ВЕСТНИК
НОВОСИБИРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Научный журнал
Основан в ноябре 1999 года

Серия: Информационные технологии 2019. Том 17, № 3

СОДЕРЖАНИЕ

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2019. Том 17, № 3
Vestnik NSU. Series: Information Technologies, 2019, vol. 17, no. 3

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2019. Том 17, № 3
Vestnik NSU. Series: Information Technologies, 2019, vol. 17, no. 3

VESTNIK
NOVOSIBIRSK STATE UNIVERSITY

Scientific Journal
Since 1999, November
In Russian

Series: Information Technologies 2019. Volume 17, № 3

CONTENTS

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2019. Том 17, № 3
Vestnik NSU. Series: Information Technologies, 2019, vol. 17, no. 3

The journal is published quarterly in Russian since 1999
by Novosibirsk State University Press

The address for correspondence

Faculty of Information Technologies, Novosibirsk State University

1 Pirogov Street, Novosibirsk, 630090, Russia

Tel. +7 (383) 363 42 46

E-mail address: inftech@vestnik.nsu.ru

On-line version: http://elibrary.ru

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2019. Том 17, № 3
Vestnik NSU. Series: Information Technologies, 2019, vol. 17, no. 3

УДК 004.932:004.925
DOI 10.25205/1818-7900-2019-17-3-5-17

Восстановление трехмерной геометрии сосудов по данным компьютерной томографии

В. В. Борисенко [1], **Н. С. Серова** [2], **А. М. Чеповский** [3]

[1] *Московский государственный университет им. М. В. Ломоносова*
Москва, Россия

[2] *Первый МГМУ им. И. М. Сеченова (Сеченовский Университет)*
Москва, Россия

[3] *Национальный исследовательский университет «Высшая школа экономики»*
Москва, Россия

Аннотация

Рассматриваются алгоритмы построения триангуляции внутренней поверхности коронарных артерий сердца по данным компьютерной томографии. Точное определение геометрии сосудов необходимо для построения гидродинамической модели кровоснабжения сердца и расчета параметров кровотока с помощью уравнений Навье – Стокса. Для восстановления трехмерной геометрии применяется комбинация двух основных методов: трехмерного алгоритма роста области из семени и ячеечного метода, использующего разбиение пространства на тетраэдры. При этом используется тетраэдрическая сеть, предложенная в работах S. Chan, E. Purisima (1998) и V. Skala (2000), в которой тетраэдры строятся в кубической решетке на общих гранях смежных кубов. Эта сеть строится не на всем пространстве, а только в окрестности границы множества вокселей, вычисленного на первом этапе как результат применения трехмерного алгоритма роста области.

Ключевые слова

компьютерная томография, триангуляция поверхности, марширующие тетраэдры, алгоритм роста области из семени, сглаживание поверхности

Благодарности

Работа выполнена при поддержке РФФИ, грант № 18-29-26012 мк

Для цитирования

Борисенко В. В., *Серова Н. С.*, *Чеповский А. М.* Восстановление трехмерной геометрии сосудов по данным компьютерной томографии // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 5–17. DOI 10.25205/1818-7900-2019-17-3-5-17

Reconstruction of Three-Dimensional Geometry of the Vessels by Computed Tomography Data

V. V. Borisenko [1], **N. S. Serova** [2], **A. M. Chepovskiy** [3]

[1] *Lomonosov Moscow State University*
Moscow, Russian Federation

[2] *Sechenov First Moscow State Medical University (Sechenov University)*
Москва, Россия

[3] *National Research University "Higher School of Economics"*
Москва, Россия

Annotation

We consider algorithms of 3D reconstruction for the internal surface of cardiac vessels. The precise reconstruction of vessel geometry is necessary for the creating a hydrodynamic model of blood supply for the heart and computing vari-

ous parameters of blood flow. To compute a triangulation of blood vessel walls, we use the combination of two methods. At the first stage we apply the 3D seeded region growing algorithm to reconstruct a set of voxels inside vessels. At the second stage we use the isosurface reconstruction algorithm based on the tessellation of 3D space into small tetrahedral cells. We use the tetrahedral mesh, which was proposed in the works of S. Chan, E. Purisima (1998), and V. Skala (2000). Tetrahedra in this mesh are constructed on common faces of adjacent cubes in a cubic lattice, so it fits well with the voxel model. The mesh is constructed only in the neighborhood of the border of voxel set obtained at the first stage as the result of seeded region growing algorithms.

Keywords

computed tomography, isosurface triangulation, tetrahedral mesh, 3D seeded region growing

Acknowledgements

The research was supported by the RFBR grant no. №18-29-26012 мк

For citation

Borisenko V. V., Serova N. S., Chepovsky A. M. Reconstruction of Three-Dimensional Geometry of the Vessels by Computed Tomography Data. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 5–17. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-5-17

Введение

Ишемическая болезнь сердца является одним из самых распространенных заболеваний. Ее основной причиной является нарушение кровоснабжения сердечных мышц, связанное, как правило, с сужением коронарных артерий (стеноз). В настоящее время имеются достаточно эффективные методы ее лечения, главным из которых является установка расширителей сосудов (стентов). Для этого в коронарную артерию вводится катетер. Перед проведением подобной операции необходимо вначале убедиться в ее необходимости. Одним из основных критериев является так называемая величина FFR (fraction flow reserve), показывающая, насколько уменьшается кровоток в месте сужения артерии. FFR определяется как отношение давления крови в точках до и после сужения. В медицине используется так называемый «золотой стандарт» измерения FFR, использующий инвазивный метод – введение катетера с датчиком давления в артерию. Несмотря на то, что в настоящее время эта процедура хорошо отработана и может быть совмещена с установкой стента, желательно все-таки иметь возможность определить FFR каким-либо неинвазивным методом, например, используя данные компьютерной томографии области сердца. Это поможет избежать введения катетера в коронарную артерию в случаях, когда нет необходимости в установке стента.

При исследовании кровеносных сосудов с помощью компьютерной томографии в кровь пациента через вену вводится контрастное вещество, резко увеличивающее степень поглощения рентгеновских лучей кровью. Томографические срезы, полученные компьютерным томографом, представляют собой матрицы размером 512×512, содержащие двухбайтовые целые числа со знаком. Их величины обозначаются через HU (housfield units), они отражают степень поглощения рентгеновских лучей тканью пациента в данной точке среза. Величина HU воздуха равна –1 024, воды – 0, крови без контрастного вещества – около 40–50. При введении контрастного вещества в кровь ее HU резко возрастает до нескольких сотен, и кровеносные сосуды становятся хорошо различимыми на фоне окружающих тканей.

Данная работа является частью проекта создания компьютерной модели кровоснабжения сердца для конкретного пациента. Основная идея состоит в том, чтобы определить всевозможные параметры кровотока методами гидродинамики, применяя численные методы решения уравнений течения жидкости – уравнения Навье – Стокса, уравнение состояния и др. Эти численные методы предполагают построение геометрии расчетной области в виде задания ее триангуляции, затем вычисление расчетной сетки и численное решение по этой сетке дифференциальных уравнений в частных производных. Предполагается использовать систему FlowVision [1; 2], успешно применяемую в расчетах, необходимых для кораблей, турбин, летательных аппаратов и т. п. Для ее применения в области медицинской диагностики при изучении системы кровообращения требуется на первом шаге точно воссоздать трехмерную

геометрию кровеносных сосудов, в нашем случае аорты и коронарных артерий. В данной работе рассматривается именно эта задача.

Томографические срезы представляют собой последовательность параллельных сечений тела пациента. Современный томограф позволяет за очень короткое время (около 0,35 с) получить серию срезов с очень высоким разрешением: обычно сканируется область размером 16 см по оси Z (вертикальной оси тела пациента) с шагом 0,25 мм по вертикали, при этом горизонтальные размеры пикселя по осям X и Y равны 0,5 мм. Компьютерный томограф практически мгновенно выдает серию из 640 срезов в области сердца. Эти срезы определяют функцию рентгеновской плотности ткани человеческого тела в трехмерном пространстве $d(x, y, z)$ (в промежутках между срезами она вычисляется с помощью интерполяции). Стандарт DICOM (digital imaging and communication in medicine) определяет следующие направления координатных осей: ось X направлена от правой части тела пациента к левой (Right → Left), ось Y – от передней стороны тела к задней (Anterior → Posterior), ось Z – от ног к голове (Feet → Head).

Задачу трехмерной реконструкции в самом простом случае можно сформулировать как построение триангуляции поверхности, заданной уравнением

$$d(x, y, z) = t,$$

где $d(x, y, z)$ – функция плотности, t – пороговое значение (threshold). В зависимости от выбора значения t мы восстанавливаем разные структуры человеческого тела – например, при $t = 200$ получается поверхность не очень твердых костей, при $t = 1\,000$ – зубов, при $t = -200$ – опухолей в легких и т. п. При восстановлении внутренней поверхности кровеносных сосудов в случае использования контрастного вещества применяется пороговое значение t в пределах HU от 150 до 400. Отметим также, что практически всегда восстановление такой поверхности выполняется не во всем пространстве, а только в интересуемой области, в медицине для этого используется термин ROI (region of interect) или, реже, VOI (volume of interest). ROI обычно задается либо наборами контуров на каждом срезе, либо наборами битовых маск (чаще всего используются оба метода одновременно).

Типы трехмерных алгоритмов

Существует два основных класса алгоритмов трехмерной реконструкции. Первый класс состоит из так называемых ячеечных методов, среди них наиболее известен метод «марширующих кубов» ([3], 1987). В таких методах пространство разбивается на многогранники маленького размера (в несколько пикселей на томографических срезах), например, на маленькие кубики в методе «марширующие кубы» или тетраэдры в методах типа «марширующие тетраэдры» (методы MT5, MT6 и др.). (Под словом «тетраэдр» всюду в этой статье подразумевается любая треугольная пирамида, не обязательно правильный тетраэдр.) Значение функции

$$f(x, y, z) = d(x, y, z) - t$$

вычисляется в вершинах этих многогранников. Рассматриваются ребра каждого многогранника. Если в вершинах ребра функция f принимает значения разных знаков, то при допущении, что в точках ребра функция меняется линейно, вычисляется точка на ребре, в которой функция принимает значение 0. Полученные точки на ребрах соединяются одним или несколькими треугольниками внутри рассматриваемого многогранника. Объединяя полученные таким образом точки и треугольники, получаем триангуляцию всей поверхности (см. далее, рис. 4 для случая тетраэдрических ячеек).

Второй класс алгоритмов состоит из так называемых методов «роста области из семени» (seeded region grow), мы рассматриваем их трехмерные варианты. Алгоритм роста области из семени строит множество S вокселей внутри интересуемой области, которая может задаваться пороговым значением t функции $d(x, y, z)$ в трехмерном пространстве (область состоит

из вокселей, значения функции d в которых не ниже порогового), а также какими-либо другими признаками. В интересуемой области выбирается начальный воксель («семя»), он добавляется к множеству S. Затем программа в цикле добавляет к множеству S воксели, соседние с уже содержащимся в S и удовлетворяющие критерию принадлежности к области. В процессе работы алгоритм использует какую-либо динамическую структуру D, содержащую воксели, которые уже были добавлены к множеству S, но соседи которых еще не были рассмотрены. Алгоритм заканчивается, когда D становится пустой. В качестве D можно использовать очередь, стек, список, но наиболее удобной структурой такого рода является структура Deque (double ended queue – двусторонняя очередь), содержащаяся в стандартной библиотеке языка C++ (std::deque). Вот запись трехмерного алгоритма роста области из семени на псевдокоде:

```
S: множество вокселей; D: очередь вокселей (deque)
p0: начальный воксель (“семя”)
S.добавить(p0)
D.добавить в конец (p0)
цикл пока D не пуста выполнять
|   p = D.начало; D.удалить начало
|   цикл для каждого соседа n вокселя p выполнить
|   |   если n не принадлежит множеству S и n удовлетворяет
|   |   |          критерию принадлежности к области, то
|   |   |   S.добавить(n)
|   |   |   D.добавить в конец(n)
|   |   конец если
|   конец цикла
конец цикла
```

В качестве соседей вокселя p в трехмерном пространстве берутся либо 6 соседей, одна из координат которых отличается ровно на единицу от соответствующей координаты p, либо 26 соседей, представляющих собой все воксели куба размером 3×3 с центром p, отличные от p.

Достоинства и недостатки трехмерного алгоритма роста области из семени

Метод роста области из семени имеет совсем простую программную реализацию. Но главным достоинством этого метода является то, что он вычисляет связную область, содержащую начальный воксель. Он хорошо подходит для построения множества вокселей внутри кровеносных сосудов, поскольку такое множество невозможно задать только на основе порогового значения. Например, костная ткань (плотность которой варьируется в очень широких пределах) может иметь такие же значения плотности HU, как и кровь при использовании контрастного вещества, поэтому ячеечными методами отделить кровеносную систему от костной ткани невозможно. Но алгоритм роста области добавляет к трехмерной модели лишь те воксели, которые связаны с начальным вокселем («семенем»). Если начальный воксель находится внутри кровеносного сосуда, то и все добавленные воксели будут находиться внутри кровеносной системы (учитывая, что кровеносная система замкнута) – алгоритм роста не должен «протекать» в пространство вне сосудов. Таким образом, костная ткань автоматически исключается из трехмерной модели.

После построения множества вокселей остается вычислить триангуляцию его поверхности. Эта процедура также несложная – рассматриваются те грани «внешних» вокселей из множества S, к которым не примыкают другие воксели из S. Каждая такая прямоугольная грань разбивается диагональю на 2 треугольника, и получаем требуемую триангуляцию поверхности множества S.

Еще одно достоинство алгоритма роста области состоит в том, что несложно ограничить рост области, заполнив контуры на некоторых срезах и запустив алгоритм роста повторно [1].

На рис. 1 изображены результаты восстановления области сердца, полученные трехмерным алгоритмом роста области. Левая модель получена ростом из начального вокселя, расположенного внутри аорты. В правой модели используется задание области интереса (ROI), которая отделяет часть аорты и коронарные артерии от остальной части кровеносной системы; правая модель представляет собой часть модели, изображенной слева.

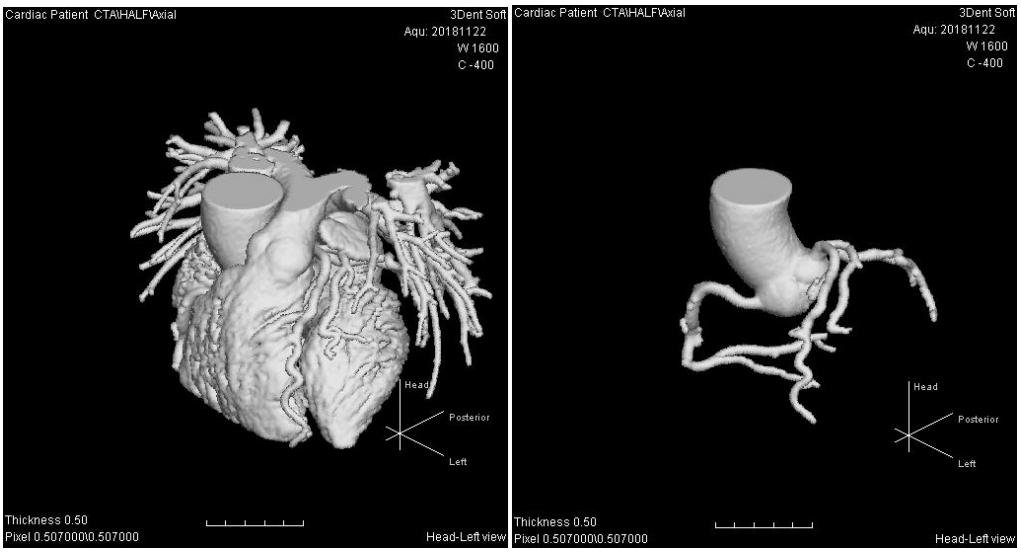


Рис. 1. Результаты трехмерного восстановления области сердца. Левая модель получена алгоритмом роста из начального вокселя, расположенного внутри аорты. Правая модель включает часть аорты и коронарных артерий, она получена из левой модели применением того же алгоритма роста при установленных ограничениях

Fig. 1. The results of 3D reconstruction of the heart area. The left model is obtained by the 3D region growing algorithm from the initial voxel inside the aorta. The right model includes a part of aorta and coronal arteries, it obtained from the left model using the same algorithm, but taking into account the restrictions that separates coronal arteries from other parts of the circulatory system

Но у алгоритма роста области из семени есть и недостатки. Первым из них является большее время работы, чем у ячеечных алгоритмов, поскольку рассматриваются все воксели трехмерной модели, которая строится, а их число может быть очень большим. В ячеечных алгоритмах можно выбирать размер шага (т. е. размер многогранников, на которые разбивается пространство), но в алгоритме роста области шаг всегда минимально возможный – он равен размеру одного вокселя. Кроме того, ячеечные алгоритмы добавляют к модели лишь треугольники, расположенные на поверхности модели, а алгоритм роста области проходит по всему ее объему. Однако для наших целей этот недостаток не столь существенен, поскольку в любом случае шаг нужно выбирать минимально возможным, чтобы достичь наилучшего разрешения, так как коронарные артерии могут иметь очень небольшой поперечный размер. К тому же современные компьютеры позволяют достичь очень высокой скорости, и вдобавок отсутствуют существенные ограничения на объем оперативной памяти.

Отметим, что второй недостаток алгоритма роста области, увы, не позволяет напрямую его использовать при построении триангуляции для последующего гидродинамического рас-

[1] Подробно этот метод описан в работе: *Борисенко В. В.*, *Веселова Т. Н.*, *Терновой С. К.*, *Чеповский А. М.* Реконструкция трехмерной геометрии коронарных артерий // Фундаментальная и прикладная математика. 2018. Т. 22 (в печати).

чета. Дело в том, что поверхность построенной модели получается негладкой, она составлена из поверхностей кубиков, представляющих собой воксели полученного множества (рис. 2, 3). Если бы цель состояла только в том, чтобы получить изображение поверхности, то эта проблема была бы не столь существенна – трехмерный алгоритм Фонга с использованием нормалей в вершинах триангуляции позволяет визуально сгладить изображаемую поверхность (см. рис. 2). Но при гидродинамическом расчете поверхность нуждается в реальном, а не только визуальном сглаживании. Алгоритмы сглаживания из класса Subdivision Surface (дробление поверхности), такие как алгоритм Лупа [4] или Кетмулла – Кларка [5], также мало подходят для нашей цели, поскольку они, лишь сглаживая углы, в целом не меняют форму поверхности, к тому же в медицинской диагностике желательно избежать даже минимального изменения формы вычисляемой поверхности.

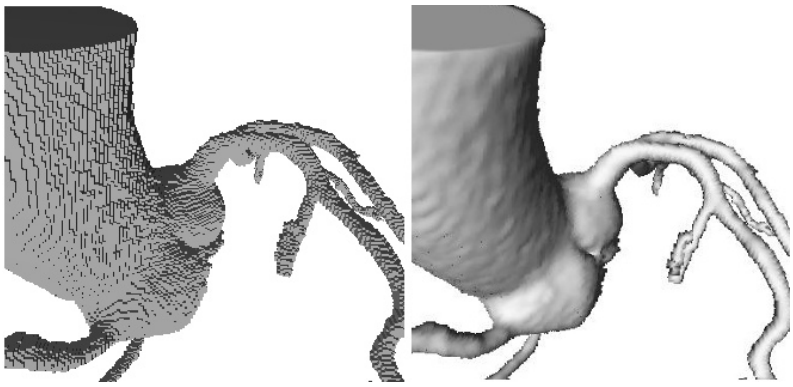


Рис. 2. Триангуляция поверхности множества вокселей, полученного трехмерным алгоритмом роста области. На левом изображении грани вокселей закрашиваются как плоские многоугольники (flat shading). Правое изображение той же модели получено с помощью алгоритма Фонга (smooth shading): использование нормалей к поверхности, вычисленных как градиент функции плотности, позволяет визуально сгладить триангуляцию без изменения ее формы

Fig. 2. Surface triangulation of the set of voxels, obtained by the 3D region growing algorithm. On the left image, voxel faces are drawn as flat polygons (flat shading). The right image of the same model is obtained using the Phong algorithm (smooth shading): the use of surface normals, calculated as a gradient of the density function, visually smooths the triangulation without changing its shape

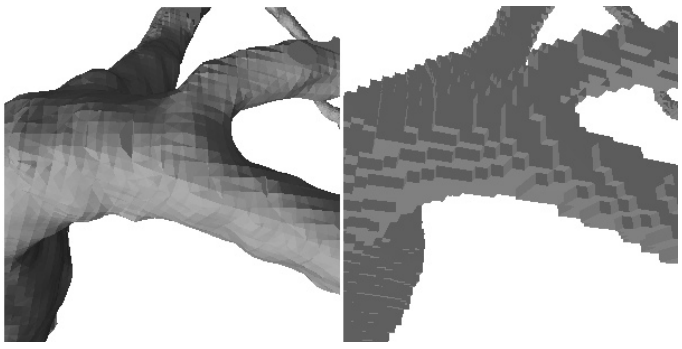


Рис. 3. Разница между ячеечным алгоритмом триангуляции и трехмерным алгоритмом роста области. Крупно показан один и тот же фрагмент модели, созданный разными алгоритмами

Fig. 3. The difference between the cell triangulation algorithm and the 3D region growing. The same fragment of two models, created by different algorithms, is shown in large

Отметим, что ячеечные алгоритмы строят более гладкую поверхность, поскольку построенные в них треугольники получаются приблизительно перпендикулярными вектору градиента функции плотности, в отличие от метода роста области, в котором плоскости треугольников параллельны координатным плоскостям. Решение проблемы состоит в последовательном применении двух алгоритмов. Сначала с помощью алгоритма роста области из семени вычисляется множество вокселей, составляющее трехмерную модель интересующего участка кровеносной системы [2]. На втором этапе используется ячеечный алгоритм вычисления триангуляции, причем он применяется не ко всему пространству, а только к окрестности поверхности модели, вычисленной на первом этапе.

Тетраэдрические сети

Самым первым ячеечным алгоритмом построения триангуляции изоповерхности был метод марширующих кубов [3]. Он неплох, если речь идет только об изображении трехмерного объекта. Однако быстро выяснился его серьезный недостаток: в некоторых случаях вершины триангуляции на ребрах куба можно соединить треугольниками разными способами. При несогласованном выборе треугольников внутри смежных кубов в построенной триангуляции сплошного объекта могут образоваться дыры. Всего с точностью до симметрии имеется 14 вариантов расположения вершин триангуляции на ребрах куба, неоднозначность возможна в 5 случаях. Вероятность этих ситуаций мала, и они почти не влияют на визуальное восприятие объекта, но если речь идет о точном воспроизведении топологии, то подобное недопустимо. Этого недостатка лишены ячеечные методы, в которых пространство разбивается на тетраэдры. С точностью до симметрии имеется лишь 2 варианта расположения вершин триангуляции на ребрах тетраэдра. В случае трех точек они соединяются одним треугольником, в случае четырех – двумя (рис. 4), других случаев не бывает.

