УДК 004.423.23

ИССЛЕДОВАНИЕ ПОДХОДА К ПРЕОБРАЗОВАНИЮ ПРОГРАММ В ВЕКТОРНЫЕ ПРЕДСТАВЛЕНИЯ НА ОСНОВЕ ЦЕПЕЙ МАРКОВА

Горчаков А.В.

МИРЭА - Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: worldbeater-dev@yandex.ru

В статье рассматривается подход к преобразованию текстов программ в векторные представления на основе цепей Маркова, применяемый при построении матрицы попарных сходств программ, подаваемой на вход алгоритму иерархической кластеризации. Кластеризация программ выполняется с целью выявления способов решения уникальных задач по программированию, сгенерированных системой «Цифровой ассистент преподавателя», автоматизирующей массовый курс программирования на языке Python в РТУ МИРЭА. Установлено, что предварительное преобразование текстов программ в векторные представления на основе цепей Маркова, построенных для деревьев абстрактного синтаксиса, позволяет алгоритму иерархической кластеризации находить лучшие разбиения набора текстов программ на кластеры. Для иллюстрации эффективности подхода по сравнению с аналогами приведены визуализации векторных представлений текстов программ на основе цепей Маркова в пространстве низкой размерности, полученные при помощи алгоритма UMAP.

Ключевые слова: анализ текстов программ, анализ программного кода, деревья абстрактного синтаксиса, цепи Маркова, алгоритм кластеризации, алгоритм понижения размерности, векторные представления программ.

A STUDY OF THE APPROACH TO CONVERSION OF PROGRAM TEXTS INTO VECTOR REPRESENTATIONS BASED ON MARKOV CHAINS

Gorchakov A.V.

MIREA - Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, e-mail: worldbeater-dev@yandex.ru

The article discusses an approach to converting program texts into vector representations based on Markov chains, the obtained vectors are then used to construct a matrix of pairwise program text similarities, which is fed to the input of the hierarchical clustering algorithm. Clustering of vector representations of program texts is performed in order to identify the main approaches to solving unique programming exercises, automatically generated by the Digital Teaching Assistant system which automates the massive Python programming course at RTU MIREA. It is revealed that the preliminary transformation of program texts into vector representations based on Markov chains built for abstract syntax trees allows the hierarchical clustering algorithm to find the best partitions of a set of program texts into clusters. To illustrate the effectiveness of the approach, visualizations of vector representations of program texts in a low-dimensional space obtained using the UMAP algorithm are given.

Keywords: text analysis, source code analysis, abstract syntax trees, Markov chains, clustering algorithm, dimension reduction algorithm, vector representations of program texts.

Введение

Цифровизация российской экономики приводит к постоянному расширению круга задач, стоящих перед специалистами с компетенциями в информационных технологиях, прежде всего, перед разработчиками программного обеспечения. Для ускорения и упрощения разработки и отладки новых программных продуктов широко используются инструменты статического анализа кода, в том числе основанные на применении методов и алгоритмов интеллектуального анализа данных и машинного обучения. При помощи подобных инструментов автоматизируется решение таких задач, как поиск схожих участков кода в текстах программ с целью их вынесения в разделяемую библиотеку [1], формирование подсказок средой разработки на основании контекстной информации [2], предсказание названий функций на основе их содержимого, предсказание имён переменных на основании окружающих их синтаксических конструкций [3]. Существует потребность в дальнейшем совершенствовании интеллектуальных анализаторов текстов программ.

В результате расширения круга задач в ИТ-секторе экономики возникает дефицит квалифицированных кадров, чем обусловлена необходимость массовой подготовки ИТ-специалистов в высших учебных заведениях [4]. Организация массовых курсов по программированию в высшей школе влечёт за собой ряд трудностей для преподавателей, включая необходимость разработки разнообразных учебных задач по программированию, необходимость детальной проверки текстов программ и предоставления обратной связи каждому студенту, принимая во внимание проблему заимствований текстов программ без указания авторства при сдаче студентами решений на проверку преподавателю [5].

