текста не изменяются. Одной из таких характеристик графа текста является медианное значение степеней вершин графа. Другими словами, если

$$d_1 \leq d_2 \leq ... \leq d_N \quad -$$

степенная последовательность графа G_U, то медианное значение равно $d_{[N/2]}$. Надо сказать, что по аналогии для заданного текста можно построить и другие графы. Например, если в качестве предложений использовать n-граммы, тогда граф будет отражать свойства следования слов друг за другом в тексте. Этот подход исследовался в работе [5] для определения естественного происхождения текста.

Пусть для заданного текста U построен граф $G_U = (V, E)$. Припишем каждому ребру $e \in E$ графа вес $w(e) = 1$. Как было указано выше, граф G может быть несвязным. Индекс Винера вычисляется для связного графа. Поэтому мы будем строить по графу G новый связный граф, добавляя с этой целью новые ребра к множеству E. Это можно сделать различными способами. В нашем исследовании использовался простой метод добавления $k-1$ новых ребра, соединяющих последовательно случайно выбранных k слов по одному из каждой компоненты графа G_U. При этом веса добавленных ребер выбираются равными 1. По построенному графу мы будем вычислять индекс Винера специально сконструированного остовного дерева $T \in S(G_U)$. Это дерево будет приближенным решением вариационной задачи (2). Алгоритм построения этого решения будет описан ниже.

3. Алгоритм приближенного вычисления остовного дерева с минимальным индексом Винера

Для приближенного решения задачи (2) мы совместим два случая. Первый случай – построение приближенного решения для конечного набора точек V евклидова пространства $\mathbb{R}^n, n > 1$ и второй случай – для взвешенного графа. Совмещение производим за счет того, что в первом случае в качестве графа берется взвешенный полный граф с весами ребер, равными их длинам в евклидовом пространстве. Нам понадобится следующая простая формула для индекса Винера произвольного дерева. Пусть T – произвольное N-вершинное дерево с неотрицательными весами ребер $w(e) \geq 0, e \in E$. Для ребра $e = (a, b) \in E$ введем обозначения T_a, T_b для деревьев, получаемых из дерева T удалением ребра e, причем $a \in T_a,\ b \in T_b$. Пусть N_a, N_b – это число вершин в этих деревьях. Ясно, что $N_a + N_b = N$. Нетрудно заметить, что

$$W(T) = \sum_{e=(a,b)\in E} w(e) N_a N_b. \tag{3}$$

Эта формула является основой для получения оценок (1). Получается эта формула из определения индекса Винера посредством изменения порядка суммирования длин ребер – вместо суммирования по всем парам вершин необходимо перейти к сумме по всем ребрам и подсчитать, что число путей, проходящих через ребро $e = (a, b)$, равно в точности $N_a N_b$.

Пусть $G = (V, E)$ – связный граф с N вершинами и $T = (VT, ET)$ – произвольное его поддерево. Выберем в T произвольную вершину $v \in VT$ такую, что найдется ребро

$e' = (v, u) \notin ET$. Присоединив к T это ребро, мы получим новое поддерево $T' = (VT', ET') = (VT \cup \{u\}), ET \cup \{e'\}$. Определим также функцию $h : VT \to \mathbb{R}$ на вершинах $x \in VT$ дерева T по формуле

$$h(x) = \sum_{y \in VT} |xy|_T.$$

Имеет место следующая лемма.

Лемма 1. *Справедлива формула*

$$W(T') = W(T) + h(v) + w(e')|VT|,$$

где $|VT|$ – число вершин в поддереве T.

Доказательство. Из формулы (3) следует

$$W(T') = \sum_{e=(a,b) \in ET} w(e) N'_a N'_b + w(e') N'_v N'_u.$$

Здесь через N'_a, N'_b, N'_v, N'_u, как и выше, обозначены числа вершин в поддеревьях дерева T' после удаления соответствующего ребра. Не ограничивая общности, будем предполагать, что ребра (a, b) заданы так, что $v \in T_b$ (см. рис. 1). Тогда

$$N'_a = N_a, \;\; N'_b = N_b + 1.$$

Поэтому

$$W(T') = \sum_{e=(a,b) \in ET} w(e) N_a N_b + \sum_{e=(a,b) \in ET} w(e) N_a + w(e') N'_v N'_u =$$

$$= W(T) + \sum_{e=(a,b) \in ET} w(e) N_a + w(e') N'_v N'_u.$$

Заметим, что число путей, ведущих из вершины $a \in VT$ в вершину v через ребро (a, b), равно N_a. Так, что

$$\sum_{e=(a,b) \in ET} w(e) N_a = h(v).$$

Замечая, что $N'_v = |VT|, \;\; N'_u = 1$, приходим к требуемому.