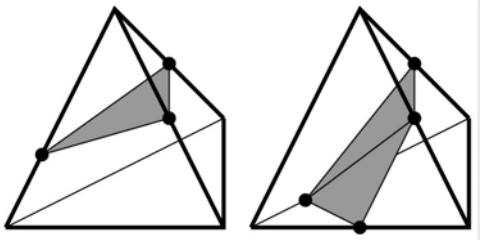


Рис. 4. Построение триангуляции внутри тетраэдрической ячейки

Fig. 4. Construction of triangulation inside a tetrahedral cell

При использовании тетраэдрических ячеек возникает проблема: как выбрать наилучшую тетраэдрическую сеть? Дело в том, что разбить трехмерное пространство на правильные тетраэдры невозможно (в отличие от плоскости, которая разбивается на правильные треугольники). Это означает, что априори «самой лучшей» тетраэдрической сети не существует. Обычно под качеством сети понимают средний аспект входящих в нее тетраэдров. Аспектом тетраэдра называется отношение радиусов описанной и вписанной сфер. Аспект правильного тетраэдра в точности равен 3, у любого тетраэдра, отличного от правильного, аспект строго больше трех, так что аспект может служить мерой того, насколько тетраэдр отличается от правильного. Также желательными свойствами тетраэдрической сети являются равенство входящих в нее тетраэдров, инвариантность сети относительно сдвигов в трех независимых направлениях, удобство программирования и т. п.

Самым простым методом построения тетраэдрической сети является разбиение пространства сначала на кубы, а затем разбиение каждого куба на тетраэдры. Не добавляя дополнительных вершин, куб можно разбить на тетраэдры 6 способами. В методе MT5 (см. [6]) куб разбивается на 5 тетраэдров, из которых один правильный с объемом $^1/_3$, он образован скрещивающимися диагоналями граней куба; остальные 4 тетраэдра имеют объем $^1/_6$, каждый

[2] См.: *Борисенко В. В. и др.* Реконструкция трехмерной геометрии коронарных артерий.

из них образован тремя перпендикулярными ребрами куба, выходящими из одной вершины. Это разбиение изображено слева на рис. 5.

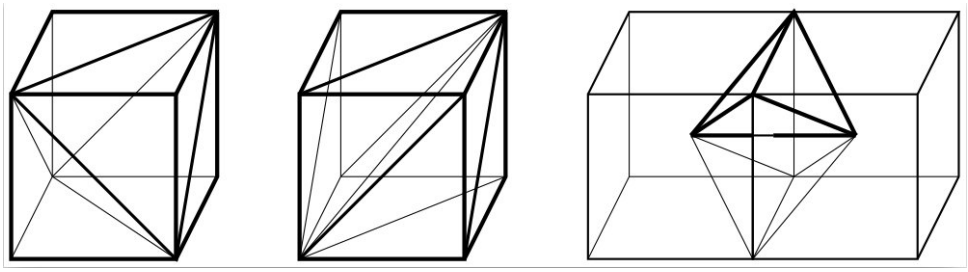


Рис. 5. Разбиение куба на тетраэдры (слева и в середине на 5 и 6 тетраэдров в методах MT5 и MT6 соответственно; справа показан процесс построения тетраэдров в методе Скала)

Fig. 5. The various partitions of a cube into tetrahedra. The partitions into 5 and 6 tetrahedra are shown on the left and in the middle. The right shows the process of constructing tetrahedra in the Skala method

В методе MT6 (см. [7]) куб разбивается на 6 равных тетраэдров (см. рис. 5, в центре). Сеть MT6 удобна тем, что она инвариантна относительно сдвигов на вектор любого ребра куба. Однако для нашей цели лучше всего подходит сеть, предложенная в работе S. L. Chan, E. O. Purisima [8]; она стала широко известна также и благодаря статье V. Skala [9], в литературе ее часто называют сетью Скала. Тетраэдры этой сети изображены на рис. 5, справа, они строятся на границах смежных кубов кубической решетки. В сети Скала все тетраэдры равны между собой (это тетраэдры Кокстера, сеть инвариантна при отражениях пространства относительно любых граней тетраэдров). Сеть Скала имеет наилучший аспект 3.162 среди всех известных на данный момент тетраэдрических сетей. Для сравнения: аспект сети MT6 равен 4.182, средний аспект сети MT5 равен 3.878.

Использование тетраэдрической сети Chan – Purisima – Skala для сглаживания 3D-модели, построенной алгоритмом роста области

Помимо остальных прекрасных свойств сети Chan – Purisima – Skala (равенство входящих в нее тетраэдров, наилучший аспект среди всех известных сетей, инвариантность относительно сдвигов и отражений), она обладает свойством, крайне ценным для нашей задачи: в ней тетраэдры строятся на гранях смежных кубов кубической решетки. Справа на рис. 5 изображены 4 тетраэдра, построенные на общей грани двух смежных кубов, расположенных горизонтально, аналогично строятся тетраэдры на общей грани соседних кубов, расположенных вертикально (и во всех трех независимых направлениях). Вершины построенных тетраэдров находятся в вершинах исходной кубической решетки, а также в центрах кубов. Если размер ребра куба равен 1, то длины ребер тетраэдров равны 1 (два ребра, совпадающие с ребрами куба) и $\sqrt{3/2} \approx 0{,}866$ (остальные 4 ребра, равные половине большой диагонали куба). Если каждый куб решетки представляет воксель в пространстве, то построенная на этих кубах тетраэдрическая сеть Скала позволяет достичь субпиксельной точности, которая необходима для точного восстановления геометрии кровеносных сосудов.

Ключевой момент состоит в том, что тетраэдрическая сеть строится не во всем пространстве, а только в окрестности границы трехмерной воксельной модели, полученной алгоритмом роста области из семени. Благодаря этому при последующем применении ячеечного метода трехмерного восстановления мы не добавляем нежелательных объектов к модели, а лишь уточняем форму ее поверхности.

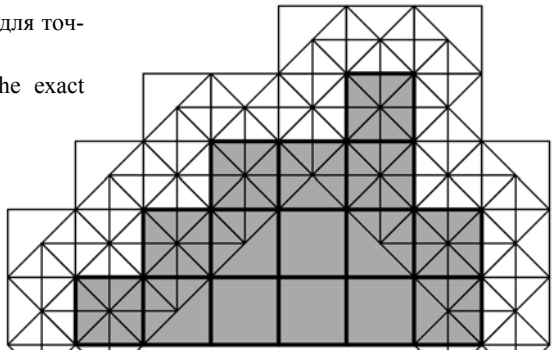
Точный алгоритм построения «уточняющей» тетраэдрической сети следующий. Обозначим через M исходную модель, полученную трехмерным алгоритмом роста области из семе-

ни, она состоит из кубических вокселей. Каждый воксель пространства – это куб с 8 вершинами. Какие-то вершины вокселей из M могут лежать на поверхности модели M, какие-то нет. Построим множество вокселей S следующим образом: произвольный воксель v трехмерного пространства добавляется к S, если хотя бы одна из его вершин лежит на поверхности модели M. Таким образом, множество S будет содержать как некоторые воксели из модели M, так и некоторые воксели, не входящие в M, но «касающиеся» ее границы.

После определения множества вокселей S строится уточняющая тетраэдрическая сеть: для любых двух соседних вокселей из S, имеющих общую грань, строятся 4 тетраэдра сети (см. рис. 5, справа). Схематично процесс построения сети изображен на рис. 6 в двумерной проекции (трехмерный рисунок получился бы слишком громоздким).

Рис. 6. Построение «уточняющей» тетраэдрической сети для точного вычисления сглаженной границы воксельной модели

Fig. 6. Construction of a refining tetrahedral mesh for the exact calculation of the smoothed boundary of the voxel model



На рис. 6 темными квадратами показаны воксели исходной модели (множество M), светлыми – воксели из построенного множества S, не принадлежащие M, но имеющие вершину на границе M. Каждые 2 соседних вокселя из множества S определяют 4 тетраэдра, которые строятся на их общей грани, проекции этих тетраэдров нарисованы тонкими линиями.

После построения уточняющей тетраэдрической сети запускается алгоритм трехмерного восстановления изоповерхности ячеечного типа. Результат применения алгоритма для модели тора показан на рис. 7.

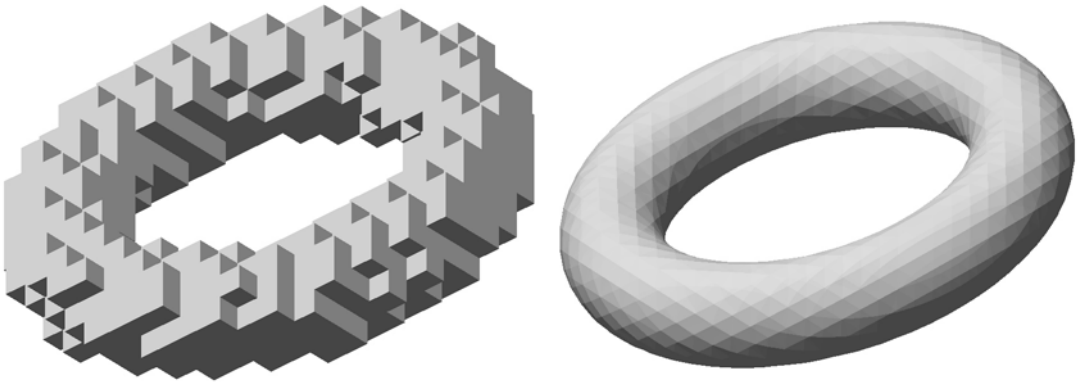


Рис. 7. Уточнение границы для воксельной модели тора, полученной трехмерным алгоритмом роста области (слева). Справа показана модель, полученная вычислением границы с использованием уточняющей сети Chan – Purisima – Skala. Для изображения используется OpenGL и плоское закрашивание треугольников (Flat Shading)

Fig. 7. The refinement of the boundary for the voxel model of the torus obtained by the 3D region growing algorithm (left). On the right is the model obtained by calculating the boundary using the Chan – Purisima – Skala refining mesh. OpenGL and Flat Shading are used for the image

В правой части рис. 7 специально используется плоский режим закрашивания треугольников, чтобы они были лучше видны на изображении. Если применить алгоритм Фонга, использующий нормали в вершинах триангуляции для сглаживания изображения, то результат получается визуально совершенно гладким. На рис. 8 уточненная модель изображена слева как проволочный каркас (Wireframe), справа используется алгоритм Фонга.

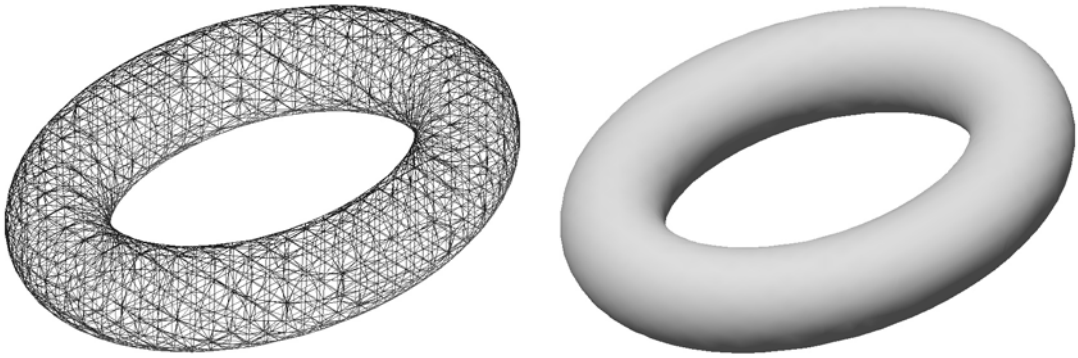


Рис. 8. Изображение уточненной модели тора как проволочного каркаса (слева) и с помощью алгоритма Фонга, использующего нормали для сглаживания поверхности (справа)

Fig. 8. Image of the refined model of the torus as a wire frame (left) and by using normals to the surface and Phong algorithm for smooth shading (right)

На рис. 9. показан результат применения уточняющего алгоритма для компьютерной модели кровеносного сосуда. Отметим, что толщина сосуда в этом примере минимальна (3–4 вокселя), а исходная воксельная модель имеет форму, далекую от цилиндрической, однако уточненная триангуляция получается достаточно гладкой.

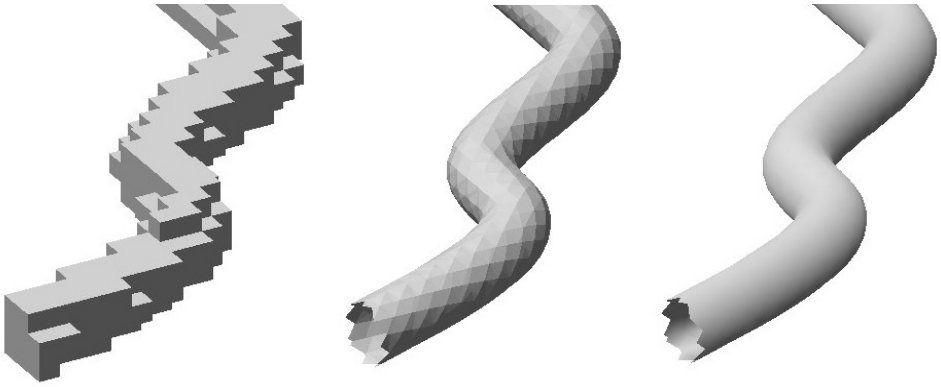


Рис. 9. Уточнение границы для компьютерной имитации кровеносного сосуда. Слева показана воксельная модель, полученная алгоритмом роста области, в центре и справа – триангуляция ее границы, построенная по уточняющей сети Chan – Purisima – Skala. В правом изображении применен алгоритм Фонга (Smooth Shading)

Fig. 9. Boundary refinement for a computer simulated blood vessel. The voxel model obtained by the region growing algorithm is shown on the left, the triangulation of its boundary built with the Chan – Purisima – Skala refining mesh is shown in the center and on the right. The Phong algorithm (Smooth Shading) is applied to the right image

На рис. 10 и 11 показан результат применения алгоритма к реальным данным. Слева изображена воксельная модель, в центре и справа – триангуляция, полученная уточняющим ал-

горитмом (используется плоское и гладкое закрашивание граней). Рисунок 11 крупно показывает небольшой фрагмент модели, представленной на рис. 10.

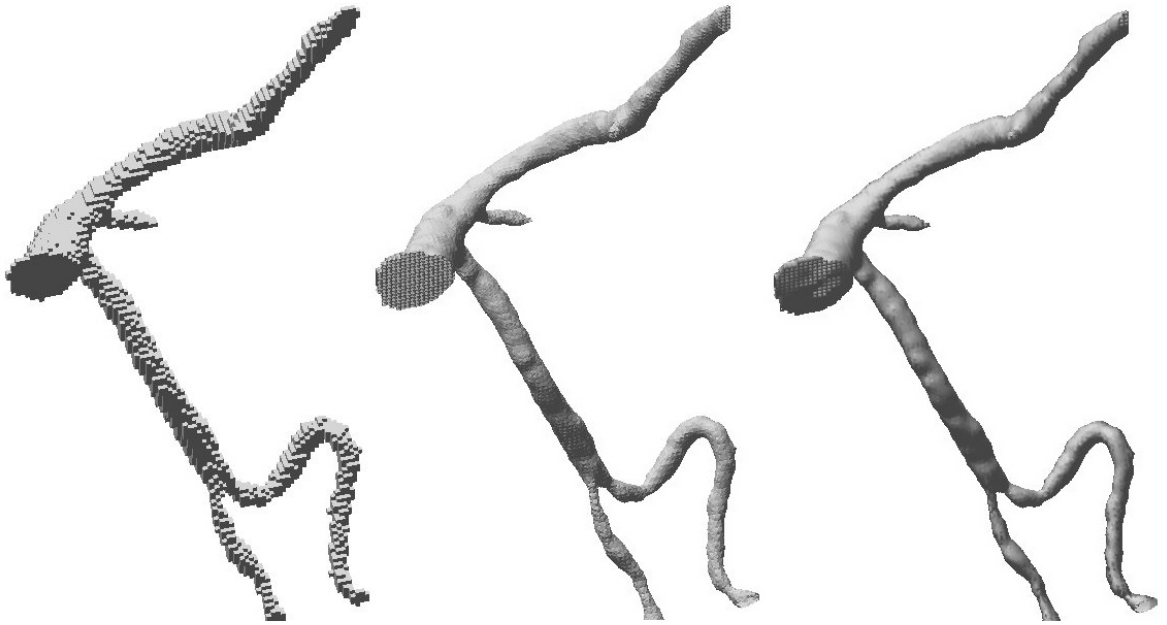


Рис. 10. Применение уточняющего алгоритма к реальным данным. Слева изображена воксельная модель, в центре и справа – триангуляция, полученная уточняющим алгоритмом (используется Flat Shading и Smooth Shading)

Fig. 10. Application of the refinement algorithm to real data. On the left is a voxel model, in the center and on the right is a triangulation obtained by a refinement algorithm (using Flat Shading and Smooth Shading)

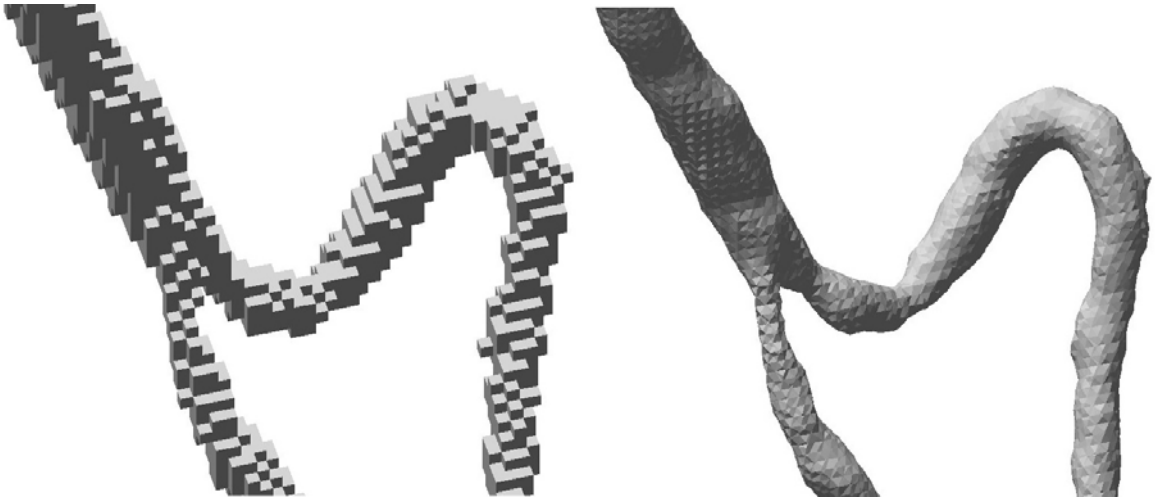


Рис. 11. Крупное изображение фрагмента модели, представленной на рис. 10

Fig. 11. A large image of a fragment of the model shown in Fig. 10

Все алгоритмы, описанные в данной работе, реализованы на языке C++ с использованием библиотеки классов Qt и системы разработки приложений QtCreator. Для получения изображений использовалась библиотека трехмерной графики OpenGL. Был реализован как трехмерный алгоритм роста области из семени, так и ячеечный алгоритм восстановления изопо-

верхности, использующий тетраэдрическую сеть Chan – Purisima – Skala. Последний алгоритм реализован в двух вариантах: классический вариант строит сеть во всем пространстве, в варианте уточнения границы сеть строится только в окрестности границы воксельной модели, как описано выше.

Отметим, что алгоритм уточнения границы воксельной модели работает чрезвычайно быстро, поскольку он проходит не по всему пространству, а только по границе модели, до этого вычисленной алгоритмом роста области. Тетраэдры сети имеют размеры, не превышающие размера вокселя, и строятся очень естественно по вокселям исходной модели. Это позволяет достичь необходимой гладкости и субпиксельной точности результирующей модели. Причем в модель не вносятся никакие искажения формы при сглаживании, неизбежные, к примеру, в методах Лупа или Кетмулла – Кларка [4; 5], поскольку все вычисления используют только исходную функцию плотности объекта в трехмерном пространстве, полученную как результат томографического обследования пациента. Таким образом, комбинация этих двух алгоритмов позволяет точно и эффективно воссоздать сглаженную трехмерную модель сосудов кровеносной системы, необходимую для программ гидродинамических расчетов.

Список литературы / References

1. **Аксенов А. А., Гудзовский А. В.** Пакет прикладных программ FlowVision // Тр. МФТИ. Серия: Аэрофизика и прикладная математика. М., 1998. С. 45–56.
   Aksenov A. A., Gudzovskiy A. V. Paket prikladnikh program FlowVision. In: Trudy MFTI. Ser. Aerofizika i prikladnaya matematika. Moscow, 1998, p. 45–56. (in Russ.)
2. **Аксенов А. А.** FlowVision: индустриальная вычислительная гидродинамика // Компьютерные исследования и моделирование. 2017. Т. 9, № 1. С. 5–20.
   Aksenov A. A. FlowVision: industrialnaya vichislitelnay gidrodinamika. *Komputernye issledovaniya i modelirovanie*, 2017, vol. 9, no. 1, p. 5–20. (in Russ.)
3. **Lorensen W. E., Cline H. E.** Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 1987, vol. 21, no. 4.
4. **Loop Ch.** Smooth Subdivision Surfaces Based on Triangles. M.S. Mathematics thesis. University of Utah, 1987.
5. **Catmull E., Clark J.** Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 1978, vol. 10, iss. 6, p. 350–355.
6. **Carneiro B. P., Kaufman A. E.** Tetra-Cubes: An algorithm to generae 3D isosurfaces based upon tetrahedra. In: SIGGRAPH, 1996, p. 205–210.
7. **Gueziec A.** Exploiting Triangulated Surface Extraction using Tetrahedral Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1995, vol. 1, iss. 4, p. 328–342.
8. **Chan S. L., Purisima E. O.** A new tetrahedral tesselation scheme for isosurface generation. *Computers & Graphics*, 1998, vol. 22, iss. 1, p. 83–90.
9. **Skala V.** Precision of Isosurface Extraction from Volume Data and Visualization. In: Conference on Scientific Computing, 2000, p. 368–378.

Материал поступил в редколлегию
Received
22.02.2019

Сведения об авторах / Information about the Authors

Борисенко Владимир Витальевич, кандидат физико-математических наук, ведущий научный сотрудник механико-математического факультета Московского государственного университета им. М. В. Ломоносова (Ленинские Горы, 1, Москва, 119991, ГСП-1, Россия)

Vladimir V. Borisenko, PhD (Math.), Leading Research Scientist, Lomonosov Moscow State University (1 Leninskie Gory, Moscow, 119991, Russian Federation)

vladimir_borisen@mail.ru

Серова Наталья Сергеевна, член-корреспондент РАН, доктор медицинских наук, профессор, директор Института электронного медицинского образования, Первый МГМУ им. И. М. Сеченова (Сеченовский Университет) (ул. Трубецкая, д. 8, стр. 2, Москва, 119991, Россия)

Nataliya S. Serova, Corresponding Member of the Russian Academy of Sciences, Dr. Sc. (Med.), Professor, Sechenov First Moscow State Medical University (Sechenov University) (8/2 Trubetskaya Str., Moscow, 119991, Russian Federation)

dr.serova@yandex.ru

Андрей Михайлович Чеповский, доктор технических наук, профессор, профессор НИУ Высшая Школа Экономики (ул. Мясницкая, 20, Москва,101000, Россия)

Andrey M. Chepovskiy, Dr. Sc. (Eng.), Professor, National Research University Higher School of Economics (20 Myasnitskaya Str., Moscow, 101000, Russian Federation)

achepovskiy@hse.ru

УДК 004.942
DOI 10.25205/1818-7900-2019-17-3-18-28

Локализация источников активности мозга на его трехмерной модели при помощи совместного анализа данных ЭЭГ/сМРТ

Е. П. Быкова [1], **А. Н. Савостьянов** [1–3]

[1] *Новосибирский государственный университет*
Новосибирск, Россия

[2] *Научно-исследовательский институт физиологии и фундаментальной медицины*
Новосибирск, Россия

[3] *Институт цитологии и генетики СО РАН*
Новосибирск, Россия

Аннотация

Несмотря на большое количество существующих методов диагностики головного мозга, он остается наименее изученной частью человеческого организма. Электроэнцефалография (ЭЭГ) – один из наиболее популярных методов исследования мозговой активности. Это обусловлено относительной дешевизной, безвредностью и мобильностью оборудования.