Для решения данной проблемы в РТУ МИРЭА была разработана система «Цифровой ассистент преподавателя» (ЦАП) [5, 6]. Система была введена в эксплуатацию в 2021 году для автоматизации порождения и проверки уникальных задач по программированию 11 различных типов для более чем 1500 студентов второго курса института информационных технологий РТУ МИРЭА. Ключевой особенностью системы ЦАП является полное исключение плагиата, поскольку каждый студент решает уникальные задачи, порожденные генераторами задач на основе вероятностных грамматик [7]. В весеннем семестре 2022-го года в систему ЦАП студентами было прислано более 66 тысяч текстов программ, 52 тысячи неправильных решений были автоматически отклонены системой. Более 14 тысяч принятых текстов программ, успешно решивших уникальные задачи, выданные ЦАП, в конце семестра были проанализированы при помощи алгоритма иерархической кластеризации для выявления способов решения, использованных студентами курса в течение семестра [8]. Наиболее многочисленные кластеры, выявленные для задач каждого типа, использовались для обучения классификаторов, позволяющих определять способ решения задачи по тексту программы [9]. На основе разработанных классификаторов была реализована и введена в эксплуатацию система учёта учебных достижений ЦАП. В весеннем семестре 2023 года студентам предлагается решить уникальные задачи 11 различных типов всеми известными ЦАП способами для лучшего ознакомления с синтаксисом языка Python и приёмами программирования.

Как показано в работе [8], перед применением алгоритма иерархической кластеризации аналитическая подсистема ЦАП осуществляет преобразование текстов

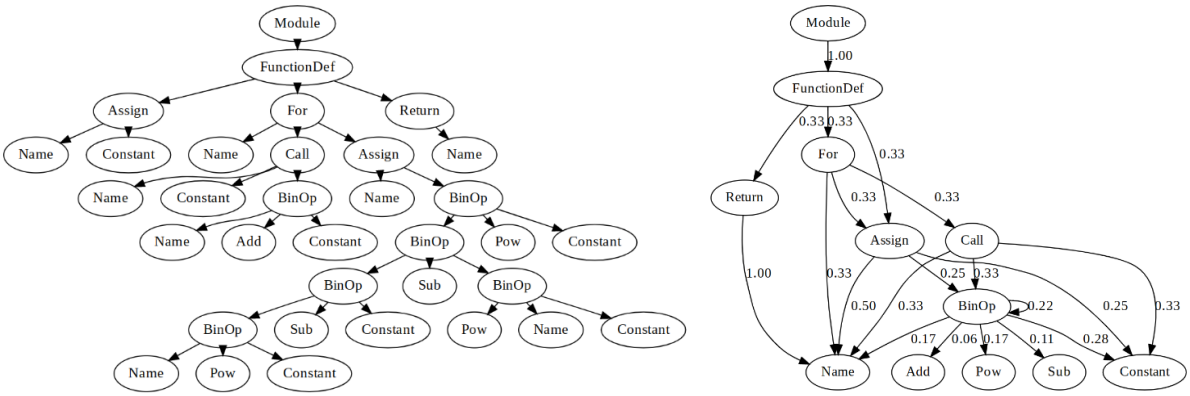
программ в векторные представления на основе цепей Маркова, после чего вычисляет попарные сходства полученных векторов. Однако, в [8] рассмотрен только один подход к парным сравнениям текстов программ при решении задачи кластеризации. В данной статье приведено сравнение различных подходов к парным сравнениям текстов программ с целью численного подтверждения или опровержения преимуществ применения представлений на основе цепей Маркова в задаче выявления способов решения уникальных задач по программированию. Приводится сравнение таких подходов, как:

1. Преобразование текстов программ в векторные представления на основе цепей Маркова, попарные сравнения полученных векторных представлений при помощи меры Йенсена-Шеннона [8].
2. Применение алгоритма word2vec [10] для получения векторов ключевых слов, суммирование векторов ключевых слов для получения векторных представлений текстов программ, попарные сравнения векторных представлений текстов программ при помощи косинусного расстояния [11].
3. Вычисление попарных редакционных расстояний между деревьями абстрактного синтаксиса текстов программ (англ. Tree Edit Distance) [12] для объектов из исследуемого набора данных.

Кластеризация векторных представлений текстов программ выполняется с целью выявления основных способов решения уникальных задач по программированию, автоматически сгенерированных системой ЦАП [5], решения были присланы в систему в 2022-м учебном году, успешно проверены и приняты. Оценка качества полученных алгоритмом иерархической кластеризации разбиений для различных способов построения матрицы попарных сходств проводится на основе метрик чистоты кластеризации [13] и индекса Рэнда [14]. Приведены визуализации взаимного расположения текстов программ на плоскости, полученные алгоритмом UMAP [15].