Идея алгоритма построения приближенного решения вариационной задачи (2) заключается в том, чтобы к построенной части T искомого дерева добавить новое ребро, дающее, по возможности, минимальный прирост индекса Винера. Заметим, что прирост индекса Винера при добавлении нового ребра $e' = (v, u)$ равен

$$h(v) + w(e')|VT|.$$

При каждом фиксированном $v \in VT$ минимум этой суммы достигается для вершины u, дающей ребро с минимальным весом. Это дает нам следующую последовательность

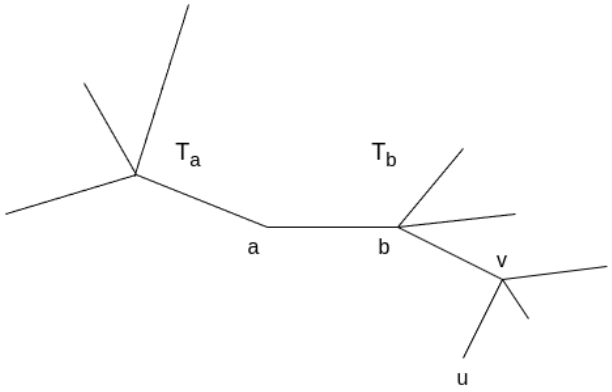

Рис. 1. К доказательству леммы 1

действий. Для каждой вершины $v \in VT$ обозначим через $\mu(v) \in V$ вершину ближайшую, то есть с наименьшим весом $w(e')$, смежную с v и не принадлежащую VT. Теперь на множестве вершин $x \in VT$ определим функцию

$$g(x) = h(x) + w((x, \mu(x))) \cdot |VT|.$$

Искомой вершиной для присоединения нового ребра $(x^*, \mu(x^*))$ к дереву T будет точка минимума функции $g(x)$

$$x^* = \text{argmin}_{x \in VT} g(x).$$

Для поиска точки минимума этой функции строим матрицу P размера $|VT| \times |VT|$, в i-й строке которой содержится величина длины пути от нее до j-й вершины. Затем для каждого $i = 1, 2, ..., |VT|$ вычисляем сумму элементов этой матрицы и величину $w((i, \mu(i)))$. В полученном массиве находим минимальное значение. Так, что номер искомой вершины вычисляется по формуле

$$i^* = \text{argmin}_i \left(\sum_{j=1}^{|VT|} P_{ij} + w((i, \mu(i))) \right). \tag{4}$$

Матрица $P = ||P_{ij}||$ вычисляется посредством алгоритма Дейкстры.

4. Вложение слов

Помимо вычисления индекса графа текста, мы будет вычислять индекс Винера для евклидова вложения графа. Для этого нам понадобится построить отображения некоторого достаточно большого словаря слов естественного языка в многомерное евклидово пространство. Пусть V обозначает множество слов этого словаря, $L = |V|$ – мощность этого множества. Искомое отображение $f : V \to \mathbb{R}^n$ можно строить различными способами. Это одна из проблем компьютерной лингвистики, и в литературе носит название «проблема вложения слов» или «проблема векторизации слов». В качестве примера приведем формулы для вычисления так называемой $TF - IDF$ характеристики, которая позволяет приписывать словам векторы евклидова пространства. Рассмотрим несколько текстов как документов $D_1, D_2, ..., D_M$. Для заданного слова $w \in V$ вычислим следующие величины

$$TF(w, D_i) = \text{ частота слова } w \text{ в документе } D_i.$$

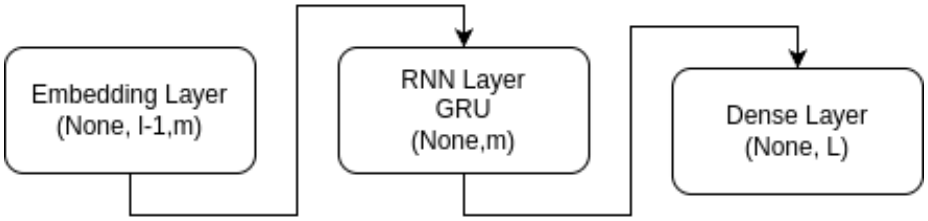

Рис. 2. Архитектура нейронной сети

$$IDF(w) = \log\left(\frac{M}{\text{число документов, содержащих } w}\right).$$

Наконец, характеристика $TF \diamond IDF(w, D_i)$ вычисляется так:

$$TF \diamond IDF(w, D_i) = TF(w, D_i) \cdot IDF(w).$$

Таким образом, получаем вложения слов из коллекции документов в векторное пространство размерности M. Полученное таким образом вложение слов из словаря V может быть использовано во многих задачах, связанных с классификацией текстов. Однако для решения задач выявления информативного текста данная характеристика плохо подходит, поскольку не учитывает структуру текста, последовательность следования слов. Поэтому мы будем использовать другие способы векторизации слов. Один из таких способов построения вложения слов описан в статье [10]. Этот метод основан на обучении нейронной сети, представленной на рисунке 2.

Первый слой Embedding во время обучения сохраняет текущие значения коэффициентов искомой матрицы отображения вложения слов. На вход этому слою подаются унитарные коды последовательности слов текста. Более подробно это выглядит так. Пусть $w_1, w_2, ..., w_l$ – очередная порция последовательно расположенных слов текста. Заменим каждое слово $w_j, j = 1, 2, ..., l$ его унитарным представлением, то есть вектором $v(w_j) \in \mathbb{R}^L$, в котором компоненты равны нулю, кроме одной, чей номер равен номеру слова w_j в словаре. Соответствующая компонента равна 1. Обозначим через $A \in M_{m,L}$ матрицу весов первого слоя. Здесь m – это размерность вложения. Тогда этот слой вычисляет векторы

$$z_j = A \cdot v(w_j), z_j \in \mathbb{R}^m, j = 1, 2, ..., l - 1.$$

Следующий рекуррентный слой выполняет усреднение полученных векторов и выдает один вектор $z \in \mathbb{R}^m$ размерности вложения. Последний плотный слой вырабатывает вектор размерности, равный размеру словаря, то есть вектор $y \in \mathbb{R}^L$. В процессе обучения минимизируется ошибка

$$\delta = \sum |y - v(w_l)|^2,$$

где сумма вычисляется по всем наборам слов $w_1, w_2, ..., w_l$ заданного текста. Таким образом приведенная нейронная сеть обучается предсказывать следующее слово, то есть по словам $w_1, ..., w_{l-1}$ предсказать слово w_l. Результатом обучения данной нейронной сети является матрица весов A первого слоя. Именно эта матрица и задает вложение $f : V \to \mathbb{R}^m$. Столбец с номером j этой матрицы соответствует m координатам вложения j-го слова из V. В нашем исследовании мы воспользовались готовой такой матрицей, взятой с ресурса RusVectōrēs (https://rusvectores.org/ru/models/). Эта матрица извлекается из текстового файла, в каждой строке которого записано слово и координаты его вложения. Размерность вложения $m = 300$, размер словаря $L = 237\ 255$.