При анализе данных ЭЭГ возникает проблема решения обратной задачи электроэнцефалографии – локализации источников электрической активности мозга. Ее можно сформулировать следующим образом: по сигналам, регистрируемым на поверхности головы, необходимо определить, в какой области мозга расположены источники этих сигналов.

Целью исследования является разработка программной системы для локализации источников мозговой активности на основе совместного анализа данных ЭЭГ и сМРТ.

Существуют различные подходы к решению обратной задачи ЭЭГ. Для получения наиболее точных результатов некоторые из них предполагают использование данных сМРТ (изображений структурной магнитно-резонансной томографии), описывающих индивидуальную анатомию головы человека. В этой работе используется один из таких подходов – EMSICA (Electromagnetic Spatiotemporal Independent Component Analysis), предложенный A. Tsai.

В статье рассмотрены основные этапы работы системы, такие как предварительная обработка исходных данных; расчет специальной матрицы подхода EMSICA, значения которой показывают уровень активности определенного участка мозга; визуализация источников активности мозга на его трехмерной модели.

Ключевые слова

ЭЭГ, сМРТ, локализация источников мозговой активности, EMSICA, трехмерная модель мозга

Благодарности

Сбор данных ЭЭГ и МРТ выполнен при поддержке гранта РНФ № 17-18-01019. Разработка алгоритма анализа данных поддержана грантом РФФИ № 18-29-13027.

Для цитирования

Быкова Е. П., *Савостьянов А. Н.* Локализация источников активности мозга на его трехмерной модели при помощи совместного анализа данных ЭЭГ/сМРТ // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 18–28. DOI 10.25205/1818-7900-2019-17-3-18-28

The Localization of Activity Sources in a Three-Dimensional Model of the Human Brain Using the Joint Analysis of EEG / sMRI Data

E. Bykova [1], **A. Savostyanov** [1–3]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation

[2] *State Scientific-Research Institute of Physiology and Basic Medicine*
Novosibirsk, Russian Federation

[3] *Institute of Cytology and Genetics SB RAS*
Novosibirsk, Russian Federation

Abstract

Despite the large number of existing methods of the diagnosis of the brain, brain remains the least studied part of the human body. Electroencephalography (EEG) is one of the most popular methods of studying of brain activity due to its relative cheapness, harmless, and mobility of equipment.

While analyzing the EEG data of the brain, the problem of solving of the inverse problem of electroencephalography, the localization of the sources of electrical activity of the brain, arises. This problem can be formulated as follows: according to the signals recorded on the surface of the head, it is necessary to determine the location of sources of these signals in the brain.

The purpose of my research is to develop a software system for localization of brain activity sources based on the joint analysis of EEG and sMRI data.

There are various approaches to solving of the inverse problem of EEG. To obtain the most exact results, some of them involve the use of data on the individual anatomy of the human head – structural magnetic resonance imaging (sMRI data). In this paper, one of these approaches is supposed to be used – Electromagnetic Spatiotemporal Independent Component Analysis (EMSICA) proposed by A. Tsai.

The article describes the main stages of the system, such as preprocessing of the initial data; the calculation of the special matrix of the EMSICA approach, the values of which show the level of activity of a certain part of the brain; visualization of brain activity sources on its three-dimensional model.

Keywords

EEG, sMRI, localization of brain activity sources, EMSICA, three-dimensional model of the brain

Acknowledgements

The set of EEG and MRI data was carried out with the support of the grant of the Russian Science Foundation no. 17-18-01019. The development of the data analysis algorithm was supported by the RFBR grant no. 18-29-13027.

For citation

Bykova E. P., Savostyanov A. N. The Localization of Activity Sources in a Three-Dimensional Model of the Human Brain Using the Joint Analysis of EEG / sMRI Data. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 18–28. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-18-28

Введение

Мозг выполняет огромное количество жизненно важных функций человека, поэтому точная и своевременная диагностика состояния головного мозга – одна из ключевых задач современной медицины. Мозг человека активно исследуют посредством измерения и анализа его анатомии, биоэлектрической активности и т. п.

Один из наиболее популярных методов исследования мозговой активности – электроэнцефалография (ЭЭГ) – метод оценки функционального состояния головного мозга при помощи регистрации и анализа электрических сигналов на поверхности головы. Электроэнцефалография занимает важное место среди методов функциональной диагностики состояния мозга. Несмотря на появление таких диагностических методов, как магнитно-резонансная томография (МРТ), функциональная магнитно-резонансная томография, позитронно-эмиссионная томография (ПЭТ) и др., интерес врачей и исследователей к электроэнцефалографии и методам ее анализа в последнее время возрастает. Популярность ЭЭГ связана со сравнительно низкой стоимостью оборудования и его мобильностью, безвредностью ввиду отсутствия вредных для мозга излучений. Также стоит отметить, что ЭЭГ – чувствительный метод

исследования, он отражает малейшие изменения состояния коры головного мозга и глубинных мозговых структур, обеспечивая миллисекундное временное разрешение, недоступное другим методам исследования мозговой активности.

При анализе ЭЭГ-данных возникает проблема решения обратной задачи электроэнцефалографии – локализации источников электрической активности мозга. Ее можно сформулировать следующим образом: по ЭЭГ сигналам, регистрируемым на поверхности головы, необходимо определить, в какой области мозга расположены источники этих сигналов.

Целью данной работы является разработка программной системы для локализации источников мозговой активности на основе совместного анализа данных ЭЭГ и сМРТ.

Локализация источников ЭЭГ основана на широком спектре методов обработки сигналов, включающих цифровую фильтрацию, анализ трехмерных изображений, обработку массивов сигналов, моделирование и реконструкцию изображений, слепое разделение источников [1].

Наиболее распространенным подходом к решению обратной задачи ЭЭГ является определение пространственных характеристик и силы токовых диполей для компонент, вычисленных при помощи анализа независимых компонент (Independent Component Analysis – ICA). Данный алгоритм был разработан Aapo Hyvärinen и Erkki Oja [2]. Его основное преимущество – использование только ЭЭГ-данных для нахождения источников сигналов, кроме того данный метод включен в распространенный программный пакет EEGLAB. Но такой подход не учитывает индивидуальную анатомию человеческого мозга, что ведет к ошибкам в определении расположения источников активности.

Еще одним распространенным методом локализации источников мозговой активности является электромагнитная томография низкого разрешения (LORETA) [3].

Для получения наиболее точных результатов некоторые подходы предполагают использование данных МРТ, которые дают высокоточную информацию об индивидуальной анатомии головы человека. МРТ – способ получения томографических медицинских изображений для исследования внутренних органов и тканей с использованием явления ядерного магнитного резонанса. Изображения структурной МРТ (сМРТ) мозга предоставляют данные об анатомических структурах мозга с высоким пространственным разрешением.

Одним из подходов, использующих МРТ для решения обратной задачи ЭЭГ, является метод одновременной регистрации данных ЭЭГ и МРТ, описанный K. J. Mullinger и др. [4]. Данный способ точно локализует источники сигнала в головном мозге, но он крайне затратен по временным и денежным ресурсам.

В этой работе предполагается использование подхода, который также использует МРТ-данные – электромагнитный пространственно-временной анализ независимых компонент (Electromagnetic Spatiotemporal Independent Component Analysis – EMSICA), предложенного A. Tsai [5; 6]. Данный метод является модификацией анализа независимых компонент, в то же время он использует МРТ-данные. Его преимущество перед одновременной регистрацией данных ЭЭГ и МРТ – использование одних данных МРТ для разных ЭЭГ-экспериментов одного и того же обследуемого.

В статье подробно описана математическая модель электромагнитного пространственно-временного анализа независимых компонент, оригинальный алгоритм, а также предложенная модификация этого алгоритма. Также описан процесс предварительной обработки данных ЭЭГ и МРТ для использования алгоритмом. Представлена архитектура разработанной системы, этапы ее работы, пользовательский интерфейс, а также некоторые детали реализации. Особое внимание уделено подходу к визуализации источников мозговой активности.

Подход EMSICA

Электромагнитный пространственно-временной анализ независимых компонент, предложенный A. Tsai [5], основан на совместном анализе данных ЭЭГ и МРТ. Этот метод является модификацией ICA, и его основное преимущество – использование информации об индивидуальной анатомии головы обследуемого. Стоит отметить, что данные сМРТ регистрируются

один раз независимо от регистрации ЭЭГ-данных. Иначе говоря, можно получать источники мозговой активности для каждого нового ЭЭГ-обследования без повторного МРТ-обследования.

Математическая модель EMSICA

В данном подходе расположение источников мозговой активности будет определяться на поверхности коры головного мозга. Для этого предполагается разделить поверхность головного мозга на J тесселяционных элементов (здесь под тесселяционным элементом подразумевается небольшой участок на поверхности коры головного мозга).

Математическая модель данного метода [5] выглядит следующим образом:

$$x^{(t)} = Au^{(t)} = LBs^{(t)} \quad (1)$$

$x^{(t)} \in R^M$ – известная векторная функция от времени, содержащая данные ЭЭГ-сигналов, в реальном наборе данных представляется как $x \in R^{M \times T}$ $t \in [1, 2, \dots, T]$.

$s^{(t)} \in R^K$ – неизвестная векторная функция от t (времени), описывающая искомые разделенные сигналы, в реальном наборе данных представляется как $s \in R^{K \times T}$ $t \in [1, 2, \dots, T]$.

$L \in R^{M \times J}$ – известная матрица потенциальных полей, содержащая информацию о геометрии и проводимости модели.

$B \in R^{J \times K}$ – неизвестная матрица весов с элементами b_{jk}, которые задают уровень активности k-го источника в j-м тесселяционном элементе на коре. Таким образом, по значениям столбца b_k $(k = 1, \dots, K)$ можно определить активные участки коры мозга, соответствующие k-й компоненте.

Задача заключается в поиске матрицы B по известным $x(t)$ и L для последующей визуализации активных участков мозга в виде K компонент EMSICA.

Предполагается, что $M = K$, чтобы гарантировать существование матрицы $(LB)^{-1}$.

Также в [5] утверждается, что матрица A в (1) может быть интерпретирована аналогично матрице смешивания подхода анализа независимых компонент.

В байесовском подходе [1] обратную задачу можно сформулировать как оценку распределения источников на коре, а также соответствующих им сигналов по данным, записанным с M электродов. Расположение источников, т. е. B, можно оценить, максимизируя следующую апостериорную вероятность [11]:

$$p\left(B, s^{(t)} \middle| x^{(t)}, L\right) \propto p\left(x^{(t)} \middle| L, B, s^{(t)}\right) p(B, s^{(t)}|L). \quad (2)$$

Здесь $\propto$ обозначает пропорциональность, т. е. $y \propto x$ то же самое, что $y = kx$. Если предположить независимость расположения источников на поверхности коры и независимость сигналов во времени, справедливо $p(B, s^{(t)}|L) \propto p(B)p(s^{(t)})$. Тогда

$$p\left(B, s^{(t)} \middle| x^{(t)}, L\right) \propto p\left(x^{(t)} \middle| L, B, s^{(t)}\right) p(B) p(s^{(t)}).$$

Чтобы избавиться от неудобного параметра $s^{(t)}$, по правилам байесовского подхода уравнение (2) можно преобразовать к

$$p(B|x^{(t)}, L) \propto p(B) \int ds^{(t)}\, p\left(x^{(t)} \middle| L, B, s^{(t)}\right) p(s^{(t)}). \quad (3)$$

Поскольку исходные компоненты считаются независимо распределенными, априорную вероятность $p(s^{(t)})$ можно представить в виде произведения априорных вероятностей отдельных источников, т. е. $p\left(s^{(t)}\right) = \prod_k p_k\left(s^{(t)}\right)$.

Тогда на основе подхода алгоритма Infomax ICA [7], также использующего байесовский подход, интеграл в правой части уравнения (3) можно преобразовать следующим образом:

$$\int ds^{(t)}\, p\left(x^{(t)} \middle| L, B, s^{(t)}\right) p\left(s^{(t)}\right) = \frac{1}{\det(LB)} \prod_k p_k\left(s_k{}^{(t)}\right).$$

Элементы k-го столбца матрицы B могут рассматриваться как случайный вектор b_k, и предполагается, что эти векторы имеют независимое распределение в пространстве, тогда их распределение $p(B) = \prod_k p_k(b_k)$.

На основе изложенного апостериорная вероятность $p(B|x^{(t)}, L)$ имеет следующий вид:

$$p(B|x^{(t)}, L) \propto \prod_k p_k(b_k) \frac{1}{\det(LB)} \prod_k p_k\left(s_k{}^{(t)}\right). \quad (4)$$

В работе A. Tsai [6] выдвинуто предположение, что $p_k(b_k)$ и $p_k\left(s_k{}^{(t)}\right)$ имеют следующий вид:

$$p_k(b_k) \propto \exp\{-\beta f(b_k)\} \operatorname{sech}^2(b_k), \quad (5)$$

$$p_k\left(s_k{}^{(t)}\right) \propto \exp\left\{-\left(s_k{}^{(t)}\right)^2\right\} \operatorname{sech}^2\left(s_k{}^{(t)}\right). \quad (6)$$

На основе представленных в статье [6] формул приведем подробное описание используемых величин функций:

$f: R^K \to R$ и $f(b_k) = b_k C^{-1} b_k{}^T, C^{-1} \in R^{J \times J}$ – диагональная матрица, каждое значение которой имеет вид $r_{jj} = \sqrt{\sum_m^M L_{mj}}$;

$\exp\{-\beta f(b_k)\}: R \to R$;

sech – гиперболический секанс, т. е. $\operatorname{sech}^2(b_k)$ – скалярный квадрат вектора $sech(b_k)$;

$p_k(b_k)$: $R^K \to R^K$;

$p_k\left(s_k{}^{(t)}\right)$: $R \to R$.

Скаляр $\beta \in R$ является гиперпараметром.

Из-за того, что временной интервал ЭЭГ-обследования (T временных единиц) может быть достаточно большим, в подходе EMSICA предлагается разбить его на интервалы по τ единиц времени. Другими словами вектор-функция $x^{(t)}$ в реальном наборе данных представляет собой матрицу $x \in R^{M \times T}$ $t \in [1, 2, \dots, T]$, тогда разобьем ее на $\frac{T}{\tau}$ блоков $x \in R^{M \times \tau}$.

Учитывая это, в работе A. Tsai было установлено, что апостериорная логарифмическая вероятность, соответствующая апостериорной вероятности в (4), имеет следующий вид:

$$l = \tau \sum\nolimits_{j,k} \log\left(p_k\left(b_{jk}\right)\right) - \tau \log(\det(LB)) + \sum\nolimits_{k,t} \log\left(p_k\left(s_k{}^{(t)}\right)\right).$$

На основе вышеизложенного для нахождения матрицы B необходимо решить следующую задачу безусловной оптимизации:

$$l = \sum_{k=1}^{K} \left\{ \tau \cdot log\left(\sum_{j=1}^{J} \operatorname{sech}^2\left(b_{jk}\right)\right) + \sum_{t=1}^{\tau} log(\operatorname{sech}^2(s_{tk})) - log\left(\sum_{t=1}^{\tau} (s_{tk})^2\right) - \beta \cdot \tau \sum_{j=1}^{J} \left(b_{jk} r_j\right) \right\} +$$

$$+\tau \cdot \log(\det(LB)),$$

$$s_{tk} = (LB)^{-1} x_{tk}, \quad (7)$$

где $k \le K, t \le \tau,\ l \to max_B$.

Максимизация логарифмической апостериорной вероятности

A. Tsai предлагает найти максимум логарифмической вероятности l с помощью метода градиентного спуска. Для этого берется ее градиент по B (здесь подразумевается взятие частной производной по каждому элементу матрицы B):

$$\frac{\partial l}{\partial B} = \phi(B) - L^T (LB)^{-T} - \frac{1}{\tau} L^T (LB)^{-T} \varphi(S) S^T,$$

где $S = [s^{(1)}, \dots, s^{(\tau)}]$,

$\phi(B) \in R^{J\times K}$, $\varphi(S) \in R^{K\times\tau}$ – матрицы со значениями $\phi(b_{jk}) = \partial\log(p_k(b_{jk}))/\partial b_{jk}$ и $\varphi(s_k{}^{(t)}) = \partial\log(p_k(s_k{}^{(t)}))/\partial s_k{}^{(t)}$ соответственно.

На основании (5) и (6) нами были выведены следующие формулы:

$$\phi(b_{jk}) = -\left(2\tanh(b_{jk}) + \beta\sum_{j'=1}^{J} b_{j'k}r_{jj'}\right),$$

где $r_{jj'} - jj'$ – элемент матрицы C^{-1};

$$\varphi(s_k{}^{(t)}) = -2\,(\tanh(s_k{}^{(t)}) + s_k{}^{(t)}).$$

Умножим правую часть уравнения (7) на BB^T [5]:

$$\Delta B = (BB^T)^{-1}\cdot BB^T\frac{\partial l}{\partial B} = (BB^T)^{-1}\cdot B\left[B^T\phi(B) - I + \frac{1}{\tau}B^T\varphi(S)S^T\right].$$

Тогда на каждой итерации градиентного спуска

$$B_{i+1} = B_i + \alpha(BB^T)^{-1}\Delta_i B, \qquad (8)$$

$$s_{i+1}^{(t)} = (LB_i)^{-1}x^{(t)}.$$

В качестве правила окончания цикла было выбрано правило

$$\max(\Delta b_{jk}) < \varepsilon.$$

Здесь возникает проблема поиска параметра α в уравнении (8). Принятие параметра α за константное значение и его эмпирический подбор не показали хорошего результата. Поэтому поиск параметра α был произведен на основе модификации алгоритма Нелдера – Мида [8].

В результате эмпирических исследований нами были выбраны следующие значения гиперпараметров алгоритма $\beta = 100000$, $\tau = 100$. Такие значения обеспечили наименьшее количество общего числа итераций оптимизационного алгоритма для предоставленного набора тестовых данных.

Предварительная обработка данных

Для исследования Научно-исследовательским институтом физиологии и фундаментальной медицины были предоставлены очищенные от шума ЭЭГ-данные в формате EEGLAB с расширением .set и данные структурной МРТ с расширением .nii.

Алгоритм EMSICA в качестве входных данных принимает матрицу потенциальных полей $L \in R^{M\times J}$, полученную на основе обработки данных МРТ и ЭЭГ, а также вектор-функцию $x^{(t)} \in R^M$ $(t \in [1, 2, \dots, \tau])$ или матрицу $x \in R^{M\times\tau}$ (в дискретном виде), извлеченную из ЭЭГ-данных. Исходя из предоставленных данных размерности $M = K = 126$, $\tau = 3500$. Количество тесселяционных элементов в этом исследовании $J = 15000$.

Программная система включает в себя модуль по обработке данных. После того, как пользователь загрузит файл с ЭЭГ-данными (с расширением .set), а также файл с матрицей потенциальных полей (с расширением .mat), запускается реализованный на языке MATLAB скрипт для извлечения необходимых данных.

Далее описывается способ получения файла с матрицей потенциальных полей, примененный в результате подготовки данных для нашей программной системы.

Вычисление матрицы потенциальных полей

Решение обратной задачи ЭЭГ с использованием данных об анатомии мозга предполагает предварительное моделирование тканей головы и характеристик электродов.

Матрица потенциальных полей (матрица усиления) L, используемая в (1), содержит информацию о том, как электрический ток, протекающий в мозге, создает различия в электрических потенциалах на внешних датчиках ЭЭГ (электродах), учитывая различные ткани головы.

На основании подхода EMSICA для каждого обследуемого матрица L рассчитывается на основе высококачественной реалистичной модели головы методом граничного элемента (BEM – boundary element method) [9].

В настоящее время существует несколько программных продуктов, выполняющих подобные вычисления. В этой работе для вычисления матрицы усиления было решено использовать программный пакет Brainstorm, который задокументирован и предназначен для скачивания онлайн под общедоступной лицензией GNU [10]. Для расчета матрицы L этому программному пакету необходимы данные модели головы, которые могут быть получены с помощью разных средств. Мы решили остановиться на программе BrainSuite. Выбор данных продуктов был обусловлен прежде всего удобством и простотой в использовании, а также наличием подробного руководства.

Итак, сначала необходимо построить реалистичную модель головы в программном пакете BrainSuite. BrainSuite создает индивидуальные модели структур мозга на основе T1-взвешенной МРТ головы человека, в том числе модели внутренних и внешних границ коры головного мозга. Данные этой модели также будут использованы для визуализации активных участков мозга на его трехмерной модели.

Предоставленные данные МРТ имеют высокое пространственное разрешение, и при построении модели головы в программе BrainSuite возникает сбой из-за превышения максимального размера стека памяти. Для решения этой проблемы был реализован скрипт OVERRIDE_MRI для сжатия МРТ-данных на языке MATLAB на основе метода *reslice_nii()* библиотеки Tools for NIfTI and ANALYZE image.

После использования скрипта для сжатия нужно загрузить полученный файл в приложение BrainSuite и выбрать опцию для выполнения всех шагов извлечения поверхности коры головного мозга. В результате выполнения программы будут созданы файлы с моделями анатомических структур мозга.

Теперь на основе полученных данных можно вычислить L матрицу с помощью программного пакета Brainstorm. Сначала нужно импортировать данные анатомии мозга и файл с ЭЭГ-данными.

В программном обеспечении Brainstorm по умолчанию считается, что электрическая активность, которая регистрируется датчиками, создается в основном набором электрических диполей, расположенных на поверхности коры [9]. Используемая сетка источников (диполей) определяется поверхностью коры, каждая вершина этой поверхности рассматривается как диполь. Анализ данных в этой работе был проведен на основе 15 000 вершин (значение, выставленное в Brainstorm по умолчанию). Использование меньшего количества вершин просто снизит разрешение результатов; использование большего количества приводит к чрезмерному увеличению объема данных и может привести к проблемам с памятью.

Для вычисления L матрицы Brainstorm предлагает несколько методов. Основываясь на работах [5; 6], нужно использовать OpenMEEG BEM – метод граничного элемента из программного обеспечения с открытым исходным кодом OpenMEEG [11]. В результате создается файл с расширением .mat, содержащий L матрицу, а также координаты вершин сетки. Для извлечения этих данных из полученного файла был реализован скрипт на языке MATLAB, который включается в модуль по обработке данных нашей программной системы.