Преобразование текстов программ в векторные представления на основе цепей Маркова

Для ряда методов и алгоритмов интеллектуального анализа данных и машинного обучения, таких как нейросетевые алгоритмы, необходимо, чтобы объекты, подаваемые на вход, были представлены в виде векторов, компоненты которых кодируют характеристики объектов. Задача преобразования текста программы s в векторное представление состоит в построении отображения $f: S \to \mathbb{R}^h$, где S – множество строк, содержащих тексты программ, а h – размерность целевого векторного пространства. В предложенном в работе [8] методе для текста программы s предлагается сперва построить дерево абстрактного синтаксиса (англ. Abstract Syntax Tree, AST) A, а затем преобразовать полученное дерево в граф переходов между состояниями цепи Маркова для A. Вершинами полученного графа переходов цепи Маркова являются типы вершин дерева абстрактного синтаксиса A, весами рёбер графа являются вероятности наличия связи между вершинами двух типов. Преобразование текста программы s в цепь Маркова осуществляется по Алгоритму 1. Пример дерева абстрактного синтаксиса и соответствующего ему графа переходов между состояниями цепи Маркова представлен на рисунке 1.



(а) (б)

Рис. 1 - Примеры:
а) дерева абстрактного синтаксиса A,
б) соответствующей дереву A цепи Маркова

Алгоритм 1. Построение цепи Маркова для текста программы.

Вход: s – текст программы.

1. **Построить** дерево абстрактного синтаксиса $\mathrm{A} = (V, E)$ для текста программы s.
2. **Удалить** из A вершины $\in \{\mathrm{Load}, \mathrm{Store}, \mathrm{alias}, \mathrm{arguments}, \mathrm{arg}\}$.
3. **Определить** отображение g, ставящее в соответствие вершине $v \in V$ её тип.
4. $M = \emptyset$.
5. $T = \{g(v)\colon v \in V\}$ – множество типов вершин из V.
6. **Для каждого** типа вершины $t \in T$ **выполнять:**
7. $\quad V_d = \{v_d\colon (v_s, v_d) \in E \wedge g(v_s) = t\}$ – потомки вершин типа t.
8. $\quad T_d = \{g(v)\colon v \in V_d\}$ – типы потомков вершин типа t.
9. $\quad$ **Для каждого** типа потомка $t_d \in T_d$ **выполнять:**
10. $\quad\quad \omega = \frac{1}{|V_d|}|\{v_d\colon v_d \in V_d \wedge g(v_d) = t_d\}|$ – нормированное число потомков типа t_d.
11. $\quad\quad M \leftarrow M \cup \{(t, t_d, \omega)\}$.
12. $\quad$ **Завершить цикл.**
13. **Завершить цикл.**
14. **Возвратить** (T, M) – взвешенный ориентированный граф переходов цепи Маркова.

При построении дерева абстрактного синтаксиса в Алгоритме 1 предлагается использовать модуль AST из стандартной библиотеки языка Python. В деревьях абстрактного синтаксиса, построенных для текстов программ, решающих уникальные задачи различающимися способами, могут встречаться различные типы вершин. В некоторых программах могут отсутствовать типы вершин, присутствующие в других программах. Вследствие этого при построении матриц смежности для графов переходов цепей Маркова, полученных Алгоритмом 1, предлагается построить множество H, содержащее все различающиеся вершины, встречающиеся в цепях Маркова для программ из набора S. Такой подход позволяет привести все матрицы смежности к $\mathbb{R}^{m \times m}$, где $m = |H|$ – число различающихся вершин, встречающихся в цепях Маркова для программ из S. Преобразование S в набор векторных представлений V, каждое из которых принадлежит $\mathbb{R}^h$, $h = m^2$, происходит по Алгоритму 2.

Результатом применения Алгоритма 2 к набору текстов программ S является множество V векторных представлений программ, причём $\forall \vec{v} \in V\colon \vec{v} \in \mathbb{R}^h$, $h = m^2$, m – число различающихся вершин, встречающихся во всех цепях Маркова, построенных для программ из S.

Алгоритм 2. Преобразование набора текстов программ в набор векторных представлений.

Вход: S – множество текстов программ для преобразования в векторные представления.