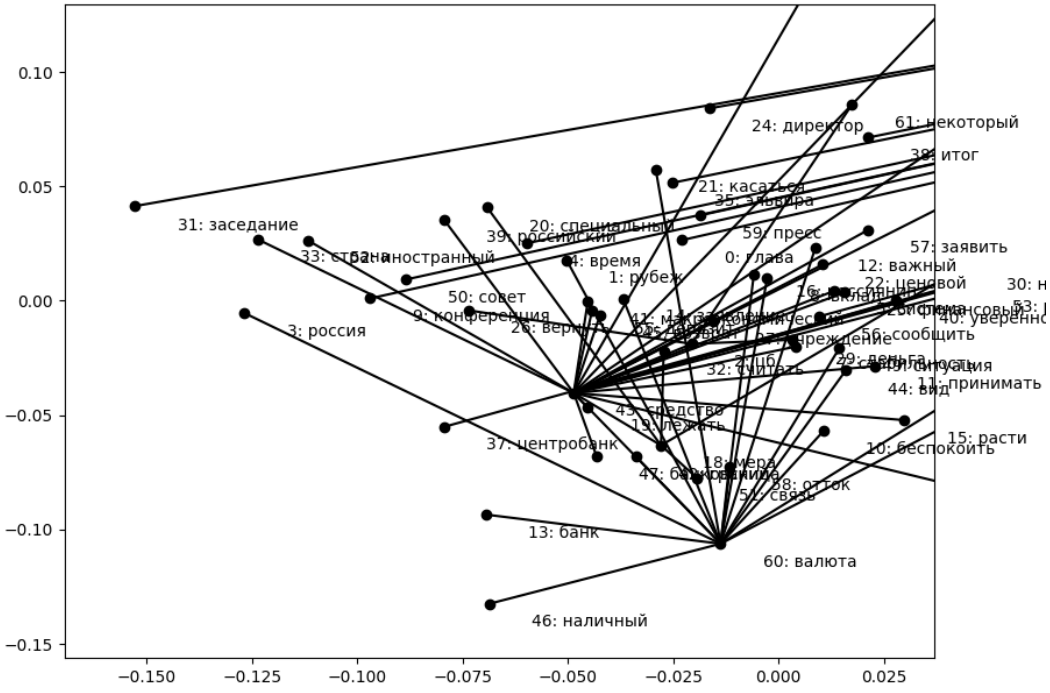

Рис. 3. Пример части дерева, формируемого представленным в § 3 алгоритмом

Часть дерева, построенного приведенным в § 3 алгоритмом для реального текста, с учетом вложения слов, показана на рисунке 3. Это проекция плоского дерева на случайную двумерную координатную плоскость 300-мерного пространства, в которое вложены слова словаря. Номер перед словом на графике соответствует номеру слова в словаре, составленному по исходному тексту.

5. Обучение модели распознавания информативного текста

Для обучения использовался набор размеченных текстов на два класса: 873 информативных текстов и 864 неинформативных текстов. Можно считать выборку сбалансированной. Эти тексты были подготовлены и размечены нашими коллегами с кафедры теории и практики перевода и лингвистики Волгоградского государственного университета. Для каждого текста были вычислены следующие характеристики.

- Первые две величины x_1, x_2 вычисляются для конечного множества точек, которое строится так. Заданный текст U разбивается на отдельные предложения, а каждое предложение – на слова. Далее, используя матрицу A вложения слов, вычисляем координаты каждого слова каждого предложения. Напомним, как это делается. Очередное слово текста (за исключением стоп-слов) имеет номер j в словаре языковой модели, взятой с указанного выше ресурса RusVectōrēs. Матрица A умножается слева на вектор унитарного представления этого слова – то есть на вектор, составленный из нулей и одной единицы в j-й компоненте. Затем для всякого предложения вычисляем геометрический центр вложений его слов. В результате

№	Модель ML	Точность, %
1.	Наивный байесовский классификатор	78
2.	Метод опорных векторов	70
3.	Метод k ближайших соседей, $k = 5$	70
4.	Метод решающих деревьев	69
5.	Квадратичный дискриминатор	75
6.	Случайный лес	75

получается конечное множество, число элементов которого равно числу предложений в тексте. Для полученного конечного множества строим дерево T с помощью алгоритма, описанного в предыдущем разделе, используя в качестве графа полный граф построенного конечного множества геометрических центров вложений слов предложений. Наконец, вычисляем величины $x_1 = W_n(T),\ x_2 = W_e(T)$.

- Третья величина x_3 вычисляется как значение $W_n(T)$ для дерева T, построенного алгоритмом из предыдущего раздела. При этом в качестве графа выбирается граф G_U текста U с весами ребер, равными 1.

- Четвертая величина x_4 вычисляется как величина $W_n(T)$ для минимального остовного дерева T в графе G_U.

Каждой такой четверке приписывается метка $y = 1$, если текст классифицируется как информативный и значение $y = 0$ – в противном случае. Обработанные таким образом наборы текстов генерируют два множества, представленные матрицами $X \in \in M_{K,4},\ Y \in M_{K,1}$, где K – общее число текстов.

Надо заметить, что предварительно из текстов были удалены стоп-слова. С этой целью авторы воспользовались библиотекой NLTK. В качестве моделей машинного обучения выбраны следующие методы:

- Наивный байесовский классификатор с гипотезой нормального распределения.

- Метод опорных векторов.

- Метод k ближайших соседей.

- Метод решающих деревьев.

- Квадратичный дискриминатор.

- Случайный лес.