Подход к визуализации компонент EMSICA

Кортикальная карта, или изображение коры головного мозга k-й компоненты EMSICA, может быть описана на основе значений соответствующего столбца матрицы B, т. е. b_k, содержащего одно значение для каждого тесселяционного элемента. Чтобы найти и отобразить

вершины, сильно активные в определенной карте компонент EMSICA, столбцы матрицы B нормализуются по стандартному отклонению. Нормализованные значения матрицы B обозначим z_{jk}. Участки мозга, для которых соответствующие абсолютные значения z_{jk} превышают некоторый порог, считаются сильно активными [5] и выделяются цветом на трехмерной модели головного мозга. Именно таким образом отображаются компоненты EMSICA в нашей системе.

Здесь стоит отметить, что интерпретировать физический смысл ненормализованных значений довольно сложно, так как единицей измерения значений матрицы B по СИ является $1\ \frac{А \cdot м}{мкВ}$. Эта величина выведена из уравнения (1) на основе единиц измерения матрицы потенциальных полей – $1\ \frac{мкВ}{А \cdot м}$ и матрицы x ЭЭГ-данных – 1 *мкВ* (именно такие единицы измерения имели извлеченные из файлов данные). Нормализованное значение отображает степень отклонения состояния соответствующего участка коры.

При реализации модуля для визуализации компонент EMSICA возникла проблема совмещения сетки мозга, полученной в программном пакете Brainstorm, с реалистичной моделью мозга BrainSuite. Как уже отмечалось, наша программная система использует эти промежуточные данные для визуализации трехмерной модели головного мозга.

На основе подхода EMSICA нам известен уровень активности вершин модели Brainstorm. Но так как мы визуализируем компоненты EMSICA на модели мозга BrainSuite, для каждой вершины этой модели необходимо найти ближайшую вершину модели Brainstorm.

Для решения возникшей проблемы был разработан следующий подход. Сначала осуществляется поиск центров моделей мозга. Мы предполагаем, что центр – это точка, имеющая минимальное среднее расстояние r до вершин сетки. Формально это можно представить следующим образом:

$v_j \in R^3$ – вектор координат j-й вершины сетки модели мозга ($j \in [1, J]$);

$c \in R^3$ – искомый центр сетки модели мозга.

Тогда поиск центра модели будет состоять в решении задачи оптимизации:

$$r = \frac{\sum \sqrt{(v_j - c)^2}}{J} \to \min_c.$$

В качестве начального приближения центра модели мы выбрали центр масс $c_0 = \frac{\sum v_j}{J} \in R^3$.

Будем считать что

${v_j}^1$ – вершины модели Brainstorm ($j \in [1, J]$);

${v_i}^2$ – вершины модели BrainSuite ($i \in [1, I]$);

c^1 – центр модели Brainstorm, c^2 – центр модели BrainSuite, полученные на основе приведенного выше алгоритма.

Тогда если вершина ${v_{j_{k1}}}^1$ – активная вершина, то все вершины ${v_{j_{k2}}}^2$ такие, что угол между векторами ${v_{j_{k1}}}^1 - c^1$ и ${v_{j_{k2}}}^2 - c^2$ меньше γ, будем считать активными.

Таким образом мы совмещаем центры моделей друг с другом. Стоит отметить, что приведение базисов не требуется, так как системы координат обеих моделей уже имеют один и тот же базис. Поэтому мы можем интерпретировать векторы ${v_{j_{k1}}}^1 - c^1$ и ${v_{j_{k2}}}^2 - c^2$ как радиус-векторы вершин соответствующих моделей в одной системе координат, при этом центры моделей будут совпадать с началом координат.

Теперь поясним роль угла γ в нашем подходе. Предположим, что мы спроецировали вершины обеих моделей на сферу с центром в начале координат. Тогда будем считать, что для некоторой активной вершины v_j модели Brainstorm, все вершины, проекции которых находятся в малой окрестности проекции вершины v_j, являются активными. Очевидно, что при такой постановке, явное проецирование не требуется, так как такая окрестность может быть задана с помощью угла γ – угла между радиус-векторами вершин обеих моделей.

Описание программной системы

Разработанная система для локализации источников мозговой активности имеет три основных модуля.

1. Модуль предварительной обработки данных.
2. Модуль для вычисления компонент EMSICA. Данный модуль находит решение оптимизационной задачи (7).
3. Модуль для визуализации источников мозговой активности.

Модуль для обработки данных был реализован на языке MATLAB. Выбор языка определен прежде всего тем, что файл с матрицей потенциальных полей имеет формат данных программной системы MATLAB, поэтому здесь имеется богатый функционал для обработки подобных данных.

Второй модуль также реализован на языке MATLAB и реализует алгоритм максимизации апостериорной логарифмической вероятности. После вычисления матрицы B он осуществляет экспорт этой матрицы и набор координат модели Brainstorm для использования модулем визуализации. Модуль для визуализации компонент EMSICA разработан с помощью фреймворка Qt на языке C++ и предоставляет пользовательский интерфейс для работы со всей системой.

Основной опцией нашей программной системы является визуализация кортикальных карт компонент EMSICA с заданным порогом, который характеризует уровень активности участков коры мозга.

Также для осуществления дополнительного анализа компонент на основе программного комплекса EEGLAB предоставляется опция для создания и сохранения файла с расширением .set, который содержит данные о компонентах EMSICA, подобно данным о компонентах ICA (матрицу смешивания A), поэтому все механизмы анализа независимых компонент применимы и к новым компонентам.

Результаты

Программная система опробована на данных 10 человек, и, по экспертной оценке сотрудников НИИФФМ, было подтверждено, что полученные компоненты действительно локализуют наиболее активные участки мозга, а также имеют тенденцию к разделению коры мозга на стандартные анатомические области. Кроме того, отображение новых компонент с помощью программного пакета EEGLAB соответствует изображениям на трехмерной модели мозга для всех тестовых данных, что подтверждает выбранный для визуализации подход. На рис. 1 представлена визуализация компоненты EMSICA на трехмерной модели мозга нашей программной системы (слева), а также двумерное изображение этой же компоненты, полученное в программном пакете EEGLAB (справа).

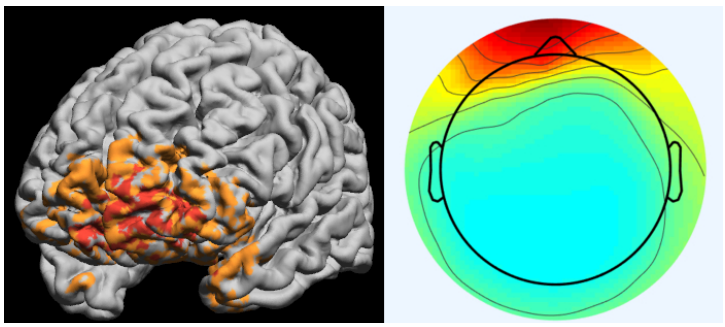


Рис. 1. Визуализация компоненты EMSICA

Fig. 1. Visualization of EMSICA Component

На рис. 2 изображена информация о частотной характеристике сигнала источника, соответствующей этой компоненте. Данные также получены в программном пакете EEGLAB. По частотному анализу видно, что найденный источник действительно проявляет активность в момент начала ЭЭГ-эксперимента (выраженный пик в момент времени 0).

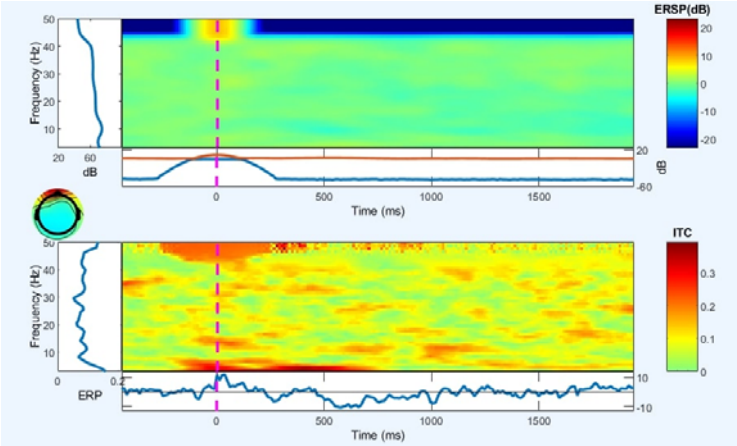


Рис. 2. Частотный анализ компоненты EMSICA
Fig. 2. Frequency Analysis of EMSICA Component

Заключение

Подход EMSICA для решения обратной задачи ЭЭГ объединяет сильные стороны существующих подходов, поэтому нами был выбран этот подход для разработки программной системы локализации источников мозговой активности.

Во время исследования данного подхода были внесены некоторые уточнения и модификации, необходимые с точки зрения реализации. Также был разработан подход к визуализации компонент EMSICA на трехмерной модели головного мозга, полученной с помощью приложения BrainSuite.

Результатом работы является программная система, которая позволяет визуализировать кортикальные карты компонент EMSICA, а также активные участки мозга в определенный момент времени. Кроме того, для получения большей информации о компонентах предусмотрена опция сохранения данных в файл с расширением .set, в результате чего может быть проведен дополнительный анализ полученных результатов в программном пакете EEGLAB.

На данный момент разработанная система проходит апробацию в Научно-исследовательском институте физиологии и фундаментальной медицины.

Список литературы / References

1. **Baillet S., Mosher J. C., Leahy R. M.** Electromagnetic Brain Mapping. *IEEE Signal Processing Magazine*, 2001, vol. 18, no. 6, p. 14–30. DOI 10.1109/79.962275
2. **Hyvärinen A., Oja E.** Independent Component Analysis: Algorithms and Applications. *Neural Networks*, 2000, vol. 13, no. 4–5, p. 411–430. DOI 10.1016/S0893-6080(00)00026-5
3. **Pascual-Marqui R. D., Michel C.M., Lehmann D.** Low resolution electromagnetic tomography: a new method for localizing electrical activity of the brain. *Int. J. Psychophysiol*, 1994, vol. 18, no. 1, p. 49–65. DOI 10.1016/0167-8760(84)90014-x
4. **Mullinger K. J., Castellone P., Bowtell R.** Best Current Practice for Obtaining High Quality EEG Data During Simultaneous fMRI. *Journal of Visualized Experiment*, 2013, vol. 76. DOI 10.3791/50283

5. **Tsai A. C.** Mapping single-trial EEG records on the cortical surface through a spatiotemporal modality. *NeuroImage*, 2006, vol. 32, no. 1, p. 195–207. DOI 10.1016/j.neuroimage.2006.02.044
6. **Tsai A. C., Jung T. P., Chien V S. C., Savostyanov A. N., Makeig S.** Cortical surface alignment in multi-subject spatiotemporal independent EEG source imaging. *NeuroImage*, 2014, vol. 87, p. 297–310. DOI 10.1016/j.neuroimage.2013.09.045
7. **Pearlmutter B. A., Parra L. C.** A context-sensitive generalization of ICA. In: International Conference on Neural Information Processing, Hong Kong, 1996.
8. **Lagarias J. C., Reeds J. A., Wright M. H., Wright P. E.** Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal of Optimization*, 1998, vol. 9, no. 1, p. 112–147. DOI 10.1137/s1052623496303470
9. **Munck J. C. de.** A linear discretization of the volume conductor boundary integral equation using analytically integrated elements. *IEEE Transactions on Biomedical Engineering*, 1992, vol. 39, no. 9, p. 986–990. DOI 10.1109/10.256433
10. **Tadel F., Baillet S., Mosher J. C., Pantazis D., Leahy R. M.** Brainstorm: A User-Friendly Application for MEG/EEG Analysis. *Computational Intelligence and Neuroscience*, 2011, p. 1–13. DOI 10.1155/2011/879716
11. **Gramfort A., Papadopoulo T., Olivi E., Clerc M.** OpenMEEG: opensource software for quasistatic bioelectromagnetics. *BioMedical Engineering OnLine*, 2010, vol. 9, no. 1, p. 45. DOI 10.1186/1475-925x-9-45

Материал поступил в редколлегию
Received
16.05.2019

Сведения об авторах / Information about the Authors

Быкова Екатерина Павловна, магистрант 2 курса факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Ekaterina P. Bykova, Master's Student, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

e.bykova1@g.nsu.ru

Савостьянов Александр Николаевич, доктор философских наук, профессор, заместитель кафедры общей информатики факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия), ведущий научный сотрудник Научно-исследовательского института физиологии и фундаментальной медицины (ул. Тимакова, 4, Новосибирск, 630117, Россия), Институт цитологии и генетики СО РАН (пр. Академика Лаврентьева, 10, Новосибирск, 630090, Россия)

Alexander N. Savostyanov, Doctor of Philosophy, Professor, Deputy Chair of General Informatics, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation), Senior Researcher, Research Institute of Physiology and Fundamental Medicine (4 Timakov Str., Novosibirsk, 630117, Russian Federation), Institute of Cytology and Genetics SBRA of Sciences (10 Academician Lavrentiev Ave., Novosibirsk, 630090, Russian Federation)

alexander.savostyanov@gmail.com

УДК 004.825
DOI 10.25205/1818-7900-2019-17-3-29-42

Разработка методов интеграции автоматических средств логического вывода для порождения знаний в онтологической модели

А. И. Капустина [1], **Д. Е. Пальчунов** [1, 2]

[1] *Новосибирский государственный университет*
Новосибирск, Россия
[2] *Институт математики им. С. Л. Соболева СО РАН*
Новосибирск, Россия

Аннотация

Статья посвящена разработке методов порождения новых знаний на основе анализа текстов естественного языка. Для извлечения знаний из текстов на естественном языке используется метод представления предложений в виде двухместных предикатов с введенной константой ситуацией. Для представления знаний в формальном виде используются бесквантворные предложения логики предикатов, а также язык OWL DL. Порождение новых знаний реализуется при помощи автоматических средств логического вывода с использованием заранее заданных шаблонов правил вывода. Разработана программная система, которая дает возможность пользователям получать ответы на определенные вопросы, относящиеся к данным текстам естественного языка. Ответы строятся на естественном языке, при этом используются не только явно содержащиеся в обрабатываемом документе знания, но и знания, порожденные при помощи автоматических средств логического вывода.

Ключевые слова

извлечение знаний, порождение знаний, фрагменты атомарных диаграмм, OWL DL, SWRL, SQWRL, автоматические средства логического вывода, онтология, онтологическая модель

Благодарности

Исследование выполнено при частичной финансовой поддержке Президиума СО РАН (проект «Инженерия интенсиональных онтологий в дедуктивных и информационных системах» Комплексной программы ФНИ СО РАН II.1).

Для цитирования

Капустина А. И., *Пальчунов Д. Е.* Разработка методов интеграции автоматических средств логического вывода для порождения знаний в онтологической модели // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 29–42. DOI 10.25205/1818-7900-2019-17-3-29-42

Development of Methods for Integrating Automatic Logical Inference Tools to Generate Knowledge in the Ontological Model

A. I. Kapustina [1], **D. E. Palchunov** [1, 2]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation
[2] *Sobolev Institute of Mathematics SB RAS*
Novosibirsk, Russian Federation

Abstract

The article is devoted to the development of new knowledge generation methods based on the analysis of natural language texts. To extract knowledge from natural language texts, the method of presenting sentences in the form of binary predicates with new constant-situation is used. For the representation of knowledge in a formal form, we use quantifier-free sentences of predicate logic, as well as the OWL DL language. The generation of new knowledge is re-

alized with the help of reasoners, using pre-defined patterns of inference rules. A software system has been developed that allows users to get answers to specific questions related to given natural language texts. The answers are built in a natural language, using not only the knowledge that is explicitly contained in the document being processed, but also the knowledge generated by the reasoners.

Keywords

knowledge extraction, knowledge generation, fragments of atomic diagrams, OWL DL, SWRL, SQWRL, reasoners, ontology, ontological model

Acknowledgements

The study was carried out with partial financial support from the Presidium of the SB RAS (the project "Engineering of Intensional Ontologies in Deductive and Information Systems" of the Comprehensive Program of the Federal Research Institute of SB RAS II.1).

For citation

Kapustina A. I., Palchunov D. E. Development of Methods for Integrating Automatic Logical Inference Tools to Generate Knowledge in the Ontological Model. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 29–42. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-29-42

Введение

Данная работа посвящена разработке и программной реализации методов порождения новых знаний по текстам естественного языка. Порождение новых знаний происходит при помощи автоматических средств логического вывода с использованием шаблонов, предварительно созданных для заданной предметной области. Знания, извлеченные из текстов, представляются в виде фрагмента атомарной диаграммы модели в сигнатуре, состоящей из символов двухместных предикатов и констант. Далее атомарные предложения транслируются в язык OWL DL, благодаря чему мы можем использовать автоматические средства логического вывода (ризонеры). Корректность порожденных знаний контролируется пользователями с помощью запросов к программной системе на естественном языке.

В своей жизни мы сталкиваемся с большим количеством разных документов. Очень часто документы имеют большие объемы, и при этом информация в документах может быть не структурирована. Ключевые моменты могут находиться в разных местах документа: одна часть важной информации может быть в начале документа, а другая часть – в конце. Для точного определения содержания документа нужно выяснять смысл каждого предложения в отдельности, а также семантическое влияние одних предложений на другие.

Сходные проблемы возникают, когда для решения какой-либо задачи нужно извлечь знания из нескольких связанных между собой документов и затем сопоставить знания, содержащиеся в разных документах.

Так, например, для определения порядка действий студента ФИТ в течение ряда лет его обучения, нужно изучить следующие регламентирующие документы: «Образовательная программа по направлению 09.04.01 информатика и вычислительная техника», «09.04.01 Программа государственной итоговой аттестации по образовательной программе высшего образования – программе магистратуры», «Приложения к программе государственной итоговой аттестации по образовательной программе высшего образования – программе магистратуры», «Приказы о проведении сессий» и другие связанные документы. В одних документах описывается порядок действий студентов, в других описаны сроки и т. п. Поэтому, чтобы извлечь правильную информацию из таких текстов естественного языка, нужно изучить все связанные между собой документы. Учитывая количество и объемы таких документов, сделать это может быть крайне затруднительным.

Целью разработки программной системы, описываемой в данной статье, является автоматизация извлечения знаний из текстовых документов с возможностью порождения новых знаний на естественном языке за счет сопоставления и комбинации извлеченных знаний и осуществления логического вывода.

Программная система должна решать следующие задачи:

- извлекать знания из текстовых документов на естественном языке и представлять их в формальном виде (на языке логики предикатов и OWL DL);
- осуществлять логический вывод с использованием ризонеров для OWL DL и таким образом порождать новые знания;
- предоставлять пользователям возможность задавать вопросы на естественном языке, относящиеся к рассматриваемым текстам документов.

В результате применения разработанных методов была создана программная система, предназначенная для порождения новых знаний по текстам на естественном языке, основанная на представлении извлеченных из текстов знаний в виде онтологической модели. Созданная программная система позволяет получать знания, которые явно не содержатся в документах, но логически следуют из них.

Модульная система для реализации онтологической модели

При создании онтологических моделей различных предметных областей мы основываемся на четырехуровневой модели представления знаний [1; 2]. В рамках этого подхода разработка онтологической модели предметной области включает в себя:

- создание онтологии предметной области, т. е. задание набора ключевых понятий – сигнатуры онтологической модели и спецификация смысла этих ключевых понятий, задание взаимосвязей, отношений между понятиями и т. д.;
- формальное представление общих знаний (универсальных утверждений и регламентов, истинных для каждого прецедента данной предметной области);
- формальное представление набора прецедентов предметной области;
- формальное представление вероятностных, оценочных и экспертных знаний в виде предложений, имеющих нечеткие значения истинности [1; 2].

При этом происходит настройка взаимосвязи со сторонними программными системами, описание взаимодействия с другими онтологиями и др. [3–5]. В рамках нашего подхода создание онтологической модели можно представить как разработку модульной системы [6]. Модулями этой системы, в частности, являются: ядро, модуль пополнения онтологической модели, модуль взаимодействия с онтологической моделью.

Ядро модульной системы

Ядро системы позволяет создавать онтологии на языке OWL DL [1]. Создавать онтологию можно как посредством доступных редакторов онтологий с графическим интерфейсом, например Protégé, так и напрямую на языке OWL DL. Для ядра системы построения используется описанная выше четырехуровневая модель представления знаний [1; 2]: 1) онтология, 2) общие знания, 3) прецеденты, 4) вероятностные и оценочные знания.

Ядро системы, в частности, позволяет проверять созданные онтологические модели на непротиворечивость при помощи автоматических средств логического вывода.

Модуль пополнения онтологической модели

Модуль пополнения онтологической модели позволяет пополнять онтологическую модель новыми знаниями – как извлеченными из текстов естественного языка, так и полученными в результате логического вывода. Пополнение онтологической модели можно сделать несколькими способами.

Первый способ – это объединение онтологий. Для создания онтологических моделей существует несколько языков, основными из которых являются RDF, RDF Schema, OWL, логика описаний. Данный модуль позволяет объединять онтологии, написанные на разных языках. Для этого существуют программы для перевода с одного языка на другой.

[1] OWL 2 Web Ontology Language Document Overview (Second Edition) [Electronic resource]. W3C OWL Working Group. URL: http://www.w3.org/TR/owl2-overview

Второй способ – это использование правил вывода. Правила вывода могут быть написаны на разных языках. Для получения знаний с помощью правил вывода используются машины логического вывода.

Модуль взаимодействия с онтологической моделью

Модуль взаимодействия с онтологической моделью позволяет с помощью языков запросов SQWRL и SPARQL и правил вывода SWRL выполнять запросы к онтологической модели. Также есть возможность взаимодействовать со сторонними программными системами, такими как Ontograf (визуализация онтологии в виде графа), Logic Text (представление текста в виде атомарных диаграмм) и др.

Схема работы с онтологической моделью представлена на рис. 1.

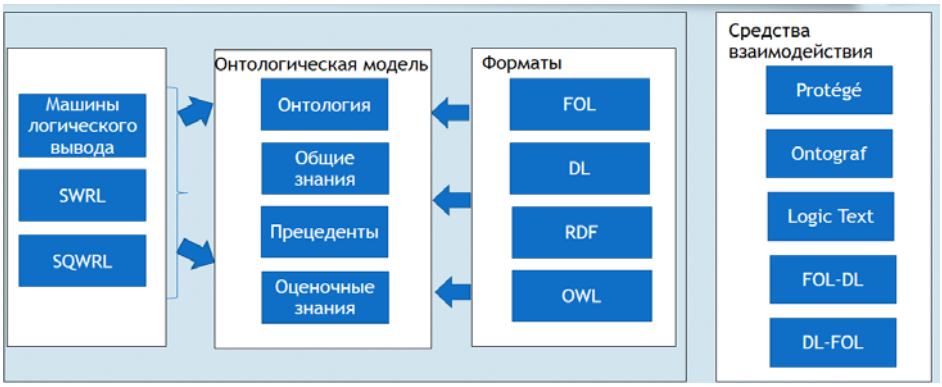


Рис. 1. Обобщенная схема работы с онтологической моделью

Fig. 1. General scheme of work with the ontological model

Автоматические системы логического вывода (ризонеры)

Семантический механизм рассуждений [7] – это часть программного обеспечения, способная вывести логические умозаключения (новые знания) из набора формализованных базовых понятий и представленных аксиом. Понятие семантического механизма рассуждений обобщает понятие машины логического вывода. Семантический механизм рассуждений предоставляет богатый набор механизмов для работы. Правила вывода обычно определяются с помощью языка описания онтологий. Многие семантические механизмы рассуждений используют логику предикатов первого порядка для выполнения рассуждений.