1. $H = \emptyset$ – множество из различающихся вершин, встречающихся в цепях Маркова.
2. $G = \emptyset$.
3. **Для каждого** текста программы $s \in S$ **выполнять:**
4. **Построить** для s цепь Маркова (T, M) по Алгоритму 1.
5. $H \leftarrow H \cup T$ – объединить множество H с множеством вершин цепи Маркова.
6. $G \leftarrow G \cup \{M\}$ – добавить в множество G множество рёбер цепи Маркова.
7. **Завершить цикл.**
8. $V = \emptyset$ – множество векторных представлений текстов программ.
9. $m = |H|$ – число различающихся вершин, встречающихся во всех цепях Маркова.
10. **Для каждого** множества связей цепи Маркова $M \in G$ **выполнять:**
11. **Построить** матрицу смежности $B \in \mathbb{R}^{m \times m}$ для графа (H, M).
12. **Преобразовать** $B \in \mathbb{R}^{m \times m}$ к вектору $\vec{v} \in \mathbb{R}^h$, где $h = m^2$.
13. $V \leftarrow V \cup \{\vec{v}\}$.
14. **Завершить цикл.**
15. **Возвратить** V – множество векторных представлений текстов программ из S.

Для попарных сравнений векторов $\vec{v}_i \in \mathbb{R}^h$ и $\vec{v}_j \in \mathbb{R}^h$ в [8] предлагается использовать меру Йенсена-Шеннона (англ. Jensen-Shannon Divergence, JSD) по формуле:

$$\mathrm{JSD}(\vec{v}_i, \vec{v}_j) = \frac{1}{2}\sum_{k=1}^{h} v_{ik} \log_2 \frac{v_{ik}}{\frac{1}{2}(v_{ik} + v_{jk})} + \frac{1}{2}\sum_{k=1}^{h} v_{jk} \log_2 \frac{v_{jk}}{\frac{1}{2}(v_{ik} + v_{jk})}, \quad (1)$$

где $\vec{v}_i \in \mathbb{R}^h$ и $\vec{v}_j \in \mathbb{R}^h$ – сравниваемые векторные представления, h – число компонент в векторах $\vec{v}_i$ и $\vec{v}_j$, вычисленное в процессе выполнения Алгоритма 2, v_{ik} – k-я компонента $\vec{v}_i$, v_{jk} – k-я компонента $\vec{v}_j$.

Полученное для каждой пары векторов $\vec{v}_i, \vec{v}_j$ значение меры Йенсена-Шеннона (1) записывается в матрицу попарных сходств, причём в случае, если $i = j$, в матрицу записывается единица. Полученная таким образом матрица попарных сходств подаётся на вход агломеративному алгоритму иерархической кластеризации по методу средней связи [8] для выявления кластеров в исследуемом наборе данных, а также алгоритму UMAP [15] для визуализации текстов программ на плоскости.

Численный эксперимент

Рассмотренный метод преобразования текстов программ в векторные представления на основе цепей Маркова, применяемый при построении матрицы попарных сходств программ из S сравнением каждой пары векторов из V друг с другом по мере Йенсена-Шеннона (1), сравнивался со следующими методами:

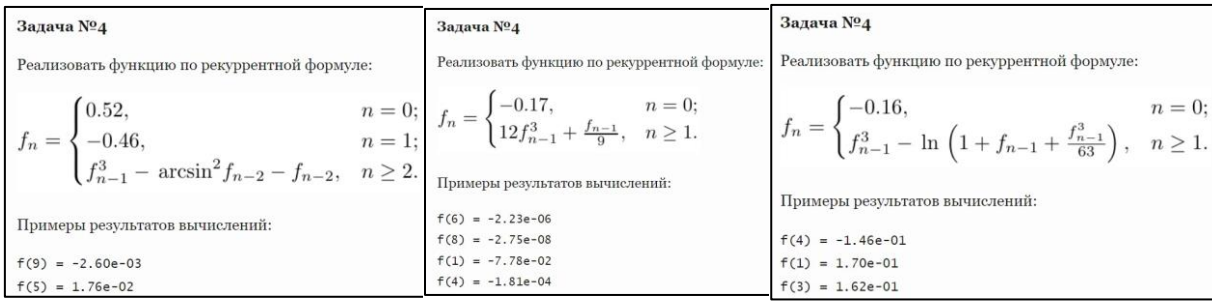
1. Метод, основанный на извлечении ключевых слов языка программирования Python из каждого текста программы $s \in S$, обучении модели word2vec [10] на наборах ключевых слов для преобразования ключевых слов в векторы, суммировании векторных представлений ключевых слов для получения векторного представления программы, попарных сравнениях полученных векторных представлений текстов программ $\vec{v}_i, \vec{v}_j$ при помощи косинусного расстояния по формуле:

$$\mathrm{Cosine}(\vec{v}_i, \vec{v}_j) = 1 - \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\|_2 \left\|\vec{v}_j\right\|_2}, \quad (2)$$

где $\|\vec{v}_i\|_2$ и $\left\|\vec{v}_j\right\|_2$ – евклидовы нормы сравниваемых векторов $\vec{v}_i$ и $\vec{v}_j$, $\vec{v}_i \cdot \vec{v}_j$ – скалярное произведение.