Обучающий и тестовый наборы данных были поделены в отношении $2 : 1$. Точность предсказания перечисленных моделей представлена в таблице выше. Данные из этой таблицы представляют собой усредненные значения по серии экспериментов. На каждом эксперименте обучающий и тестовый набор данных формировались заново, после случайного перемешивания всего датасета.

Лучший результат для байесовского классификатора вполне объясним. Дело в том, что в качестве признаков используются различные варианты индекса Винера, которые представляют собой суммы длин кратчайших путей. Принимая гипотезу о том, что длины таких путей имеют одинаковое распределение и используя центральную предельную теорему, можно сделать вывод о том, что величины x_1, x_2, x_3, x_4 имеют распределение, близкое к нормальному.

6. Программная реализация

Представленный в статье алгоритм реализован в виде отдельного пакета Word Graph на языке программирования Python. Для поиска кратчайших путей во взвешенном графе использовался пакет scipy.sparse.csgraph. В этом пакете имеются реализации нескольких алгоритмов. В своих расчетах мы использовали реализацию алгоритма Дейкстры. Этот же пакет использовался для поиска минимального остовного дерева во взвешенном графе. Также в этом пакете имеется реализация алгоритма вычисления компонент связности произвольного графа $G = (V, E)$, которая необходима для корректного вычисления индекса Винера. Все матричные представления графов и результатов вычислений на них использовали массивы из пакета numpy. В частности, для поиска решения задачи на минимум (4) была применена функция numpy.argmin пакета numpy.

С целью решения задачи классификации текстов использовалась библиотека машинного обучения scikit-learn. При этом вышеупомянутые массивы X, Y, созданные по набору текстов, были упакованы в файл формата NPZ. Этот формат используется специально для хранения больших numpy массивов на диске, и для работы с таким форматом имеются пакеты как для языка Python, так и для языка программирования Go.

Все тексты программ, файл массивов X, Y, а также частичный набор текстов доступны свободно в репозитории по ссылке: https://github.com/KlyachinVA/WordGraph. Для загрузки исходного кода необходимо в терминале выполнить команду

```
git clone git@github.com:KlyachinVA/WordGraph.git
```

Для использования скачанных текстов программ дополнительно понадобится скачать с ресурса https://rusvectores.org/ru/models/ языковую модель tayga_1_2 и разместить соответствующий файл в каталоге data. Для индексирования слов из этой модели с целью ускорения вычисления вложений слов необходимо установить значение `build_index=True` в вызов метода

```
PreTrainedEmbeddings.from_embeddings_file("./data/tayga_1_2.csv
build_index=False, index_file="./indexes/index-tayga-csv.bin").
```

СПИСОК ЛИТЕРАТУРЫ

1. Григорьева, Е. Г. Исследование статистических характеристик текста на основе графовой модели лингвистического корпуса / Е. Г. Григорьева, В. А. Клячин // Изв. Сарат. ун-та. Нов. сер. Сер.: Математика. Механика. Информатика. — 2020. — Т. 20, № 1. — С. 116–126.

2. Карабулатова, И. С. Специфика лингвистической параметризации деструктивного массмедийного текста с обесцениванием исторической памяти / И. С. Карабулатова, Г. А. Копнина // Медиалингвистика. — 2023. — Т. 10, № 3. — С. 319–335.

3. Медных, А. Д. Циклические накрытия графов. Перечисление отмеченных остовных лесов и деревьев, индекс Кирхгофа и якобианы / А. Д. Медных, И. А. Медных // УМН. — 2023. — Т. 78, № 3. — С. 115–164.

4. Некрасов, Г. А. Разработка поискового робота для обнаружения веб-контента с фейковыми новостями / Г. А. Некрасов, И. И. Романова // Инновационные, информационные и коммуникационные технологии. — 2017. — № 1. — С. 128–130.

5. Попов, В. В. Естественный текст: математические методы атрибуции / В. В. Попов, Т. В. Штельмах // Вестник Волгоградского государственного университета. Серия 2. Языкознание. — 2019. — Т. 18, № 2. — С. 147–158. — DOI: https://doi.org/10.15688/jvolsu2.2019.2.13

6. Хижнякова, Е. В. NP-полнота задачи построения графа с минимальным коэффициентом непрямолинейности / Е. В. Хижнякова // Математическая физика и компьютерное моделирование. — 2023. — Т. 26, № 2. — С. 43–51. — DOI: https://doi.org/10.15688/mpcm.jvolsu.2023.2.4

7. Anand, A. We Used Neural Networks to Detect Clickbaits: You Won't Believe What Happened Next / A. Anand, T. Chakraborty, N. Park // 39-th European Conference on Information Retrieval (ECIR). Aberdeen, United Kingdom, 8–13 April 2017. Lecture Notes in Computer Science (LNCS). — 2017. — Vol. 10193. — P. 541–547.