Для работы с онтологиями на языке OWL DL существует несколько основных машин логического вывода. Некоторые из них являются коммерческими программными продуктами: Bossam, RacerPro, OntoBroker [8]. Некоторые – свободными в использовании, но с закрытым исходным кодом: KAON2, Internet Business Logic, Cyc inference engine [9], а другие – бесплатным программным обеспечением с открытым исходным кодом: Drools [2], OpenRules [3], Pellet [10], Jena [4], Fact++ [11].

Для данной работы была выбрана машина логического вывода Pellet, так как она предоставляет наиболее полную поддержку OWL DL, а также имеет понятный программный интерфейс.

[2] Drools. Business Rules Management System. URL: http://www.drools.org
[3] OpenRules. User manual. URL: http://openrules.com/pdf/RulesSolver.UserManual.pdf
[4] Apache Jena. URL: https://jena.apache.org

Объединение правил вывода и онтологии

Применение машин логического вывода для онтологий можно осуществлять и без использования правил вывода. В таком случае машины логического вывода могут определять иерархию классов, принадлежность экземпляров к определенным классам, а также эквивалентность классов. Но новые знания в такой ситуации получить невозможно. Для этого нужно использовать правила вывода [12]. Для использования правил вывода при работе с онтологиями существуют различные инструменты: SWRL, DLP (Grosof), dl-programs (Eiter), DL-safe rules, Conceptual Logic Programs (CLP), AL-Log, DL+log [13].

Выделяют два основных подхода применения правил вывода в онтологиях: однородный (гомогенный) и гибридный [14].

Однородный подход

Однородный подход также называют семантической интеграцией. В однородном подходе онтологии и правила вывода используются на одинаковых условиях, т. е. создается единый язык. Одни и те же предикаты используются как для формулировки онтологических утверждений, так и для формулировки правил вывода. Правила в таком подходе можно использовать для определения классов и свойств. В данном подходе нет проблемы совместимости.

Недостатками данного подхода является то, что совмещение в одном языке разных средств (и для описания правил вывода, и для описания онтологии) сильно затрудняет реализацию. Также однородный подход часто может быть неприменим в работе из-за того, что онтология и правила вывода могут разрабатываться разными специалистами независимо друг от друга.

Примерами такого подхода являются SWRL [5] и DLP [15].

Гибридный подход

Гибридный подход отличается от гомогенного строгим семантическим разделением. Предикаты, используемые в онтологиях, и предикаты, используемые в правилах вывода, строго различаются. Вывод производится только с помощью отдельно реализуемых программ. Могут разрабатываться независимо от онтологий.

К недостаткам относится то, что правила не могут определять новых классов и свойств онтологий, за исключением некоторых специальных отношений. Хоть гибридный подход и позволяет разрабатывать правила вывода независимо от онтологий, это вносит ряд ограничений, для того чтобы гарантировать разрешимость.

Примерами такого подхода являются RIF [16] и Answer Set Programming (ASP) [17].

Системы для извлечения знания из текстов

В настоящее время существует ряд программных систем, предназначенных для анализа текстов естественного языка при помощи онтологий. Такими системами являются FRED [18], Apache Stanbol [6] и Text2Onto [19]. Рассмотрим особенности этих систем.

FRED – это анализатор текста, основанный на технологиях Semantic Web. Способен обрабатывать тексты естественного языка на 48 различных языках и преобразовывать его в связанные данные. Он разработан на языке Python, а также реализован в виде REST API и в виде набора библиотек языка Python. Для анализа текста FRED использует комбинаторную категориальную грамматику (C&C), теорию репрезентационного представления (DRT), семантику фреймов и шаблоны проектирования онтологий. Для определения именованного объекта анализатор FRED использует Apache Stanbol, TagMe, для устранения неоднозначности – Boxer и IMS. Результат обработки может представлять в формате RDF/OWL. Также FRED

[5] SWRL API: SWRL, SWRL API, SQWRL, SQWRL API. URL: https://github.com/protegeproject/swrlapi/wiki
[6] Apache Stanbol. Documentation Apache Stanbol. URL: https://stanbol.apache.org

генерирует графы знаний, которые могут быть интерпретированы машинами в соответствии с общей формальной семантикой. В основе работы лежат фреймы (Frame Semantic). Фрейм обычно выражается глаголами или другими лингвистическими конструкциями, которые могут распознаваться во входном тексте как *n*-арное отношение. Все *n*-арные отношения записываются как подклассы класса *dul:Event.* Предикаты представляются в виде классов, в названии которых есть сам объект. Пример такого именования: *fred:Love.* Вывод может быть представлен в виде графа (рис. 2).

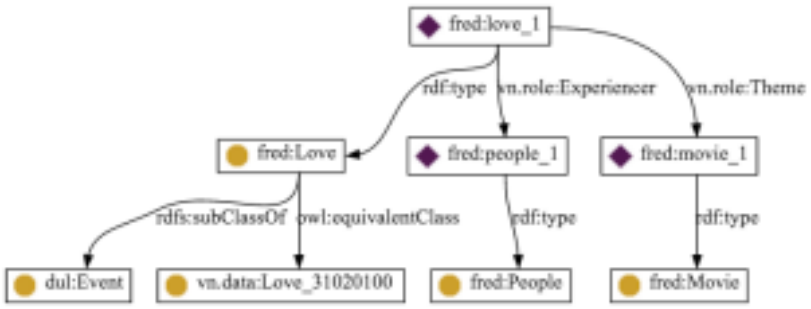


Рис. 2. Граф отношений
Fig. 2. Relationship graph

Text2Onto – программный продукт для пополнения онтологии из текстовых ресурсов. В основе его лежит вероятностная онтологическая модель (POM – Probabilistic Ontology Model). POM, используемая Text2Onto, представляет собой набор созданных примитивов моделирования, которые не зависят от конкретного языка представления онтологии. Фактически Text2Onto включает в себя библиотеку примитивов моделирования (MPL), которая определяет эти примитивы декларативным способом. Добавление новых примитивов происходит без изменения структуры онтологии. Это делает онтологию гибкой и расширяемой. Также созданные примитивы могут быть переведены на любой язык представления знаний, например RDFS3, OWL4 и F-Logic.

Text2Onto сочетает в себе машинное обучение и базовую лингвистическую обработку – токенизацию и разбор на предложения. Программа определяет в тексте некоторые виды отношений, такие как класс-подкласс, эквивалентность, экземпляры классов. Например, для выявления отношения подкласса в Text2Onto были реализованы различные алгоритмы, в частности реализовали алгоритм сопоставления шаблонов с использованием структуры WordNet.

Разработка программной системы

Краткое описание

Целью разрабатываемой программной системы является получение новых знаний на основе уже имеющихся знаний, содержащихся в текстах естественного языка, а также получения ответов на вопросы, относящиеся к этим текстам. В рамках работы в качестве примеров были взяты локальные нормативные документы факультета информационных технологий.

На вход программная система получает текстовый документ, который она преобразовывает в бесквантерные предложения логики предикатов. Для преобразования текстов в логику предикатов используется программа Logic Text [20]. С помощью данной программы можно получить фрагменты атомарных диаграмм по предложениям естественного языка.

Далее происходит преобразование многоместных предикатов в двухместные. Для того чтобы не потерять смысл предложений, вводятся константы, обозначающие ситуации. После этого полученные предложения, записанные в сигнатуре двухместных предикатов, транслируется в язык OWL.

На следующем шаге порождаются правила вывода по заранее созданным шаблонам. Шаблоны строятся для каждой предметной области отдельно. Для построения шаблонов используется логика предикатов рассматриваемой сигнатуры. С помощью машины логического вывода и созданных правил вывода в онтологическую модель добавляются новые знания. Для работы с онтологией на языке OWL и с правилами вывода, написанными на SWRL, используется машина логического вывода Pellet.

Шаблоны для запросов к онтологической модели также строятся на языке логики предикатов. Как и шаблоны для правил вывода, шаблоны для запросов строятся под конкретную предметную область. Заранее подготовленные шаблоны для запросов к онтологической модели заполняются с помощью пользовательского интерфейса, после чего происходит запрос к онтологической модели.

Учитывая новые выведенные знания, мы получаем ответ на запрос с помощью языка SQWRL, затем формулируем ответ на понятном для пользователя языке. В итоге программная система может отвечать на вопросы пользователей по текстам на естественном языке.

Подробное описание

Схема работы программной системы представлена на рис. 3.

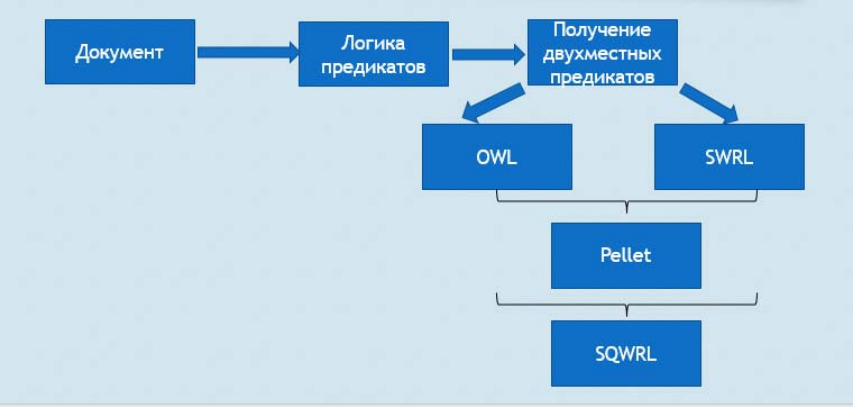


Рис. 3. Схема работы программной системы
Fig. 3. Relationship graph

В схеме работы можно выделить следующие основные этапы:

- преобразование текстовых документов выбранной предметной области в наборы атомарных предложений;
- конвертация фрагментов атомарных диаграмм в онтологическую модель;
- порождение правил вывода для построенной онтологической модели;
- вывод новых знаний;
- проверка полученных результатов.

Рассмотрим подробно каждый из этапов.

Преобразование текстовых документов выбранной предметной области в наборы атомарных предложений

Для построения по текстам естественного языка фрагмента атомарной диаграммы используется программная система LogicText [21]. Знания, представленные в тексте, формализуют-

ся на языке многоместных предикатов. Однако многоместные предикаты нельзя представить на языке OWL, так как OWL поддерживает только двухместные предикаты. Поэтому нужно преобразовать многоместные предикаты в двухместные.

Конвертация фрагментов атомарных диаграмм в онтологическую модель. Онтологическая модель строится на основе четырехуровневой модели представления знаний [1; 2]: это онтология, общие знания, прецеденты, оценочные знания. В данном случае нам потребуются только первые три уровня.

Рассмотрим уровни онтологической модели на примере нормативных документов факультета.

Первый уровень – это онтология [22]. Онтология представляет собой формальное описание языка предметной области, цель которого – в явном виде определить смысл терминов, используемых в данной предметной области. Ключевые термины – это понятия, смысл которых, во-первых, является специфичным для данной предметной области и, во-вторых, одинаково понимается всеми экспертами предметной области.

Примерами таких знаний о смысле понятий являются: «*ВКР – это выпускная квалификационная работа. Студент является обучающимся. ГИА – это государственная итоговая аттестация*».

Второй уровень – это общие, универсальные знания о предметной области, которые считаются полностью истинными (на данный момент времени). В рамках формального описания учебного процесса это знания, извлеченные из различных нормативных документов. Общие знания извлекаются из текстов и содержат общую информацию о предметной области. Например: «*Занятия в учебном году начинаются 1-го сентября. Предзащита ВКР проходит в срок с 15 по 31 мая четвертого семестра обучения*».

Третий уровень онтологической модели – прецеденты (или частные знания). Это знания, которые истины в конкретной ситуации, т. е. для определенного экземпляра предметной области. Например: «*В 2019 году государственные аттестационные испытания по направлению подготовки 09.03.01 Информатика и вычислительная техника (уровень бакалавриата) проводятся с 24.06.2019 по 26.06.2019*».

Для пополнения онтологической модели знаниями, извлеченными из текстов естественного языка, происходит преобразование многоместных предикатов в двухместные. Для этого используется алгоритм, представленный в [23]. Преобразование многоместных предикатов в двухместные происходит с помощью введения сигнатуры дополнительных констант-ситуаций.

Порождение правил вывода. Для пополнения онтологической модели новыми знаниями необходимо создавать правила вывода. Чтобы не строить каждый раз правила вывода вручную, используются шаблоны правил вывода. Заполнение шаблонов правил вывода происходит в момент создания онтологической модели.

Шаблоны строятся для конкретной предметной области. При переводе n-местных предикатов в двухместные вводятся специальные отношения между константами-ситуациями [23]. Например:

- $Include(s_1, s_2)$ – отношение, обозначающее, что ситуация s_1 является частью более общей ситуации s_2;
- $Id(s_1, s_2)$ – отношение, обозначающее, что ситуации s_1 и s_2 эквивалентны.

Для отношения Id справедливы, например, следующие правила вывода:

$$\big(Id(s_1, s_2) \rightarrow Id(s_2, s_1)\big)$$

$$\Big(\big(Id(s_1, s_2)\ \&\ P(s_1, x)\big) \rightarrow P(s_2, x)\Big)$$

А для отношения $Include$ справедливы правила вывода:

$$\Big(\big(Include(s_1, s_2)\ \&\ Include(s_2, s_1)\big) \rightarrow Id(s_1, s_2)\Big)$$

$$\Big(\big(Include(s_1, s_2)\ \&\ Include(s_2, s_3)\big) \rightarrow Include(s_1, s_3)\Big)$$

Эти правила вывода являются примерами шаблонов. Вместо P мы можем подставить любой предикат из сигнатуры онтологической модели. s_1 и s_2 тоже могут быть любыми ситуациями, которые связанны предикатами $Include$ или Id соответственно. Шаблоны правил вывода могут заполняться автоматически и добавляться в онтологическую модель уже как готовые правила вывода.

Порождение новых знаний. Для получения по онтологической модели новых знаний используются правила вывода, построенные на предыдущем этапе, и машина логического вывода Pellet. Сначала машина логического вывода проверяет, что созданная онтологическая модель не является противоречивой. Если мы выявили, что какие-либо знания противоречат друг другу, то специальная утилита для редактора онтологий Protégé подскажет, какие утверждения вступают в противоречие. Разрешить такие противоречия в рамках разрабатываемой системы можно только вручную.

После того как машина логического вывода подтвердила, что онтологическая модель является непротиворечивой, мы переходим к порождению новых знаний. Система Pellet использует порожденные ранее правила вывода и добавляет новые знания в онтологическую модель.

Проверка полученных результатов. Чтобы проверить, получилось ли вывести новые знания, используются запросы к онтологической модели, которые также строятся по шаблонам. Шаблоны запросов порождаются для конкретной предметной области аналогично шаблонам правил вывода.

Редактирование шаблона происходит пользователем через графический интерфейс. Шаблон имеет вид какого-либо вопроса. Например: «*Что должен сделать X?»* Вместо X может быть подставлен любой класс из онтологической модели, который связан с ситуацией свойством *кто* или *что*. При этом можно выбрать любой вопрос, который заполнился по шаблону. Если заполнение шаблона не может произойти однозначно, то пользователю предоставляется возможность самому выбрать правильный вопрос. Например, если вместо X можно подставить несколько слов: *студент*, *рецензент*, *научный руководитель* и т. д., то на экране появится выпадающий список со всеми этими словами. Пользователь сам выберет нужное и задаст вопрос. Также пользователь может задать несколько вопросов, по очереди выбирая нужное слово.

После того как пользователь сформировал вопрос, он представляется в виде SQWRL запроса к онтологической модели. Так как SQWRL имеет доступ и к тем знаниям, которые были выведены с помощью машины логического вывода, то в ответ пользователю попадет не только та информация, которая в явном виде содержится в текстах, но и та, которая была выведена с помощью правил вывода.

SQWRL возвращает ответ в виде таблицы. По данным из таблицы, учитывая ситуацию, с которой связано это слово, мы строим текстовый ответ пользователю и отображаем его на экране.

Чтобы проверить, какие знания получилось вывести, можно, например, задать следующие вопросы:

- *Что должен сделать студент?*
- *Когда научный руководитель должен предоставить отзыв на ВКР?*
- *Что должен сделать рецензент?*
- *и другие*

Рассмотрим ответ на вопрос: «*Что должен сделать рецензент?*». По этому вопросу составится SQWRL запрос вида

$$кто(?x, рецензент) \& \text{ что делать } (?x, ?y) \rightarrow sqwrl{:}\, select(?y)$$

Это основной запрос, по нему получим информацию о том, что должен сделать рецензент. В данном случае это будет: *проводить* и *предоставить*. Чтобы ответ получился полным, нужно сделать запросы, уже используя полученную информацию. В итоге получим ответ: *Рецензент должен проводить анализ и предоставлять письменные рецензии.* Далее пользователю предлагаются уточняющие вопросы. Например, *когда рецензент должен предоста-*

вить письменные рецензии? В ответ получим: *с 15 мая 4 семестра обучения до 31 мая 4 семестра обучения.*

Разработка шаблонов правил вывода и запросов к онтологической модели

Была разработана программная система на языке Java, работающая посредством OWL API с онтологической моделью, представленной на языке OWL 2 DL.

Разработка шаблонов правил вывода

Шаблоны правил вывода создаются индивидуально для каждой предметной области. Выше были рассмотрены примеры шаблонов, в частности:

$$\Big(\big(Id(s_1, s_2)\ \&\ P(s_1, x)\big)\ \rightarrow P(s_2, x)\Big)$$

Помимо шаблонов правил вывода со специальными отношениями, мы рассматриваем шаблоны, которые не связаны со специальными отношениями:

$$\textit{с какой даты}(?\,x, ?\,y)\&\ \textit{до какой даты}\ (?\,x, ?\,z)\&$$
$$swrlb\colon subtractDayTimeDurations(?\,d, ?\,y, ?\,z) \rightarrow \textit{иметь период}\ (?\,x, ?\,d)$$

Подставляя в шаблоны конкретные константы-ситуации и конкретные предикаты, программа порождает правила вывода. Это происходит автоматически через OWL API.

Разработка шаблонов запросов SQWRL

Запросы к онтологической модели строятся с помощью SQWRL. Для данной предметной области можно определить основные шаблоны запросов:

- *Что должен сделать X?*
- *Когда X должен сделать Y Z?*
- *Сколько времени есть у X для выполнения задачи Y?*
- *Сколько задач нужно сделать X в период Y?*

Пользователю выводятся данные шаблоны, вместо *X* или *Y* предлагаются на выбор слова. Слова берутся из онтологической модели. Например, для первого вопроса «*Что должен сделать X?*» для того, чтобы заполнить возможными значениями переменную *X*, в онтологическую модель делается запрос:

$$\textit{кто}(?\,x, ?\,y) \rightarrow sqwrl\colon select(?\,y)$$

Переменная *x* не содержится в $sqwrl\colon select$ потому, что нам не важно в данном случае, к какой ситуации относятся выведенные константы. Аналогичные запросы выполняются для всех переменных.

После того как пользователь сделает выбор по поводу заполнения переменных, создаются запросы к онтологии.

Ответы пользователям на естественном языке также строятся по шаблонам. Шаблоны ответов строятся аналогично шаблонам вопросов. Так, например, к шаблону вопроса *Что должен сделать X? с*троится шаблон ответа *X должен Y*. Здесь *Y* – это ответ, полученный с помощью SQWRL-запроса.

В общем виде из онтологической модели с помощью SQWRL-запроса может быть получено несколько действий *Y* объекта *X*. Обобщая шаблон на такой случай, получаем:

X должен Y и/или Z, и/или S, и/или … .

Рассмотрим примеры вопросов и их представление на SQWRL. Ответы представим сразу в читаемом для пользователя виде – на естественном языке.

- Когда *студент* должен *предъявить первый вариант ВКР?*

$$\textit{кто}(?\,x, \textit{студент})\ \&\ \textit{что делать}(?\,x, \textit{предъявлять})$$
$$\&\ \textit{что}(?\,x, \textit{первый вариант ВКР})$$
$$\&\ \textit{когда}(?\,x, ?\,a) {-}{>}\ sqwrl\colon select(?\,a)$$

На такой запрос мы не сможем получить ответ, так как промежутка времени не задано в предложении. Поэтому нужно сделать два других запроса:

$$кто(?\,x, студент)\ \&\ что\ делать(?\,x, предъявлять)$$
$$\&\ что(?\,x, первый\ вариант\ ВКР)$$
$$\&\ \text{до\_какой\_даты}(?\,x, ?\,a) -> \ sqwrl{:}\,select(?\,a)$$

или

$$кто(?\,x, студент)\ \&\ что\ делать(?\,x, предъявлять)$$
$$\&\ что(?\,x, первый\ вариант\ ВКР)$$
$$\&\ \text{с\_какой\_даты}(?\,x, ?\,a) -> \ sqwrl{:}\,select(?\,a)$$

Получим ответ по шаблону: X должен Y до Z.

Студент должен предъявить первый вариант ВКР до 15 апреля второго семестра обучения.

- Сколько времени есть у *научного руководителя* для выполнения задачи *отзыв на ВКР?*

$$кто(?\,x, научный\ руководитель)\ \&\ что\ (?\,x, отзыв\ на\ ВКР)$$
$$\&\ иметь\ период\ (?\,x, ?\,a) -> \ sqwrl{:}\,select(?\,a)$$

В запросах, начинающихся на «*сколько*», в ответ пользователю выдаем просто значение, полученное с помощью SQWRL.

Получим ответ*:16 дней.*

Апробация программной системы

Тестирование программной системы проходило на документах ФИТ НГУ, а именно: «09.04.01 Программа государственной итоговой аттестации по образовательной программе высшего образования – программе магистратуры», «Приказы о проведении сессий». Было обработано 30 предложений, содержащих информацию о сроках проведения этапов обучения, представления документов на кафедру и порядке проведения защиты квалификационной работы. По каждому предложению было составлено порядка 10 двухместных предикатов. По подготовленным предикатам составлена онтологическая модель, содержащая 367 аксиом. Автоматически были созданы 50 правил вывода. Программный продукт может отвечать на 6 видов вопросов, а именно: *Кто?, Что?, Что сделать?, Когда?, Какой промежуток?, Сколько видов работ?*

Заключение

Статья посвящена разработке методов извлечения знаний из текстов на естественном языке и порождения новых знаний, явно не содержащихся в текстах. Была создана программная система, предназначенная для порождения новых знаний на основе обработки текстов естественного языка. Разработанная программная система позволяет извлекать знания из текстов и представлять их в формальном виде в онтологической модели. Онтологическая модель строится на основе четырехуровневой модели представления знаний. Использование онтологической модели позволяет порождать новые знания с помощью добавления новых правил вывода, а также проверять полученные знания на непротиворечивость с помощью машины логического вывода.

Программная система дает возможность пользователям задавать вопросы на естественном языке, относящиеся к рассматриваемым документам. Ответы на вопросы строятся динамически, исходя из знаний, содержащихся в онтологической модели. При составлении ответа на вопрос используются как знания, явно содержащиеся в текстах, так и новые знания, порожденные в онтологической модели.