2. Метод, основанный на построении деревьев абстрактного синтаксиса для каждого текста программы с последующим вычислением попарных редакционных расстояний между деревьями по алгоритму [12] при построении матрицы попарных сходств текстов программ.

Для оценки качества и сравнения производительности 3-х рассмотренных методов построения матрицы попарных сходств программ, подаваемой на вход алгоритму иерархической кластеризации, был размечен набор данных, содержащий 40 текстов программ, решающих уникальные автоматически сгенерированные системой ЦАП задачи на перевод математической нотации рекуррентной формулы в код. Примеры условий таких задач представлены на рисунке 2.



(а) (б) (в)

Рис. 2 - Примеры условий задач ЦАП на перевод нотации рекуррентной формулы в код: а,б,в

Все тексты программ из набора данных были ранее проверены и приняты системой ЦАП. При разметке набора текстов программ вручную оценивался выбранный студентом способ решения задачи, под способом здесь подразумевается совокупность использованных в решении синтаксических конструкций. В наборе текстов программ в равных долях представлены решения уникальных задач на перевод рекуррентной формулы в код при помощи: цикла и переменных; цикла и списка с вычисленными значениями; рекурсии; рекурсии с ранним возвратом из функции (англ. early return). Примеры различающихся способов решения уникальных задач на реализацию рекуррентной формулы из анализируемого набора данных представлены на рис. 3.

```
def main(n):
    if n == 0:
        return -0.68
    if n == 1:
        return 0.56
    if n >= 2:
        c = math.ceil(main(n - 2)) ** 2
        return main(n - 1) + 47 * c
```

(а)

```
def main(n):
    mas = [-0.3]
    for i in range(1, n + 1):
        m = mas[i-1] ** 3 / 23
        x = m + mas[i-1] + mas[i-1] ** 2
        mas.append(x)
    return mas[n]
```

(б)

```
def main(n):
    res1 = -0.31
    res2 = -0.44
    for i in range(1, n):
        temp = res1 - res2 ** 3 / 38
        res1 = res2
        res2 = temp
    return temp
```

(в)

Рис. 3 - Примеры способов решения уникальных задач на реализацию рекуррентной формулы из анализируемого набора данных:
а) решение при помощи рекурсии с ранним возвратом из функции,
б) решение при помощи цикла и списка с вычисленными значениями,
в) решение при помощи цикла и переменных

Для оценки качества результатов разбиения набора текстов программ на 4 кластера агломеративным алгоритмом иерархической кластеризации с методом средней связи на основе матрицы попарных сходств, полученной 3-я различными методами, использовались такие меры, как чистота кластеризации [13] и индекс Рэнда [14]. Данные меры позволяют численно оценить, насколько полученное алгоритмом кластеризации разбиение C похоже на эталонное разбиение K, произведённое преподавателем вручную.

Оценка чистоты кластеризации [13] производилась по следующей формуле:

$$\text{Purity}(C, K) = \frac{1}{n}\sum_{c_i \in C} \max_{k_j \in K} |c_i \cap k_j|. \quad (3)$$

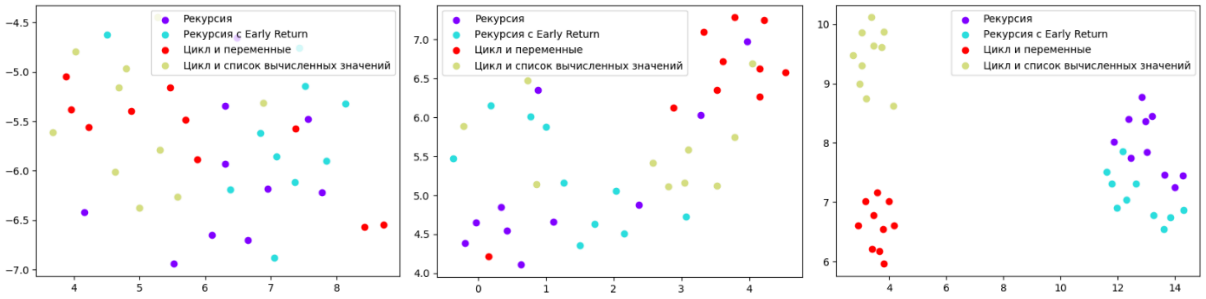
где n – число объектов в исследуемом наборе данных X, $C = \{c_1, c_2, \dots, c_s\}$ – разбиение набора объектов X на s подмножеств в результате кластеризации, $K = \{k_1, k_2, \dots, k_s\}$ – истинные разбиения объектов на s подмножеств.