8. Azjargal, E. Minimum of Product of Wiener and Harary Indices / E. Azjargal, B. Horoldagva, I. Gutman // MATCH Commun. Math. Comput. Chem. — 2024. — № 92. — P. 65–71.

9. Biyani, P. 8 Amazing Secrets for Getting More Clicks: Detecting Clickbaits in News Streams Using Article Informality / P. Biyani, K. Tsioutsiouliklis, J. Blackmer // Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016. — 2016. — Vol. 10193, № 2. — P. 541–547.

10. Efficient Estimation of Word Representations in Vector Space / T. Mikolov, K. Chen, G. Corrado, J. Dean // arXiv preprint arXiv:1301.3781. — 2013. — P. 1–12.

11. Geometric Spanning Trees Minimizing the Wiener Index / A. K. Abu-Affash, P. Carmi, O. Luwisch, J. Mitchell // Algorithms and Data Structures. Springer Nature Switzerland. — 2023. — P. 1–14.

12. Halin, R. Über Simpliziale Zerfallungen Beliebiger / R. Halin // Math. Ann. — 1964. — № 156. — P. 216–225.

13. Identifying Clickbait: A Multi-Strategy Approach Using Neural Networks / V. Kumar, D. Khattar, S. Gairola, L. Y. Kumar, V. Varma // Proceedings of the 41-st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018. — 2018. — Vol. 10193. — P. 1225–1228.

14. Knor, M. Selected Topics on Wiener Index / M. Knor, R. Ŝkrekovski, A. Tepeh // Ars mathematica contemporanea. — 2024. — Vol. 24, № 4. — P. 1–31.

15. Wang, Hedi The Minimum Wiener Index of Halin Graphs With Characteristic Trees of Diameter 4 / Hedi Wang, Kexiang Xu // Electron. J. Math. — 2025. — № 9. — P. 11–22.

16. Wiener, H. Structural Determination of Paraffin Boiling Points / H. Wiener // J. Amer. Chem. Soc. — 1947. — Vol. 69, № 1. — P. 17–20.

ON THE POSSIBILITY OF USING THE WIENER INDEX TO CALCULATE FEATURES OF NATURAL LANGUAGE TEXTS

Vladimir A. Klyachin

Doctor of Sciences (Physics and Mathematics), Head of the Department of Computer Sciences and Experimental Mathematics,
Volgograd State University
klchnv@mail.ru, klyachin.va@volsu.ru
https://orcid.org/0000-0003-1922-7849
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Ekaterina V. Khizhnyakova

Senior Lecturer, Department of Computer Sciences and Experimental Mathematics,
Volgograd State University
kate1995yakovleva@gmail.com, yakovleva.e.v@volsu.ru
https://orcid.org/0000-0002-7914-9988
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Abstract. The article demonstrates the application of the Wiener index to solving one of the problems of natural language text processing. The Wiener index is defined as the sum of all shortest distances in a weighted connected graph. This value characterizes the complexity of the graph. In this paper, two modifications of this index are introduced. In the first version, the usual Wiener index of an N vertex connected graph is divided by $(N-1)^2$. In the second version, the Wiener index of a Euclidean graph is divided by the sum of the distances between any pair of non-coinciding vertices. For application to text processing problems, the article introduces a graph of text sentences: an edge is formed by a pair of words that occur in the text in some sentence. To calculate the value of the Wiener index for a Euclidean graph, word embedding is used. The article briefly describes the algorithm for learning word embeddings by T. Mikolova. In addition, the article provides an algorithm for approximate calculation of a spanning tree with a minimal Wiener index. The algorithm is based on minimizing the new term when adding an edge to the constructed part of the tree. In order to identify uninformative text, 4 features are calculated based on the Wiener index and its modifications. Classification is carried out using standard machine learning methods.

Key words: graph, Wiener index, spanning tree, words embedding, machine learning.