Список литературы

1. **Найданов Ч. А., Пальчунов Д. Е., Сазонова П. А.** Теоретико-модельные методы интеграции знаний, извлеченных из медицинских документов // Вестник НГУ. Серия: Информационные технологии. 2015. Т. 13, № 3. С. 29–41.
2. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories. *Siberian Scientific Medical Journal*, 2016, vol. 36, iss. 1, p. 105–113.

3. **Маслов В. А., Соколов С. М.** Обработка семантических запросов в среде Protégé на примере построения онтологии дорожных знаков // Препринты ИПМ им. М. В. Келдыша. 2018. № 260. 15 с. DOI 10.20948/prepr-2018-260
4. **Асламова В. С., Берестнева О. Г., Темникова Е. А.** Онтологическое моделирование предметной области учреждения дополнительного образования // Онтология проектирования. 2015. Т. 5, № 4 (18).
5. **Ефименко И. В., Хорошевский В. Ф.** Онтологическое моделирование экономики предприятий и отраслей современной России. Ч. 3: Российские исследования и разработки в области онтологического инжиниринга и бизнес-онтологии. М.: ИД ВШЭ, 2011. 68 с.
6. **Найданов Ч. А.** Разработка ядра онтологической модели, настраиваемой под предметную область // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 1. С. 72–81. DOI 10.25205/1818-7900-2019-17-1-72-81
7. **Horrocks I.** Description logic reasoning. In: International Conference on Logic Programming and Automated Reasoning. Montevideo, Uruguay, 2000.
8. **Гаврилова Т. А., Хорошевский В. Ф.** Базы знаний интеллектуальных систем. СПб: Питер, 2000. 384 с.
9. **Baxter D., Shepard D., Siegel N., Gottesman B., Schneider D.** Interactive Natural Language Explanations of Cyc Inferences. In: AAAI 2005 International Symposium on Explanation-aware Computing, 2005. URL: https://www.aaai.org/Papers/Symposia/Fall/2005/FS-05-04/FS05-04-002.pdf
10. **Zuo M., Haarslev V.** Intelligent Tableau Algorithm for DL Reasoning. In: Proceedings of the 22nd International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. France, Nancy, 2013, p. 273–287. URL: https://link.springer.com/chapter/10.1007/978-3-642-40537-2_23
11. **Tsarkov D., Horrocks I.** FaCT++ description logic reasoner: System description. In: Third International Joint Conference on Automated Reasoning, 2006, p. 292–297. URL: http://dl.acm.org/citation.cfm?id=2136140
12. **Курейчик В. В., Бова В. В.** Моделирование процесса представления знаний в интеллектуальных обучающих системах на основе компетентностного подхода // Открытое образование. 2014. № 3. URL: https://cyberleninka.ru/article/n/modelirovanie-protsessa-predstavleniya-znaniy-v-intellektualnyh-obuchayuschih-sistemah-na-osnove-kompetentnostnogo-podhoda
13. **Heymans S., Nieuwenborgh D. V., Vermeir D.** Conceptual logic programs. *Annals of Mathematics and Artificial Intelligence*, 2006, vol. 47 (1–2), p.103–137.
14. **Авдошин С. М., Шатилов М. П.** Онтологический инжиниринг // Бизнес-информатика. 2007. № 2. С. 32–36.
15. **Papadimitriou P., Garcia-Molina H.** Data Leakage Detection. URL: ilpubs.stanford.edu:8090/839/1/2008-23.pdf
16. **Калиниченко Л. А., Ступников С. А.** Анализ мотивации, целей и подходов проекта унификации языков на правилах. URL: http://synthesis.ipi.ac.ru/synthesis/publications/11ont-uni/11ont-uni.pdf
17. **Lifschitz V.** What is answer set programming? In: Proceedings of AAAI, 2008.
18. **Gangemi A., Presutti V., Reforgiato Recupero D., Giovanni Nuzzolese A., Draicchio F., Mongiovì M.** Semantic Web Machine Reading with FRED. *Semantic Web Journal*, 2017, vol. 8 (6), p. 873–893.
19. **Cimiano P., Völker J.** Text2onto. *Nat. Lang. Process Inform. Syst*., 2005, vol. 3513, p. 227–238. DOI 10.1007/11428817_21
20. **Махасоева О. Г., Пальчунов Д. Е.** Автоматизированные методы построения атомарной диаграммы модели по тексту естественного языка // Вестник НГУ. Серия: Информационные технологии. 2014. Т. 12, № 2. С. 64–73.

21. **Махасоева О. Г., Пальчунов Д. Е.** Программная система построения атомарной диаграммы модели по тексту естественного языка. Св-во о гос. регистрации программы для ЭВМ № 2014619198, зарегистрировано 10.09.2014.
22. **Gruber T. R.** Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: International Workshop on Formal Ontology. Padova, Italy, 1993.
23. **Ненашева Е. О., Пальчунов Д. Е.** Разработка автоматизированных методов преобразования предложений естественного языка в бескванторные формулы логики предикатов // Вестник НГУ. Серия: Информационные технологии. 2017. Т. 15, № 3. С. 49–63. DOI 10.25205/1818-7900-2017-15-3-49-63

References

1. **Naidanov Ch. A., Palchunov D. E., Sazonova P. A.** Model-theoretic methods of integration of knowledge extracted from medical documents. *Vestnik NSU. Series: Information Technologies*, 2015, vol. 13, no. 3, p. 29–41. (in Russ.)
2. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories. *Siberian Scientific Medical Journal*, 2016, vol. 36, iss. 1, p. 105–113.
3. **Maslov V. A., Sokolov S. M.** Processing of semantic queries in the Protégé environment using the example of constructing an ontology of road signs. In: Preprint IPM im. M. V. Keldysh, 2018, no. 260, 15 p. DOI 10.20948 / prepr-2018-260 (in Russ.)
4. **Aslamova V. S., Berestneva O. G., Temnikova E. A.** Ontological modeling of the subject area of the institution of additional education. *Design Ontology*, 2015, vol. 5, no. 4 (18). (in Russ.)
5. **Efimenko I. V., Khoroshevsky V. F.** Ontological modeling of the economics of enterprises and branches of modern Russia. Part 3. Russian research and development in the field of ontological engineering and business ontology. Moscow, Higher School of Economics Press, 2011, 68 p. (in Russ.)
6. **Naydanov Ch. A.** Development of an Ontological Model Core Customizable for a Specific Subject Domain. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 1, p. 72–81. (in Russ.) DOI 10.25205/1818-7900-2019-17-1-72-81
7. **Horrocks I.** Description logic reasoning. In: International Conference on Logic Programming and Automated Reasoning. Montevideo, Uruguay, 2000.
8. **Gavrilova T. A., Khoroshevsky V. F.** Knowledge Base of Intellectual Systems. St. Petersburg, Peter, 2000, 384 p.
9. **Baxter D., Shepard D., Siegel N., Gottesman B., Schneider D.** Interactive Natural Language Explanations of Cyc Inferences. In: AAAI 2005 International Symposium on Explanation-aware Computing, 2005. URL: https://www.aaai.org/Papers/Symposia/Fall/2005/FS-05-04/FS05-04-002.pdf
10. **Zuo M., Haarslev V.** Intelligent Tableau Algorithm for DL Reasoning. In: Proceedings of the 22nd International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. France, Nancy, 2013, p. 273–287. URL: https://link.springer.com/chapter/10.1007/978-3-642-40537-2_23
11. **Tsarkov D., Horrocks I.** FaCT++ description logic reasoner: System description. In: Third International Joint Conference on Automated Reasoning, 2006, p. 292–297. URL: http://dl.acm.org/citation.cfm?id=2136140
12. **Kureichik V. V., Bova V. V.** Modeling the process of knowledge representation in intelligent tutoring systems based on the competence approach. *Open Education*, 2014, no. 3. URL: https://cyberleninka.ru/article/n/modelirovanie-protsessa-predstavleniya-znaniy-v-intellektualnyh-obuchayuschih-sistemah-na-osnove-kompetentnostnogo-podhoda (in Russ.)
13. **Heymans S., Nieuwenborgh D. V., Vermeir D.** Conceptual logic programs. *Annals of Mathematics and Artificial Intelligence*, 2006, vol. 47 (1–2), p.103–137.

14. **Avdoshin S. M., Shatilov M. P.** Ontological engineering. Business Informatics, 2007, no. 2, p. 32–36. (in Russ.)
15. **Papadimitriou P., Garcia-Molina H.** Data Leakage Detection. URL: ilpubs.stanford.edu: 8090/839/1/2008-23.pdf
16. **Kalinichenko L. A., Stupnikov S. A.** Analysis of the motivation, goals and approaches of the project of the unification of languages on the rules. URL: http://synthesis.ipi.ac.ru/synthesis/publications/11ont-uni/11ont-uni.pdf
17. **Lifschitz V.** What is answer set programming? In: Proceedings of AAAI, 2008.
18. **Gangemi A., Presutti V., Reforgiato Recupero D., Giovanni Nuzzolese A., Draicchio F., Mongiovì M.** Semantic Web Machine Reading with FRED. *Semantic Web Journal*, 2017, vol. 8 (6), p. 873–893.
19. **Cimiano P., Völker J.** Text2onto. *Nat. Lang. Process Inform. Syst*., 2005, vol. 3513, p. 227–238. DOI 10.1007/11428817_21
20. **Makhasoeva O. G., Palchunov D. E.** Semi-automatic methods of a construction of the atomic diagrams from natural language texts. *Vestnik NSU. Series: Information Technologies*, 2014, vol. 12, no. 2, p. 64–73. (in Russ.)
21. **Makhasoeva O. G., Palchunov D. E.** Software system for constructing an atomic diagram of a model from the text of a natural language. Certificate of state registration of computer programs № 2014619198, registered 10.09.2014.
22. **Gruber T. R.** Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: International Workshop on Formal Ontology. Padova, Italy, 1993.
23. **Nenasheva E. O., Palchunov D. E.** Semi-Automated Methods of Transforming Sentences from Natural Language into Quantifier-Free Formulas of Predicate Logic. *Vestnik NSU. Series: Information Technologies*, 2017, vol. 15, no. 3, p. 49–63. (in Russ.)

Материал поступил в редколлегию
Received
25.06.2019

Сведения об авторах / Information about the Authors

Капустина Алина Игоревна, студент, 2 курс магистратуры, факультет информационных технологий, Новосибирский государственный университет (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Alina I. Kapustina, Student, 2 year master course, Department of Information Technology, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

a.kapustina@g.nsu.ru

Пальчунов Дмитрий Евгеньевич, доктор физико-математических наук, ведущий научный сотрудник, Институт математики СО РАН (пр. Академика Коптюга, 4, Новосибирск, 630090, Россия), заведующий кафедрой ОИ ФИТ, Новосибирский государственный университет (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Dmitry E. Palchunov, Doctor of Physical and Mathematical Sciences, Leading Researcher, Sobolev Institute of Mathematics SB RAS (4 Academician Koptyug Ave., Novosibirsk, 630090, Russian Federation); Head of the Department of General Informatics of the Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

palch@math.nsc.ru

УДК 004.89
DOI 10.25205/1818-7900-2019-17-3-43-60

Разработка музыкальной рекомендательной системы на основе обработки метаданных контента

А. В. Менькин

Новосибирский государственный университет
Новосибирск, Россия

Аннотация

Музыкальная рекомендательная система (МРС) помогает пользователям музыкальных стриминговых сервисов находить интересующий их музыкальный контент. Разреженность пользовательских оценок – одна из главных проблем исследования МРС. Она вызвана тем, что пользователь оценивает лишь малую долю объектов музыкального каталога. В результате МРС часто не обладает достаточным набором данных для составления рекомендаций.

В статье предложен подход для решения проблемы разреженности пользовательских оценок на основе использования оценок связанных объектов. Описана гибридная МРС, использующая как нормализованные пользовательские оценки треков, альбомов, артистов, жанров, так и информацию о связях между объектами разных типов. Произведена оценка эффективности разработанной МРС, а также произведен сравнительный анализ предложенного подхода с коллаборативным методом предсказания пользовательских предпочтений.

Ключевые слова

музыкальные рекомендательные системы, разреженность пользовательских оценок, метаданные, музыкальный контекст

Для цитирования

Менькин А. В. Разработка музыкальной рекомендательной системы на основе обработки метаданных контента // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 43–60. DOI 10.25205/1818-7900-2019-17-3-43-60

Development of a Music Recommender System Based on Content Metadata Processing

A. V. Menkin

Novosibirsk State University
Novosibirsk, Russian Federation

Abstract

Music recommender systems (MRS) help users of music streaming services to find interesting music in the music catalogs. The sparsity problem is an essential problem of MRS research. It refers to the fact that user usually rates only a tiny part of items. As a result, MRS often has not enough data to make a recommendation.

To solve the sparsity problem, in this paper, a new approach that uses related items' ratings is proposed. Hybrid MRS based on this approach is described. It uses tracks, albums, artists, genres normalized ratings along with information about relations between items of different types in the music catalog. The proposed MRS is evaluated and compared to collaborative method for users' preferences prediction.

Keywords

music recommender systems, sparsity, metadata, music context

For citation

Menkin A. V. Development of a Music Recommender System Based on Content Metadata Processing. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 43–60. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-43-60

Введение

Рекомендательные системы – это программные инструменты и методики, которые предлагают пользователям объекты, наиболее интересные для них [1]. Музыкальная рекомендательная система (МРС) предлагает пользователям музыкального сервиса объекты музыкального каталога. Основными задачами и направлениями исследований МРС являются:

1) предсказание пользовательских оценок (треков, альбомов, артистов, жанров и др.);
2) автоматическое тегирование;
3) автоматическое составление (продолжение) плейлиста;
4) контекстно-ориентированные системы.

Задача предсказания пользовательских оценок решается для составления рекомендаций [2–5] и для восполнения пропусков в разреженной матрице пользовательских оценок [6]. Семантические дескрипторы (теги) позволяют описать объекты музыкального каталога в терминах, понятных для пользователя [7; 8], поэтому они удобны для поиска музыки. Этим вызван интерес к задачам автоматического тегирования, например, классификации жанров и распознавания эмоций [9]. Так как пользователь обычно слушает треки в составе последовательности [10], задача автоматического составления (продолжения) плейлиста [8; 11] также представляет интерес. Она связана с выбором треков, соответствующих предопределенным характеристикам (например, семантическим дескрипторам [8]), порядком выбранных треков и др. [10]. Контекст пользователя (локация, время, деятельность и др.) значительно влияет на его ситуативные предпочтения [10], поэтому разработка контекстно-ориентированных систем [8] является перспективным направлением.

Для решения задач исследований МРС используются три вида данных:

1) контентные данные;
2) музыкальный контекст;
3) пользовательские данные.

Контентные данные [12] – это значения акустических признаков, полученные в результате обработки аудиосигнала [2; 3; 8; 13]. Музыкальный контекст [14] – это данные, которые описывают музыкальные объекты, но не являются результатом обработки аудиосигнала. Например, текстовые описания объектов, информация о связях между объектами разных типов, теги, поставленные экспертной группой, и др. Пользовательские данные [15] включают в себя персональные данные [3; 5], историю воспроизведений [4; 11; 16], историю сессий (повторные воспроизведения треков, пропуски треков и др.) [4; 6], контекст пользователя [6; 8], поставленные оценки [2; 3; 5; 6], присвоенные теги [11; 13] и др. Также используются данные о взаимодействии пользователей с другими видами контента [5].

При решении задач исследований МРС возникают следующие проблемы:

1) холодный старт (новый пользователь, новый объект);
2) разреженность пользовательских оценок.

Проблема холодного старта [10] является обобщением проблемы нового пользователя и проблемы нового объекта. Проблема нового пользователя возникает, когда пользователь регистрируется в музыкальном сервисе, и МРС не обладает данными об этом пользователе, чтобы составить для него рекомендацию. Для решения этой проблемы применяются опрос пользователя при регистрации и кросс-доменный подход [5]. Проблема нового объекта возникает, когда объект добавляется в каталог, и МРС не обладает данными об этом объекте, чтобы включить его в рекомендацию. Для решения этой проблемы применяются контентно-ориентированный подход [2] и контентные гибридные системы [13; 16].

Проблема разреженности пользовательских оценок [10] возникает, когда число поставленных оценок значительно меньше числа возможных. Разреженность определяется согласно формуле

$$sparsity = \left(1 - \frac{N_R}{N_U \times N_I}\right) \times 100\%, \quad (1)$$

где N_U – число пользователей, N_I – число объектов, N_R – число пользовательских оценок (пользователь может поставить только одну оценку объекту).

Эта проблема особенно актуальна для исследований МРС. Из-за того, что музыкальные каталоги содержат огромное число объектов (десятки миллионов треков [10]), пользователь обычно взаимодействует лишь с малой их долей. Также пользователь обычно слушает треки в составе последовательности и не прерывает воспроизведение, чтобы оценить их. Более того, пользователь нередко не концентрируется на воспроизводимых треках и не оценивает их. В результате доля объектов, оцененных пользователем, обычно близка к нулю. Значение разреженности одного из наибольших наборов данных "C15 – Yahoo! Music user ratings of musical tracks, albums, artists and genres, v 1.0" составляет 99,96 % (набор данных содержит пользовательские оценки треков, альбомов, артистов, жанров). Для сравнения: разреженность набора данных от Netflix составляет 98,82 % (набор данных содержит пользовательские оценки фильмов и сериалов) [10]. Разница более чем в процент существенна.

Для решения этой проблемы применяются контентные гибридные системы и методы, в которых вычисляются неявные пользовательские оценки. В контентных гибридных системах [2] обычно для предсказания пользовательской оценки трека используются оценки треков похожих по значениям акустических признаков. Также может быть обучена модель [3], которая использует значения акустических признаков совместно с персональными данными. В методах, основанных на неявных оценках [4; 6] для треков, с которыми пользователь взаимодействовал, но которые не оценивал, вычисляются неявные оценки, которые восполняют пропуски в разреженной матрице пользовательских оценок.

Для решения проблемы разреженности пользовательских оценок предлагается использовать оценки связанных (согласно метаданным контента) объектов музыкального каталога как похожих. В статье описана гибридная МРС, в которой применен предложенный подход. Этапы работы МРС:

1) определение похожих альбомов, артистов, жанров;
2) предварительный выбор треков;
3) определение похожих пользователей;
4) вычисление значений признаков;
5) обучение модели для предсказания пользовательских оценок треков;
6) предсказание пользовательских оценок треков;
7) составление рекомендации.

На этапах 1–2 для пользователя предварительно выбираются треки. Для этого определяются альбомы, артисты, жанры, которые (предположительно) нравятся ему, и выбираются треки, связанные с ними. На этапах 3–6 для каждого выбранного трека предсказывается оценка. Для этого используются оценки, поставленные похожими пользователями как этому треку, так и связанным с ним объектам. На этапе 7 выбираются треки с наибольшими предсказанными оценками.

Музыкальная рекомендательная система

Объекты представлены в виде целочисленных идентификаторов. Множество объектов I определено согласно формуле

$$I = \{0 \dots N_I - 1\},$$

где N_I – число объектов.

Метаданные контента содержат информацию о типах объектов и о связях между объектами разных типов. В результате обработки метаданных контента определены:

1) I_{tracks}, I_{albums}, $I_{artists}$, I_{genres} – разбиение множества I на множества треков, альбомов, артистов, жанров;

2) $c_{track,album}: I_{tracks} \to I_{albums} \cup \{None\}$ – отображение, которое сопоставляет с треком связанный альбом. Если трек не связан ни с одним альбомом, то сопоставляется $None$;

3) $c_{track,artist}: I_{tracks} \to I_{artists} \cup \{None\}$ – отображение, которое сопоставляет с треком связанного артиста. Если трек не связан ни с одним артистом, то сопоставляется $None$;

4) $c_{track,genres}: I_{tracks} \to \mathcal{P}(I_{genres})$ – отображение, которое сопоставляет с треком множество связанных жанров;

5) $c_{album,artist}: I_{albums} \to I_{artists} \cup \{None\}$ – отображение, которое сопоставляет с альбомом связанного артиста. Если альбом не связан ни с одним артистом, то сопоставляется $None$;

6) $c_{album,genres}: I_{albums} \to \mathcal{P}(I_{genres})$ – отображение, которое сопоставляет с альбомом множество связанных жанров.

Также определены вспомогательные отображения:

1) $c_{album,tracks}: I_{albums} \to \mathcal{P}(I_{tracks})$ – отображение, которое сопоставляет с альбомом множество связанных треков;

2) $c_{artist,tracks}: I_{artists} \to \mathcal{P}(I_{tracks})$ – отображение, которое сопоставляет с артистом множество связанных треков;

3) $c_{artist,albums}: I_{artists} \to \mathcal{P}(I_{albums})$ – отображение, которое сопоставляет с артистом множество связанных альбомов;

4) $c_{genre,tracks}: I_{genres} \to \mathcal{P}(I_{tracks})$ – отображение, которое сопоставляет с жанром множество связанных треков;

5) $c_{genre,albums}: I_{genres} \to \mathcal{P}(I_{albums})$ – отображение, которое сопоставляет с жанром множество связанных альбомов.

Пользователи представлены в виде целочисленных идентификаторов. Множество пользователей U определено согласно формуле

$$U = \{0 \dots N_U - 1\},$$

где N_U – число пользователей.

МРС использует пользовательские оценки объектов:

1) $r: U \times I \to [0,1] \cup \{None\}$ – отображение, которое сопоставляет с пользователем и объектом нормализованную оценку. Если пользователь не оценивал объект, то сопоставляется $None$;

2) $r_2: U \times I \to \{0, 1, None\}$ – отображение, которое сопоставляет с пользователем и объектом бинарную оценку. Определено согласно формуле

$$r_2(u,i) = \begin{cases} 0, & r(u,i) \in [0, t_r) \\ 1, & r(u,i) \in [t_r, 1] \\ None, & r(u,i) = None \end{cases}, \quad t_r \in (0,1],$$

где t_r – пороговое значение. Если $r(u,i)$ достигает t_r, то считается, что пользователю u нравится объект i. Если пользователь не оценивал объект, то сопоставляется $None$.

Определение похожих альбомов, артистов, жанров

Рассмотрим на примере альбомов. В первую очередь по формуле

$$I'_{albums} = \{i \in I_{albums} | c_{album,tracks}(i) \neq \emptyset\}$$

определяется множество альбомов, связанных хотя бы с одним треком, I'_{albums}.

Для каждого альбома $i \in I'_{albums}$ согласно формуле

$$v_i(u) = \begin{cases} -1, & r_2(u,i) = 0 \\ 1, & r_2(u,i) = 1 \\ 0, & r_2(u,i) = None \end{cases}, \quad u \in U,$$

строится вектор v_i.