Индекс Рэнда [14] используется для оценки сходства двух разбиений объектов на кластеры. Как и чистота (3), данная мера может применяться для оценки качества кластеризации C в случае, если известны метки классов объектов K, подлежащих кластеризации. Индекс Рэнда вычислялся по следующей формуле:

$$\begin{gathered} \text{Rand}(C, K) = \frac{2(|A| + |B|)}{n(n-1)}, \\ A = \{(o_i, o_j) : o_i, o_j \in c_{i_1} \wedge o_i, o_j \in k_{j_2}\}, \\ B = \{(o_i, o_j) : o_i \in c_{i_1} \wedge o_j \in c_{i_2} \wedge o_i \in k_{j_1} \wedge o_j \in k_{j_2}\}, \end{gathered} \quad (4)$$

причём $c_i, c_{i_1}, c_{i_2} \in C$, $k_j, k_{j_1}, k_{j_2} \in K$, $i_1 \neq i_2$, $j_1 \neq j_2$, $i \neq j$, $1 \leq i, j \leq n$, где $C = \{c_1, c_2, \dots, c_s\}$ – разбиение набора объектов X на s кластеров, $K = \{k_1, k_2, \dots, k_s\}$ – истинное разбиение объектов из набора X на s подмножеств, n – число объектов в исследуемом наборе данных X, A – множество пар, находящихся в одном и том же подмножестве и в C, и в K, B – множество пар, находящихся в разных подмножествах в C и K. Результаты оценки качества кластеризации для рассмотренных методов построения матриц попарных сходств текстов программ приведены в таблице 1. Для визуальной оценки пригодности рассмотренных методов были получены визуализации взаимного расположения текстов программ в виде точек на плоскости при помощи алгоритма UMAP [15] на основе матриц попарных сходств, визуализации представлены на рисунке 4.

Как показано на рисунке 4 а,б,в, каждый из рассмотренных методов попарных сравнений текстов программ способен отделить решения, в которых используется цикл (решения при помощи цикла и переменных, решения при помощи цикла и списка с вычисленными значениями) от решений, в которых используются условные операторы (рекурсия и рекурсия с ранним возвратом). Подход, основанный на анализе ключевых слов и применении word2vec, не учитывающий иерархическую структуру AST программы, оказывается неспособен привести к выделению в наборе данных 4-х кластеров (см. рис. 4а). Как показано на рисунке 4б и в таблице 1, применение подхода, основанного на вычислении попарных редакционных расстояний между AST текстов программ приводит к разбиению, более похожему на эталонное, по сравнению с показанным на рисунке 4а.



(а) (б) (в)

Рис. 4 - Визуализации, полученные при помощи алгоритма UMAP [15] на основе матриц попарных сходств текстов программ, построенных:

а) преобразованием программ в векторы на основе ключевых слов и word2vec, сравнением векторов косинусным расстоянием (2);

б) вычислением редакционного расстояния между каждой парой AST, построенных для текстов программ;

в) получением векторных представлений на основе цепей Маркова для AST, парными сравнениями векторов мерой Йенсена-Шеннона (1).

Таблица 1. Результаты оценки качества кластеризации для методов построения матрицы попарных сходств.

Метод построения матрицы попарных сходств	Преобразование программ в векторы на основе ключевых слов и word2vec, сравнение векторов косинусным расстоянием (2)	Вычисление редакционного расстояния между каждой парой AST, построенных для текстов программ	Получение векторных представлений на основе цепей Маркова для AST, сравнение векторов мерой Йенсена-Шеннона (1)
Чистота (3)	0.55	0.63	**0.78**
Индекс Рэнда (4)	0.68	0.73	**0.87**

Наконец, преобразование текстов программ в векторные представления на основе цепей Маркова и их последующие попарные сравнения при помощи меры Йенсена-Шеннона (1) позволяет получить разбиения, наиболее похожие на эталонные, как видно на рисунке 4в и в таблице 1. Как видно на рисунке 4в, этот метод позволяет выделить 3 непохожих друг на друга кластера, группируя, но не перемешивая, решения при помощи условных операторов и рекурсивных вызовов (см. рис. 3а).