Компонента $v_i(u)$ соответствует пользователю u и содержит значение из множества $\{-1, 1, 0\}$ в зависимости от того, оценивал ли пользователь альбом, и если оценивал, то понравился ли альбом ему. Далее, для каждой пары альбомов $i, i' \in I'_{albums}$ $(i \neq i')$ на основе векторов v_i и $v_{i'}$ вычисляется значение коэффициента схожести sim_I:

$$sim_I(i, i') = \begin{cases} \cos(v_i, v_{i'}), & \text{если } v_i, v_{i'} \neq \bar{0} \\ 0, & \text{иначе} \end{cases}.$$

Чем ближе значение $sim_I(i, i') > 0$ к 1, тем больше альбомы i, i' похожи. Чем ближе значение $sim_I(i, i') < 0$ к -1, тем более они различны. Если значение $sim_I(i, i')$ близко к 0, то схожесть не определена. Если хотя бы один альбом не оценен ни одним пользователем, то считается, что $sim_I(i, i') = 0$. Далее, для каждого альбома $i \in I'_{albums}$ на основе значений sim_I определяется множество похожих альбомов:

$$Sim_I(i) = \{i' \in I'_{albums} \backslash \{i\} \mid sim_I(i, i') \geq t_{sim_I}\}, \qquad t_{sim_I} \in (0, 1],$$

где t_{sim_I} – пороговое значение.

Для артистов, жанров, связанных хотя бы с одним треком, – аналогично.

Также при построении вектора v_i для альбома i, если пользователь u не оценивал альбом, но оценивал хотя бы один связанный трек, предлагается вычислять среднее значение нормализованных оценок пользователя, поставленных связанным трекам, и в зависимости от того, достигает ли это значение t_r, восполнять компоненту $v_i(u) = 0$ значением из множества $\{-1, 1\}$. В результате восполнения нулевых компонент векторов v строятся векторы v', которые могут быть использованы вместо векторов v при вычислении значений sim_I. Аналогично для артистов, жанров (используются оценки связанных треков и альбомов).

Предварительный выбор треков

Для пользователя $u \in U$ выбираются треки, которые в дальнейшем могут быть включены в рекомендацию. В первую очередь согласно формулам

$$I^1_{albums}(u) = \{i \in I'_{albums} | r_2(u, i) = 1\},$$
$$I^0_{albums}(u) = \{i \in I'_{albums} | r_2(u, i) = 0\},$$
$$I^{None}_{albums}(u) = \{i \in I'_{albums} | r_2(u, i) = None\}$$

определяются множества альбомов, связанных хотя бы с одним треком, которые нравятся пользователю, не нравятся и которые он не оценивал ($I^1_{albums}(u)$, $I^0_{albums}(u)$, $I^{None}_{albums}(u)$ соответственно).

Для каждого альбома $i \in I^{None}_{albums}(u)$ определяется число похожих альбомов, которые нравятся пользователю и не нравятся, $c^1(u, i)$, $c^0(u, i)$ соответственно:

$$c^1(u, i) = \left\| I^1_{albums}(u) \cap Sim_I(i) \right\|,$$
$$c^0(u, i) = \left\| I^0_{albums}(u) \cap Sim_I(i) \right\|.$$

На основе этих чисел предсказывается бинарная оценка:

$$\widehat{r_2}(u, i) = \begin{cases} 1, & c^1(u, i) \geq c^0(u, i), c^1(u, i) > 0 \\ 0, & c^1(u, i) < c^0(u, i) \\ None, & c^1(u, i) = c^0(u, i) = 0 \end{cases}.$$

Если оба эти числа равны нулю, то оценка не предсказывается.

Для артистов, жанров, связанных хотя бы с одним треком, – аналогично. Для пользователя случайно выбираются $N_{preselection} > 0$ треков, связанных с альбомами, артистами, жанрами, которые (предположительно) нравятся ему (согласно r_2 или $\widehat{r_2}$).

Определение похожих пользователей

В первую очередь для каждого пользователя $u \in U$ строится вектор v_u:

$$v_u(i) = \begin{cases} -1, & r_2(u,i) = 0 \\ 1, & r_2(u,i) = 1 \\ 0, & r_2(u,i) = None \end{cases}, \; i \in I.$$

Компонента $v_u(i)$ соответствует объекту i и содержит значение из множества $\{-1, 1, 0\}$ в зависимости от того, оценивал ли пользователь объект, и если оценивал, то понравился ли объект ему. Далее, для каждой пары пользователей $u, u' \in U$ $(u \neq u')$ на основе векторов v_u и $v_{u'}$ вычисляется значение коэффициента схожести sim_U:

$$sim_U(u,u') = \begin{cases} \cos(v_u, v_{u'}), & \text{если } v_u, v_{u'} \neq \bar{0} \\ 0, & \text{иначе} \end{cases}.$$

Чем ближе значение $sim_U(u,u') > 0$ к 1, тем больше пользователи u, u' похожи. Чем ближе значение $sim_U(u,u') < 0$ к -1, тем больше они различны. Если значение $sim_U(u,u')$ близко к 0, то схожесть не определена. Если хотя бы один пользователь не оценивал ни один объект, то считается, что $sim_U(u,u') = 0$. Далее, для каждого пользователя $u \in U$ на основе значений sim_U определяется множество похожих пользователей $Sim_U(u)$. Для этого выбираются $N_{Sim_U} > 0$ пользователей с наибольшими положительными значениями sim_U.

Вычисление значений признаков

Для пары пользователь $u \in U$, трек $i \in I_{tracks}$ вычисляются значения признаков $f_{k=0\ldots6}(u,i) \in [0,1] \cup \{None\}$. В качестве признаков $f_{k=0\ldots6}(u,i)$ выбраны средние взвешенные значения нормализованных оценок, поставленных (похожими) пользователями из множества $Sim_U(u)$ объектам из множеств $I_{k=0\ldots6}(i)$ (табл. 1).

Таблица 1

Признаки и соответствующие множества объектов

Table 1

Features and Corresponding Sets of Items

Признак	Множество	Объекты множества
$f_0(u,i)$	$I_0(i)$	Трек i
$f_1(u,i)$	$I_1(i)$	Альбом, связанный с треком i
$f_2(u,i)$	$I_2(i)$	Треки, связанные с альбомом
$f_3(u,i)$	$I_3(i)$	Артист, связанный с треком i
$f_4(u,i)$	$I_4(i)$	Треки и альбомы, связанные с артистом
$f_5(u,i)$	$I_5(i)$	Жанры, связанные с треком i
$f_6(u,i)$	$I_6(i)$	Треки и альбомы, связанные с жанрами

Идея заключается в том, что если похожие пользователи не оценивали трек i (это вероятно из-за проблемы разреженности пользовательских оценок), то, возможно, они оценивали связанные альбом, артиста, жанры. В этом случае значение $f_0(u,i)$ не определено, но хотя бы

одно из значений $f_1(u,i)$, $f_3(u,i)$, $f_5(u,i)$ можно вычислить. Если похожие пользователи не оценивали и связанный альбом, то, возможно, они оценивали другие треки, связанные с ним. В этом случае значение $f_1(u,i)$ не определено, но значение $f_2(u,i)$ можно вычислить. Аналогично для связанных артиста, жанров. Более того, когда значение f_0 можно вычислить, использование значений $f_{k=1\ldots6}$ может способствовать увеличению эффективности предсказания оценок треков.

Множества объектов $I_{k=0\ldots6}(i)$ (см. табл. 1) определены согласно формулам

$$I_0(i) = \{i\},$$

$$I_1(i) = \begin{cases} \{c_{track,album}(i)\}, & c_{track,album}(i) \neq None \\ \emptyset, & иначе \end{cases},$$

$$I_2(i) = \bigcup_{i' \in I_1(i)} c_{album,tracks}(i'),$$

$$I_3(i) = \begin{cases} \{c_{track,artist}(i)\}, & c_{track,artist}(i) \neq None \\ \emptyset, & иначе \end{cases},$$

$$I_4(i) = \bigcup_{i' \in I_3(i)} \left(c_{artist,tracks}(i') \cup c_{artist,albums}(i')\right),$$

$$I_5(i) = c_{track,genres}(i),$$

$$I_6(i) = \bigcup_{i' \in I_5(i)} \left(c_{genre,tracks}(i') \cup c_{genre,albums}(i')\right),$$

Значения признаков $f_{k=0\ldots6}(u,i)$ вычисляются по формулам

$$R_{k=0\ldots6}(u,i) = \{(u',i',r') | u' \in Sim_U(u), i' \in I_k(i), r' = r(u',i') \neq None\},$$

$$f_{k=0\ldots6}(u,i) = \frac{\sum_{(u',i',r') \in R_k(u,i)} (sim(u,u') \times r')}{\sum_{(u',i',r') \in R_k(u,i)} sim(u,u')}, \; R_k(u,i) \neq \emptyset.$$

Если $R_k(u,i) = \emptyset$, то значение $f_k(u,i)$ не определено. В этом случае считается, что $f_k(u,i) = None$.

Обучение модели для предсказания пользовательских оценок треков

На этапах определения похожих пользователей и вычисления значений признаков подготавливаются тренировочное, валидационное (для выбора модели) и тестовое множества. Эти множества содержат кортежи: пользователь $u \in U$, трек $i \in I_{tracks}$, значения признаков $f_{k=0\ldots6}(u,i)$, оценка $r(u,i)$ (или $r_2(u,i)$). Важно, чтобы при подготовке этих множеств используемые оценки не пересекались. С использованием тренировочного множества обучается модель (модели) для предсказания оценок треков $\hat{r}$ (или $\widehat{r_2}$). В качестве входных данных модель использует предварительно обработанные значения $f_{k=0\ldots6}$. Обученная модель предсказывает оценки для предварительно выбранных треков.

Составление рекомендации

Для пользователя $u \in U$ составляется список треков, которые, предположительно, нравятся ему. Для этого из предварительно выбранных треков выбираются $N_{recommendation} > 0$ треков с наибольшими предсказанными нормализованными оценкам, достигающими t_r (в порядке убывания предсказанной нормализованной оценки). Если предсказываются бинарные оценки, то трек i включается в рекомендацию, если предсказанная бинарная оценка $\widehat{r_2}(u,i) = 1$.

Реализация МРС

Для реализации МРС выбран набор данных “C15 – Yahoo! Music user ratings of musical tracks, albums, artists and genres, v 1.0” (тренировочное подмножество “большого” поднабора данных, табл. 2). Пользователи и объекты (треки, альбомы, артисты, жанры) представлены в виде целочисленных идентификаторов. Для треков опционально определены связанные альбом, артист, жанры; для альбомов – связанные артист, жанры. Пользовательские оценки представлены в виде целочисленных значений из множества $\{0 \dots 100\}$, хотя 97,69 % оценок кратны 10.

Таблица 2

Основные характеристики используемой части набора данных

Table 2

Characteristics of the Dataset’s Applied Part

Показатель	Значение
Число пользователей	1000990
Число объектов	624961
Число треков	507172
Число альбомов	88909
Число артистов	27888
Число жанров	992
Число пользовательских оценок	252800275
Разреженность (согласно формуле (1))	99,96 %

Оценки (при помощи деления на 100) приведены к значениям из отрезка $[0, 1]$. Выбрано пороговое значение нормализованной оценки $t_r = 0{,}5$. Это наибольшее значение, которое достигается как минимум половиной нормализованных оценок (58,67 %, при $t_r = 0{,}51$ – 47,44 %).

Определение похожих альбомов, артистов, жанров и предварительный выбор треков

Недостатком набора данных является то, что не все альбомы, артисты, жанры связаны с треками. Поэтому на этапах определения похожих альбомов, артистов, жанров и предварительного выбора треков рассматривались 52 187 альбомов (58,7 % от числа всех альбомов), 19 691 артист (70,61 %), 205 жанров (20,67 %). На основе оценок пользователей из множества $\{0 \dots 19999\}$ определены множества похожих альбомов, артистов, жанров. Для этого использовались значения sim_I, вычисленные на основе векторов v, v', и пороговые значения t_{sim_I}: 0.01, 0.05, 0.1, 0.15, 0.2, 0.25 (12 подходов, для каждого подхода множества определены по отдельности). Для альбомов выбирается подход, при котором достигается наибольшая точность предсказания бинарных оценок. Для артистов, жанров – аналогично. Для пользователя с применением выбранных подходов предварительно выбираются $N_{preselection} = 40$ треков.

Определение похожих пользователей

Для каждого пользователя из множества $\{0 \dots 19999\}$ оценки треков разделены на тренировочные, валидационные и тестовые в долях 75, 12,5, 12,5 % соответственно. Все оценки альбомов, артистов, жанров использовались как тренировочные. Для построения векторов v

и вычисления значений sim_U использовались только тренировочные оценки. Для пользователей из множества $\{0 \dots 199\}$ хотя бы с одной оценкой трека (123 пользователя) вычислены значения sim_U относительно пользователей из множества $\{0 \dots 19999\}$. Для определения множеств похожих пользователей Sim_U выбирались $N_{Sim_U} = 1000$ пользователей с наибольшими положительными значениями sim_U. В результате для 119 из 123 пользователей определена 1 000 похожих пользователей (для остальных 4 пользователей – меньшее число). В среднем по 123 множествам Sim_U среднее значение sim_U составило 0,1481, а среднее квадратичное отклонение значений sim_U внутри множества – 0,03.

Вычисление значений признаков

Для оценок треков, поставленных пользователями из множества $\{0 \dots 199\}$, вычислены значения признаков $f_{k=0\dots6}$. В результате подготовлены тренировочное, валидационное и тестовое множества с мощностями 16769, 2797, 2787 соответственно (табл. 3).

Таблица 3

Доли пропусков значений признаков, %

Table 3

Features' Missing Parts, %

Множество / Признак	f_0	f_1	f_2	f_3	f_4	f_5	f_6
Тренировочное	32,55	20,6	17,87	13,71	13,36	4,94	2,17
Валидационное	33,25	20,38	17,73	14,48	13,55	4,83	2,32
Тестовое	32,15	19,27	17,19	13,17	12,67	5,06	1,94

В тестовом множестве значение признака f_0 (среднее взвешенное значение нормализованных оценок трека, поставленных похожими пользователями) не определено в 32,15 % случаев. Доли пропусков значений признаков $f_{k=1\dots6}$ значительно меньше. Отметим, что в используемой части набора данных 8,33 % треков не связаны ни с одним альбом, 12,52 % – ни с одним артистом, 6,15 % – ни с одним жанром (в коммерческих музыкальных каталогах эта проблема отсутствует).

Модель для предсказания пользовательских оценок треков и составление рекомендации

Для реализации модели выбран метод машинного обучения «случайный лес» [17]. Метод применен со значениями параметров «число деревьев»: 10, 20, 30, 40, 50, 60, 70; «минимальное число элементов в листе»: 1, 5, 10, 15, 20 (35 вариантов). Пропуски в значениях признаков $f_{k=0\dots6}$ заменяются средним значением признака в тренировочном множестве (для признаков $f_{k=0\dots6}$ эти значения равны 0,5747, 0,5593, 0,547, 0,6021, 0,5579, 0,6243, 0,573 соответственно). С использованием тренировочного множества обучены модели для предсказания нормализованных и бинарных пользовательских оценок треков. С использованием валидационного множества выбираются модели с наибольшей эффективностью предсказания оценок. С использованием выбранной (для нормализованных или бинарных оценок) модели выбираются $N_{recommendation} = 10$ треков для рекомендации.

Оценка эффективности реализованной МРС

Предварительный выбор треков

Для оценки эффективности предсказания бинарных оценок альбомов, артистов, жанров использовались оценки пользователей из множества {20000 ... 29999}.

На основе оценок пользователей из множества {0 ... 29999} выявлено, что в **90,55 %** случаев, когда пользователь оценивал как альбом, так и хотя бы один связанный трек, бинарная оценка альбома совпадает с приведенным к бинарному значению (с использованием порогового значения t_r) средним значением нормализованных оценок связанных треков. Для артистов, жанров совпадения выявлены в **87,09** и **65,89 %** случаев соответственно. Это означает, что если пользователю нравится альбом, артист или жанр, то с большой вероятностью (с меньшей для жанра) ему нравятся связанные треки. Таким образом, при высокой точности предсказания бинарных оценок альбомов, артистов, жанров предварительный выбор треков является эффективным.

Для каждого пользователя из множества {20000 ... 29999} с двумя и более оценками альбомов (4 652 пользователя) оценки альбомов случайно разделены пополам на оценки, которые предсказываются и которые используются для предсказания. Всего предсказывалось 223 639 оценок, значение 0 принимают 46,3 %, значение 1 – 53,7 %. Для каждого подхода получены точность по формуле

$$ACC = \frac{1}{\|T\|}\sum_{T}[\widehat{r_2} = r_2] \times 100\% \qquad (2)$$

и доля не предсказанных оценок (табл. 4).

Таблица 4

Результаты предсказания бинарных оценок альбомов
(все подходы)

Table 4

Albums Binary Ratings Prediction
(All Approaches)

t_{sim_I}	ACC, %		Доля $\widehat{r_2} = None$, %	
	v	v'	v	v'
0,01	78,51	78,62	0,85	0,47
0,05	79,57	**79,99**	1,61	1,33
0,1	76,96	76,61	6,47	7,25
0,15	66,1	63,39	20,55	24,16
0,2	49,49	44,15	41,32	48,13
0,25	34,91	28,17	59,27	67,47

Выбран подход с использованием векторов v' и значения $t_{sim_I} = 0{,}05$. По сравнению с аналогичным подходом на основе векторов v доля альбомов с хотя бы одним значением sim_I, достигающим t_{sim_I}, увеличилась с 93 до 95,86 % (для этих альбомов потенциально может быть предсказана бинарная оценка). Для выбранного подхода получены доли, в которых для бинарных оценок со значениями 0 и 1 предсказаны значения 0, 1 и $None$ (табл. 5).

Таблица 5

Результаты предсказания
бинарных оценок альбомов со значениями 0 и 1
(выбранный подход)

Table 5

Albums Binary Ratings Equal to 0 and 1 Prediction
(Selected Approach)

$r_2 \setminus \widehat{r_2}$	0, %	1, %	$None$, %
0	76,55	22,33	1,13
1	15,54	82,96	1,5

Для артистов – аналогично (табл. 6). Всего предсказывалось 359 378 оценок (9 831 пользователь), значение 0 принимают 39,39 %, значение 1 – 60,61 %.

Таблица 6

Результаты предсказания бинарных оценок артистов
(все подходы)

Table 6

Artists Binary Ratings Prediction
(All Approaches)

t_{sim_I}	ACC, %		Доля $\widehat{r_2} = None$, %	
	v	v'	v	v'
0,01	81,81	81,79	0,12	0,07
0,05	83,41	83,3	0,21	0,15
0,1	84,06	**84,15**	1,05	0,93
0,15	81,7	81,82	4,49	4,38
0,2	73,72	73,36	14,26	14,69
0,25	59,53	57,81	31,5	33,61

Выбран подход с использованием векторов v' и значения $t_{sim_I} = 0{,}1$. По сравнению с аналогичным подходом на основе векторов v доля артистов с хотя бы одним значением sim_I, достигающим t_{sim_I}, увеличилась с 91,13 до 94,57 %. Для выбранного подхода получены доли, в которых для бинарных оценок со значениями 0 и 1 предсказаны значения 0, 1 и $None$ (табл. 7).

Таблица 7

Результаты предсказания
бинарных оценок артистов со значениями 0 и 1
(выбранный подход)

Table 7

Artists Binary Ratings Equal to 0 and 1 Prediction
(Selected Approach)

$r_2 \setminus \widehat{r_2}$	0, %	1, %	$None$, %
0	75,46	23,42	1,12
1	9,39	89,8	0,81

Для жанров – аналогично (табл. 8). Всего предсказывалось 32 596 оценок (6 498 пользователей), среди которых значение 0 принимают 38,92 %, значение 1 – 61,08 %.

Таблица 8

Результаты предсказания бинарных оценок жанров
(все подходы)

Table 8

Genres Binary Ratings Prediction
(All Approaches)

t_{sim_I}	ACC, %		Доля $\widehat{r_2} = None$, %	
	v	v'	v	v'
0,01	84,35	83,91	0,01	0
0,05	84,95	84,17	0,05	0
0,1	85,89	84,85	0,23	0,05
0,15	**86,39**	86,09	0,85	0,52
0,2	86,09	86,03	1,93	1,94
0,25	84,62	84,9	4,08	3,82

Выбран подход на основе векторов v и значения $t_{sim_I} = 0{,}15$. Доля жанров с хотя бы одним значением sim_I, достигающим t_{sim_I}, составляет 100 %. Для выбранного подхода получены доли, в которых для бинарных оценок со значениями 0 и 1 предсказаны значения 0, 1 и $None$ (табл. 9).

Таблица 9

Результаты предсказания
бинарных оценок жанров со значениями 0 и 1
(выбранный подход)

Table 9

Genres Binary Ratings Equal to 0 and 1 Prediction
(Selected Approach)

$r_2 \setminus \widehat{r_2}$	0, %	1, %	$None$, %
0	86,02	13,53	0,44
1	12,26	86,63	1,11

Предсказание пользовательских оценок треков

На основе оценок пользователей из множества $\{0 \ldots 199\}$ с хотя бы одной оценкой трека (123 пользователя) и похожих пользователей из множества $\{0 \ldots 19999\}$ подготовлены валидационное и тестовое множества с мощностями 2 797, 2 787 соответственно (табл. 10).

Таблица 10

Доли оценок кратных 0,1 в валидационном и тестовом множествах, %

Table 10

Parts of Ratings that Are Multiples of 0.1 in the Validation and the Test Sets, %

Мн. $\setminus r$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Валид.	35	0,14	0,21	6,97	0,21	10,87	0,32	11,23	1,22	17,27	9,72
Тест.	33,55	0,43	0,07	8,07	0,14	10,8	0,47	10,44	1,69	16,22	11,12

Метод «случайный лес» применен со значениями параметров «число деревьев»: 10, 20, 30, 40, 50, 60, 70; «минимальное число элементов в листе»: 1, 5, 10, 15, 20. Для обученных моделей с использованием валидационного множества получены значения RMSE (табл. 11):

$$RMSE = \sqrt{\frac{1}{\|T\|}\sum_T (\hat{r} - r)^2}.$$

Таблица 11

RMSE предсказания нормализованных оценок треков из валидационного множества (все модели)

Table 11

Tracks Normalized Ratings Prediction RMSE for the Validation Set (All Models)

Ч. д.	Минимальное число элементов в листе				
	1	5	10	15	20
10	0,3133	0,3071	0,309	0,3107	0,3119
20	0,3065	0,3035	0,3048	0,3095	0,3106
30	0,3043	0,3028	0,3059	0,3083	0,311
40	0,3026	0,3014	0,3049	0,3084	0,3097
50	0,303	0,3018	0,3049	0,3075	0,3103
60	0,3036	0,30101	0,3048	0,3075	0,3099
70	0,3018	**0,30095**	0,3042	0,3079	0,3104

Выбрана модель, для которой метод «случайный лес» применен со значениями параметров «число деревьев» – 70, «минимальное число элементов в листе» – 5.

Также рассмотрен коллаборативный метод, в котором значение признака f_0 (среднее взвешенное значение нормализованных оценок трека, поставленных похожими пользователями) используется в качестве предсказания. В тех случаях, когда значение f_0 не определено (для тестового множества в 32,15 % случаев), используется среднее значение f_0 в тренировочном множестве. Для выбранной модели и коллаборативного метода с использованием тестового множества получены значения RMSE и MAE (табл. 12):

$$MAE = \frac{1}{\|T\|}\sum_T |\hat{r} - r|.$$

Таблица 12

Результаты предсказания нормализованных оценок треков из тестового множества

Table 12

Tracks Normalized Ratings Prediction for the Test Set

	RMSE	MAE
Выбранная модель	0,3037	0,2354
Зн. пр. f_0 или ср. зн.	0,3946	0,3177

Выбранная модель превосходит коллаборативный метод. Также значения RMSE и MAE получены для подмножества тестового множества, для которого значения f_0 определены (табл. 13).