Заключение

В ходе проведенного исследования было численно оценено качество разбиений текстов программ на кластеры при использовании различных методов построения матриц попарных сходств перед применением алгоритма агломеративной иерархической кластеризации с методом средней связи. Были рассмотрены такие методы, как преобразование текстов программ в векторные представления на основе цепей Маркова [8] с последующими попарными сравнениями полученных векторов при помощи меры Йенсена-Шеннона (1); преобразование программ в векторы на основе ключевых слов и word2vec [10] с последующими попарными сравнениями векторов при помощи косинусного расстояния (2); вычисление редакционного расстояния между каждой

парой деревьев абстрактного синтаксиса [12].

Выявлено, что при решении задач кластеризации уникальных текстов программ с целью выявления использованных в них способов решения задач целесообразно применение Алгоритма 1 и Алгоритма 2 для преобразования текстов программ в векторные представления, целесообразно также применение меры (1) для попарных сравнений полученных векторных представлений на основе цепей Маркова. В настоящее время рассмотренный метод применяется в системе «Цифровой ассистент преподавателя» [8,9] для анализа текстов программ, присылаемых на проверку системе студентами курса Программирования на языке Python РТУ МИРЭА, для автоматизации учёта учебных достижений. Дальнейшие исследования могут быть направлены на исследование возможности применения Алгоритма 1, Алгоритма 2 и формулы (1) в редакторах кода для интеллектуальной генерации подсказок – названий функций, имён переменных, классов, для упрощения разработки сложных программных систем.

Список литературы

1. Rahman W., Y. Xu, F. Pu, J. Xuan, X. Jia, M. Basios, L. Kanthan, L. Li, F. Wu, B. Xu. Clone detection on large scala codebases // In Proceedings of the 2020 IEEE 14th International Workshop on Software Clones (IWSC). – IEEE, 2020. – P. 38-44.

2. Terada K., Watanobe Y. Code completion for programming education based on recurrent neural network // In Proceedings of the 2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA). – IEEE, 2019. – P. 109-114.

3. Li Y., Wang S., Nguyen T. A context-based automated approach for method name consistency checking and suggestion // In Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). – IEEE, 2021. – P. 574-586.

4. Бобров Л.К., Гришняков Б.Ю., Заваруева Н.Н., Крутова Г.Л., Осипов А.Л., Пашков П.М. Развитие дополнительного образования в области ИКТ как путь сокращения дефицита ИТ-персонала // Вестник Саратовского государственного технического университета. – 2014. – Т. 1. – № 74. – С. 89-104.

5. Андрианова Е.Г., Демидова Л.А., Советов П.Н. «Цифровой ассистент преподавателя» в массовом профессиональном обучении для цифровой экономики. // Russian Technological Journal. – 2022. – Т. 10. – № 3. – С. 7-23.

6. Sovietov P.N., Gorchakov A.V. Digital Teaching Assistant for the Python Programming Course // In Proceedings of the 2022 2nd International Conference on Technology Enhanced Learning in Higher Education (TELE). – IEEE, 2022. – P. 272-276.

7. Sovietov P. Automatic generation of programming exercises // In Proceedings of the 2021 1st International Conference on Technology Enhanced Learning in Higher Education (TELE). – IEEE, 2021. – P. 111-114.

8. Демидова Л.А., Советов П.Н., Горчаков А.В. Кластеризация представлений текстов программ на основе цепей Маркова // Вестник Рязанского государственного радиотехнического университета. – 2022. – № 81. – С. 51-64.

9. Горчаков А.В., Демидова Л.А., Советов П.Н. Автоматизированный анализ текстов программ при помощи представлений на основе цепей Маркова и машин экстремального обучения // Вестник Рязанского государственного радиотехнического университета. –

2022. – № 82. – С. 162-163.

10. Mikolov T., Sutskever I., Chen K., Corrado G.S., Dean J. Distributed representations of words and phrases and their compositionality // Advances in Neural Information Processing Systems. – 2013. – Vol. 26. – P. 3111-3119.

11. Usino W., Prabuwono A.S., Allehaibi K.H.S., Bramantoro A., Hasniaty A., Amaldi W. Document similarity detection using k-means and cosine distance // International Journal of Advanced Computer Science and Applications (IJACSA). – 2019. – Vol. 10. – No. 2. – P. 165-170.

12. Zhang K., Shasha D. Simple fast algorithms for the editing distance between trees and related problems // SIAM Journal on Computing. – 1989. – Vol. 18. – No. 6. – P. 1245-1262.

13. Sripada S.C., Rao M.S. Comparison of purity and entropy of k-means clustering and fuzzy c means clustering // Indian journal of computer science and engineering. – 2011. – Vol. 2. – No. 3. – P. 343-346.

14. Rand W.M. Objective criteria for the evaluation of clustering methods // Journal of the American Statistical association. – 1971. – Vol. 66. – No. 336. – P. 846-850.

15. Demidova L.A., Gorchakov A.V. Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm // Journal of Imaging. – 2022. – Vol. 8. – No. 4. – P. 113

References

1. Rahman W., Y. Xu, F. Pu, J. Xuan, X. Jia, M. Basios, L. Kanthan, L. Li, F. Wu, B. Xu. Clone detection on large scala codebases // In Proceedings of the 2020 IEEE 14th International Workshop on Software Clones (IWSC). – IEEE, 2020. – P. 38-44.

2. Terada K., Watanobe Y. Code completion for programming education based on recurrent neural network // In Proceedings of the 2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA). – IEEE, 2019. – P. 109-114.

3. Li Y., Wang S., Nguyen T. A context-based automated approach for method name consistency checking and suggestion // In Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). – IEEE, 2021. – P. 574-586.

4. Bobrov L.K., Grishnyakov B.Yu., Zavarueva N.N., Krutova G.L., Osipov A.L., Pashkov P.M. Razvitie dopolnitelnogo obrazovaniya v oblasti IKT kak put sokrashheniya deficita IT-personala // Vestnik Saratovskogo gosudarstvennogo texnicheskogo universiteta. – 2014. – Vol. 1. – No. 74. – P. 89-104.

5. Andrianova E.G., Demidova L.A., Sovetov P.N. Pedagogical design of a digital teaching assistant in massive professional training for the digital economy. // Russian Technological Journal. – 2022. – Vol. 10. – No. 3. – P. 7-23.

6. Sovietov P.N., Gorchakov A.V. Digital Teaching Assistant for the Python Programming Course // In Proceedings of the 2022 2nd International Conference on Technology Enhanced Learning in Higher Education (TELE). – IEEE, 2022. – P. 272-276.

7. Sovietov P. Automatic generation of programming exercises // In Proceedings of the 2021 1st International Conference on Technology Enhanced Learning in Higher Education (TELE). – IEEE, 2021. – P. 111-114.

8. Demidova L.A., Sovietov P.N., Gorchakov A.V. Clustering of Program Source Text

Representations Based on Markov Chains. – 2022. – No. 81. – P. 51-64.

9. Gorchakov A.V., Demidova L.A., Sovietov P.N. Automated Program Text Analysis Using Representations Based on Markov Chains and Extreme Learning Machines // Vestnik of Ryazan State Radio Engineering University. – 2022. – No. 82. – P. 162-163.

10. Mikolov T., Sutskever I., Chen K., Corrado G.S., Dean J. Distributed representations of words and phrases and their compositionality // Advances in Neural Information Processing Systems. – 2013. – Vol. 26. – P. 3111-3119.

11. Usino W., Prabuwono A.S., Allehaibi K.H.S., Bramantoro A., Hasniaty A., Amaldi W. Document similarity detection using k-means and cosine distance // International Journal of Advanced Computer Science and Applications (IJACSA). – 2019. – Vol. 10. – No. 2. – P. 165-170.

12. Zhang K., Shasha D. Simple fast algorithms for the editing distance between trees and related problems // SIAM Journal on Computing. – 1989. – Vol. 18. – No. 6. – P. 1245-1262.

13. Sripada S.C., Rao M.S. Comparison of purity and entropy of k-means clustering and fuzzy c means clustering // Indian journal of computer science and engineering. – 2011. – Vol. 2. – No. 3. – P. 343-346.

14. Rand W.M. Objective criteria for the evaluation of clustering methods // Journal of the American Statistical association. – 1971. – Vol. 66. – No. 336. – P. 846-850.

15. Demidova L.A., Gorchakov A.V. Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm // Journal of Imaging. – 2022. – Vol. 8. – No. 4. – P. 113.