Таблица 13

Результаты предсказания нормализованных оценок треков из подмножества тестового множества, для которого значения f_0 определены

Table 13

Tracks Normalized Ratings Prediction for the Part of the Test Set where f_0 Is Calculated

	RMSE	MAE
Выбранная модель	0,3093	0,2419
Зн. пр. f_0	0,374	0,2808

Даже в тех случаях, когда коллаборативный метод предсказывает нормализованную оценку (значение f_0 определено), выбранная модель предсказывает ее с меньшими значениями ошибок RMSE и MAE.

Для выбранной модели получены значения важности признаков $f_{k=0\ldots6}$ (табл. 14). Для этого для каждого дерева вычисляется среднее взвешенное значение уменьшения неопределенности (англ. impurity) в вершинах, где используется признак. В качестве весов используются доли элементов тренировочного множества, достигающих вершины. Важность признака – это среднее этих значений, вычисленных для каждого дерева. При обучении моделей в качестве меры неопределенности используется среднеквадратичная ошибка (MSE).

Таблица 14

Важность признаков для предсказания нормализованных оценок треков (выбранная модель)

Table 14

Features Importance for the Normalized Ratings Prediction (Selected Model)

f_0	f_1	f_2	f_3	f_4	f_5	f_6
0,0986	0,0655	0,1166	0,1168	**0,3437**	0,1106	**0,1481**

Наиболее важными являются признаки f_4 и f_6 (средние взвешенные значения нормализованных оценок, поставленных похожими пользователями трекам и альбомам артиста; трекам и альбомам жанров соответственно).

Аналогично обучены модели для предсказания бинарных оценок. Среди бинарных оценок из валидационного множества значение 0 принимают 43,15 %, значение 1 – 56,85 %. Среди бинарных оценок из тестового множества значение 0 принимают 43,09 %, значение 1 – 56,91 %. Для обученных моделей с использованием валидационного множества получены значения точности (согласно формуле (2), табл. 15).

Таблица 15

Точность предсказания бинарных оценок треков
из валидационного множества (все модели), %

Table 15

Tracks Binary Ratings Prediction Accuracy for the Validation Set
(All Models), %

Ч. д.	Минимальное число элементов в листе				
	1	5	10	15	20
10	76,87	76,83	76,69	76,94	76,08
20	77,55	77,51	77,08	76,51	75,94
30	77,4	77,51	77,26	77,05	76,22
40	77,51	77,69	77,08	76,62	76,44
50	78,01	77,69	77,08	76,73	76,51
60	77,94	77,65	77,01	76,76	76,44
70	**78,05**	77,87	77,37	77,15	76,51

Выбрана модель, для которой метод «случайный лес» применен со значениями параметров «число деревьев» – 70, «минимальное число элементов в листе» – 1.

Аналогично рассмотрен коллаборативный метод, в котором приведенное к бинарному значение признака f_0 используется в качестве предсказания. В тех случаях, когда значение f_0 не определено (для тестового множества в 32,15 % случаев), используется предопределенное значение бинарной оценки – 0 или 1. Для выбранной модели и коллаборативного метода с использованием тестового множества получены значения точности и TPR, TNR, PPV, NPV (табл. 16):

$$TPR = \frac{\sum_T[(r_2=1)\wedge(\widehat{r_2}=1)]}{\sum_T[r_2=1]},$$

$$TNR = \frac{\sum_T[(r_2=0)\wedge(\widehat{r_2}=0)]}{\sum_T[r_2=0]},$$

$$PPV = \frac{\sum_T[(r_2=1)\wedge(\widehat{r_2}=1)]}{\sum_T[\widehat{r_2}=1]},$$

$$NPV = \frac{\sum_T[(r_2=0)\wedge(\widehat{r_2}=0)]}{\sum_T[\widehat{r_2}=0]}.$$

Таблица 16

Результаты предсказания бинарных оценок треков из тестового множества

Table 16

Tracks Binary Ratings Prediction for the Test Set

	ACC, %	TPR	TNR	PPV	NPV
Выбранная модель	**77,04**	0,8045	0,7252	0,7945	0,7375
Зн. пр. f_0 или 0	64,08	0,8815	0,3231	0,6323	0,6736
Зн. пр. f_0 или 1	65,81	0,6141	0,7161	0,7407	0,5842

Выбранная модель превосходит коллаборативный метод. Также значения точности и TPR, TNR, PPV, NPV получены для подмножества тестового множества, для которого значения f_0 определены (табл. 17).

Таблица 17

Результаты предсказания бинарных оценок треков из подмножества тестового множества, для которого значения f_0 определены

Table 17

Tracks Binary Ratings Prediction for the Part of the Test Set where f_0 Is Calculated

	ACC, %	TPR	TNR	PPV	NPV
Выбранная модель	76,04	0,8124	0,6776	0,8007	0,6938
Зн. пр. f_0	72,03	0,8382	0,5322	0,7407	0,6736

Даже в тех случаях, когда коллаборативный метод предсказывает бинарную оценку, выбранная модель предсказывает ее с большей точностью и значениями TNR, PPV, NPV.

Для выбранной модели аналогично получены значения важности признаков $f_{k=0\ldots6}$ (при обучении моделей используется неопределенность Джини, табл. 18).

Таблица 18

Важность признаков для предсказания бинарных оценок треков (выбранная модель)

Table 18

Features Importance for the Binary Ratings Prediction (Selected Model)

f_0	f_1	f_2	f_3	f_4	f_5	f_6
0,1483	0,0971	**0,1564**	0,1132	**0,196**	0,1258	**0,1632**

Наиболее важными являются признаки f_4, f_6, f_2, f_0 (среднее взвешенное значение нормализованных оценок, поставленных похожими пользователями трекам и альбомам артиста; трекам и альбомам жанров; трекам альбома; треку соответственно).

Заключение

В статье предложен подход для решения проблемы разреженности пользовательских оценок в исследованиях МРС. Он заключается в использовании оценок связанных (согласно метаданным контента) объектов как похожих. На основе предложенного подхода разработана и реализована МРС. Выполнена оценка реализованной МРС.

Выявлено, что если пользователю нравится альбом, артист или жанр, то с большой вероятностью (в меньшей степени для жанра) ему нравятся и треки, связанные с ним. При этом получена точность предсказания бинарных пользовательских оценок альбомов, артистов, жанров: 79,99, 84,15 и 86,39 % соответственно. Это означает, что предварительный выбор треков, реализованный в МРС, является эффективным.

Метод предсказания пользовательских оценок треков, реализованный в МРС, превосходит коллаборативный метод (использующий оценки похожих пользователей, поставленные треку) как на всем тестовом множестве (в тех 32,15 % случаев, когда коллаборативный метод не может предсказать оценку, используется предопределенное значение), так и на его подмножестве, для которого коллаборативный метод может предсказать оценку. При этом наиболее важными являются признаки, значения которых вычислены на основе оценок похожих пользователей, поставленных объектам, связанным с артистом и жанрами трека. Это означает, что использование оценок связанных объектов в реализованной МРС действительно способствует решению проблемы разреженности пользовательских оценок.

Список литературы / References

1. **Ricci F., Rokach L., Shapira B.** Recommender Systems: Introduction and Challenges. In: Recommender Systems Handbook. Springer US, 2015, p. 1–34. DOI 10.1007/978-1-4899-7637-6_1
2. **Su J.-H., Chiu T.-W.** An Item-Based Music Recommender System Using Music Content Similarity. In: Intelligent Information and Database Systems. Springer, Berlin, Heidelberg, 2016, p. 179–190. DOI 10.1007/978-3-662-49390-8_17
3. **Cheng R., Tang B.** A Music Recommendation System Based on Acoustic Features and User Personalities. In: Trends and Applications in Knowledge Discovery and Data Mining. Springer International Publishing, 2016, p. 203–213. DOI 10.1007/978-3-319-42996-0_17
4. **Hanna P.** Considering Durations and Replays to Improve Music Recommender Systems. *EAI Endorsed Transactions on Self-Adaptive Systems*, 2018, vol. 0, no. 0, p. 156379. DOI 10.4108/eai.5-2-2018.156379
5. **Fernández-Tobías I., Braunhofer M., Elahi M., Ricci F., Cantador I.** Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, 2016, vol. 26, no. 2, p. 221–255. DOI 10.1007/s11257-016-9172-z
6. **Kordumova S., Kostadinovska I., Barbieri M., Pronk V., Korst J.** Personalized Implicit Learning in a Music Recommender System. In: User Modeling, Adaptation, and Personalization. Springer, Berlin, Heidelberg, 2010, p. 351–362. DOI 10.1007/978-3-642-13470-8_32
7. **Schedl M., Knees P., McFee B., Bogdanov D., Kaminskas M.** Music Recommender Systems. In: Recommender Systems Handbook. Springer, US, 2015, p. 453–492. DOI 10.1007/978-1-4899-7637-6_13
8. **Cheng Z., Shen J.** On Effective Location-Aware Music Recommendation. *ACM Transactions on Information Systems*, 2016, vol. 34, no. 2, p. 1–32. DOI 10.1145/2846092
9. **Knees P., Schedl M.** Semantic Labeling of Music. In: Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies. Springer, Berlin, Heidelberg, 2016, p. 85–104. DOI 10.1007/978-3-662-49722-7_4
10. **Schedl M., Zamani H., Chen C.-W., Deldjoo Y., Elahi M.** Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 2018, vol. 7, no. 2, p. 95–116. DOI 10.1007/s13735-018-0154-2
11. **Vall A., Dorfer M., Schedl M., Widmer G.** A Hybrid Approach to Music Playlist Continuation Based on Playlist-song Membership. In: Proc. of the 33rd Annual ACM Symposium on Applied Computing – SAC '18. ACM, 2018, p. 1374–1382. DOI 10.1145/3167132.3167280
12. **Knees P., Schedl M.** Basic Methods of Audio Signal Processing. In: Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies. Springer, Berlin, Heidelberg, 2016, p. 33–50. DOI 10.1007/978-3-662-49722-7_2
13. **Domingues M. A., Gouyon F., Jorge A. M., Leal J., Vinagre J., Lemos L., Sordo M.** Combining usage and content in an online recommendation system for music in the Long Tail. *International Journal of Multimedia Information Retrieval*, 2013, vol. 2, no. 1, p. 3–13. DOI 10.1007/s13735-012-0025-1
14. **Knees P., Schedl M.** Contextual Music Meta-data: Comparison and Sources. In: Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies. Springer, Berlin, Heidelberg, 2016, p. 107–132. DOI 10.1007/978-3-662-49722-7_5
15. **Knees P., Schedl M.** Listener-Centered Data Sources and Aspects: Traces of Music Interaction. In: Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies. Springer, Berlin, Heidelberg, 2016, p. 161–177. DOI 10.1007/978-3-662-49722-7_7
16. **McFee B., Barrington L., Lanckriet G.** Learning Content Similarity for Music Recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, 2012, vol. 20, no. 8, p. 2207–2218. DOI 10.1109/TASL.2012.2199109

17. **Breiman L.** Random Forests. *Machine Learning*, 2001, vol. 45, no. 1, p. 5–32. DOI 10.1023/A:1010933404324

Материал поступил в редколлегию
Received
20.05.2019

Сведения об авторе / Information about the Author

Менькин Александр Валерьевич, магистрант факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Alexander V. Menkin, Undergraduate Student, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

a.menkin@g.nsu.ru
ORCID 0000-0002-6364-962X

УДК 004.89
DOI 10.25205/1818-7900-2019-17-3-61-72

Разработка автоматизированных методов представления знаний о действиях и ситуациях

Е. О. Ненашева [1], **Д. Е. Пальчунов** [1, 2]

[1] *Новосибирский государственный университет*
Новосибирск, *Россия*

[2] *Институт математики им. С. Л. Соболева СО РАН*
Новосибирск, *Россия*

Аннотация

Статья посвящена разработке автоматизированных методов интеграции знаний, извлеченных из текстов естественного языка. Для решения этой задачи используются методы преобразования предложений естественного языка во фрагменты атомарных диаграмм. Знания, извлеченные из текстов, формализуются при помощи атомарных предложений сигнатуры, состоящей из символов констант, двухместных предикатов и дополнительных констант-ситуаций. Разработаны методы интеграции знаний, содержащихся в нескольких предложениях естественного языка, позволяющие учитывать их семантические контексты.

Ключевые слова

извлечение знаний, представление знаний, интеграция знаний, анализ текстов естественного языка, теория «Смысл ⇔ текст», формализованное представление ситуаций, атомарная диаграмма

Благодарности

Исследование выполнено при частичной финансовой поддержке Президиума СО РАН (проект «Инженерия интенсиональных онтологий в дедуктивных и информационных системах» Комплексной программы ФНИ СО РАН II.1)

Для цитирования

Ненашева Е. О., *Пальчунов Д. Е.* Разработка автоматизированных методов представления знаний о действиях и ситуациях // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 61–72. DOI 10.25205/1818-7900-2019-17-3-61-72

Semi-Automated Methods for Representing Knowledge of Actions and Situations

E. O. Nenasheva [1], **D. E. Palchunov** [1, 2]

[1] *Novosibirsk State University*
Novosibirsk, *Russian Federation*

[2] *Sobolev Institute of Mathematics SB RAS*
Novosibirsk, *Russian Federation*

Abstract

The article is devoted to the development of semi-automated methods for integrating knowledge extracted from natural language texts. To solve this problem, we use methods for converting natural language sentences into fragments of atomic diagram of algebraic systems. The knowledge extracted from texts is formalized with atomic sentences of signature consisting of constant symbols, binary predicates and additional situation-constants. We developed methods for integrating knowledge contained in several sentences of natural language, allowing to take their semantic contexts into account.

Keywords

knowledge extraction, representation of knowledge, knowledge integration, analysis of natural language texts, theory “Meaning ⇔ text”, formalized representation of situations, atomic diagram

Acknowledgements

The study was conducted with the partial financial support of the SB RAS Presidium (project "Engineering of intensional ontologies in deductive and information systems" of the Integrated program of fundamental research SB RAS II.1)

For citation

Nenasheva E. O., Palchunov D. E. Semi-Automated Methods for Representing Knowledge of Actions and Situations. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 61–72. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-61-72

Введение

Существует огромное количество текстовых документов, представленных в цифровом виде. С каждым днем их объем стремительно растет. Когда перед человеком встает задача обработки информации, заключенной в этих документах, справиться вручную уже практически невозможно. Таким образом, возникает потребность в программном обеспечении, способном обрабатывать тексты естественного языка, извлекать из них необходимую информацию и объединять полученные знания. При работе с документами, представленными на естественном языке, крайне важно учитывать смысл текста, определения и смысл входящих в него понятий, иначе говоря, семантику текста.

Проблема формального представления семантики текста исследовалась многими специалистами. Существенный вклад в решение этой проблемы внес И. А. Мельчук, разработав теорию «Смысл ⇔ Текст» [1; 2]. В рамках разработки этой теории был создан толково-комбинаторный словарь, позволяющий учитывать возможность слов вступать в семантические и синтаксические связи с остальными словами предложения. Кроме того, И. А. Мельчук предложил рассматривать глаголы в качестве многоместных предикатов.

Теория «Смысл ⇔ Текст» была применена в [3] при разработке теоретико-модельного подхода к извлечению знаний из текстов естественного языка. Подход был реализован в программной системе LogicText [4], позволяющей строить атомарные предложения логики предикатов по предложениям русского языка. Для построения фрагментов атомарных диаграмм моделей используются словари номинализаций и валентностей глаголов.

Дальнейшее развитие теоретико-модельного подхода предложено в [5], где описан подход к формализации знаний при помощи двухместных предикатов и констант-ситуаций, а также методы заполнения «пустых» валентностей предикатов-глаголов для выявления недостающих в тексте знаний и для пополнения этих знаний.

При анализе и обработке текстов естественного языка крайне важно учитывать контекст. Для корректного понимания семантики необходимо обрабатывать несколько предложений текста одновременно, поэтому целью данного исследования стала разработка методов интеграции знаний, заключенных в разных предложениях естественного языка. Особое внимание в данной работе уделяется выявлению связей между предложениями текста.

Преобразование текста естественного языка в атомарные предложения логики предикатов

Четырехуровневая модель представления знаний

При обработке, формализации и интеграции извлеченных из текста знаний мы используем четырехуровневую модель представления знаний, предложенную ранее [6; 7]. Данная модель включает в себя:

1) онтологию;
2) общие знания;
3) эмпирические (прецедентные) знания;
4) оценочные и вероятностные знания.

Онтология – это спецификация смысла, определения (полные или неполные) ключевых понятий, которыми описывается данная предметная область.

Общие знания – это утверждения, принципы и закономерности, являющиеся универсальными для всей предметной области. Они считаются истинными в каждом конкретном экземпляре, прецеденте данной предметной области (на текущий момент).

Эмпирические знания (иначе говоря, прецеденты) – это описания конкретных ситуаций, экземпляров предметной области.

Оценочные и вероятностные знания – это знания, истинность которых не является точной, это приближенные, нечеткие, гипотетические или субъективные знания.

В рамках четырехуровневой онтологической модели оценочные и вероятностные знания могут порождаться на основе онтологических, общих и эмпирических знаний о предметной области.

Одной из задач данного исследования является разработка методов формализации и интеграции знаний, соответствующих разным уровням описанной модели.

Представление знаний при помощи двухместных предикатов и констант-ситуаций

Введем некоторые определения. В работе рассматриваются модели вида

$$\mathfrak{A} = < A;\ \sigma > = < A;\ P_1, \dots, P_n\ , c_1, \dots, c_l >$$

сигнатуры $\sigma = <\ P_1, \dots, P_n, c_1, \dots, c_l >$, где A – универсум модели $\mathfrak{A}$, $P_1, \dots, P_n$ – предикатные символы, а $c_1, \dots, c_l$ – константные символы. Множество предложений сигнатуры σ обозначается как $S(\sigma)$.

Предложение φ называется атомарным, если

$$\varphi = (c_1 = \ c_2) \ \text{ или } \varphi = P(c_1, \dots, c_n), \text{ где } P, c_1, \dots, c_n \epsilon\ \sigma.$$

Множество предложений

$$AD(\mathfrak{A}) = \{\ \varphi \epsilon S(\sigma)\ \mid\ \mathfrak{A}_A \vDash \varphi \text{ } и \text{ } \varphi -$$

атомарное предложение или отрицание атомарного предложения}

мы называем атомарной диаграммой модели A.

Подмножество атомарной диаграммы $AD(\mathfrak{A})$ модели $\mathfrak{A}$ мы называем фрагментом атомарной диаграммы модели $\mathfrak{A}$, конечное подмножество – конечным фрагментом. В данной работе рассматриваются только конечные фрагменты атомарных диаграмм. Сведения по теоретико-модельному подходу к извлечению и представлению знаний можно найти в [3; 5; 8].

И. А. Мельчук предложил рассматривать глаголы в качестве многоместных предикатов. Основанием этому послужила способность глаголов вступать в связи с другими словами предложения. Это свойство было названо валентностью. Подход И. А. Мельчука был применен в [3] для извлечения и формализации знаний, содержащихся в предложениях естественного языка. В [3] разрабатывался теоретико-модельный подход к извлечению и формализации знаний: они представляются в виде наборов атомарных предложений логики предикатов. Глаголы, причастия и деепричастия представляются в виде n-местных предикатов. Разработанные методы были реализованы в программе Logic Text [4]. Она осуществляет построение фрагментов атомарных диаграмм моделей по предложениям естественного языка, используя словари номинализаций и валентностей глаголов. Пример работы программы представлен на рис. 1.

Одной из главных задач данного исследования является формализация знаний, извлеченных из текстов естественного языка, в виде двухместных предикатов. В первую очередь, подобный подход решает проблемы изменения валентности предикатов. n-местные предикаты, предложенные в [3] имеют фиксированный набор аргументов и не всегда полностью соответствуют составу слов в предложении, особенно когда речь идет об однородных членах предложения. С помощью двухместных предикатов мы можем обрабатывать слова, отвечающие на один и тот же вопрос. Кроме того, выбор двухместных предикатов в качестве ба-

зовой конструкции связан с тем, что для дальнейшей обработки знаний целесообразно использовать технологии семантической паутины (Semantic Web) [9–13].

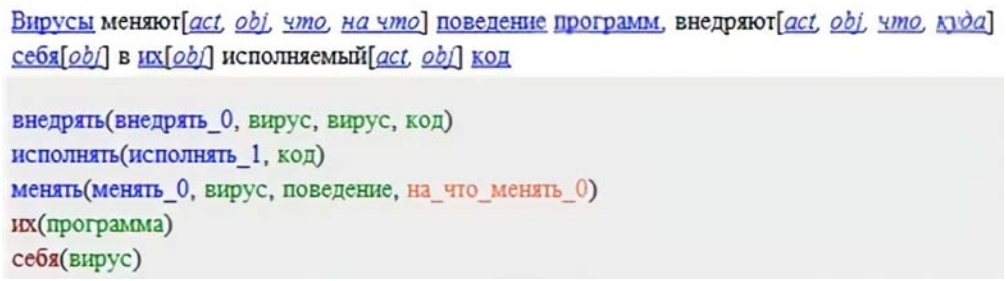


Рис. 1. Пример работы программы Logic Text
Fig. 1. The example of working with the program Logic Text

Одним из основных инструментов Semantic Web является язык описания онтологий OWL [14]. Он, в свою очередь, основан на модели представления данных RDF. RDF-утверждения имеют вид триплетов «субъект – предикат – объект».

Двухместные предикаты подходят под структуру триплетов, а значит, могут транслироваться в RDF-утверждения. Благодаря этому мы можем в дальнейшем использовать технологии Semantic Web, в частности применять ризонеры (автоматические средства логического вывода) [13; 15] [1] к наборам двухместных предикатов, построенных по текстам естественного языка. Это позволит нам выявлять противоречия, а также обрабатывать полученные знания и порождать новые.

Другим важным аспектом работы стало использование констант-ситуаций в качестве первого аргумента двухместного предиката. Как было сказано ранее, в данном исследовании используются различные уровни представления знаний. В текстах естественного языка объекты и сущности могут иметь различные смысл и свойства в зависимости от контекста и происходящей ситуации. Для корректной обработки текстов естественного языка необходимо учитывать, о каком объекте и о каких свойствах объекта в данном предложении идет речь. Таким образом, константы-ситуации могут быть применены для решения проблемы неполноты и неточности извлекаемой информации. Например, рассмотрим предложения «Вчера Вася купил хлеб за 30 рублей. Сегодня Вася купил хлеб за 35 рублей». Несмотря на внешнюю схожесть объектов (один и тот же глагол *купил*, объект *хлеб*), понятно, что эти предложения относятся к разным ситуациям, при этом Вася в обоих предложениях один, а хлеб разный. Такие моменты необходимо учитывать при обработке текстов естественного языка.

В данной работе мы продолжаем исследование проблемы формализации и интеграции знаний, извлеченных из текстов естественного языка, с применением двухместных предикатов и констант-ситуаций, начатое в [5].

Преобразование текста естественного языка в двухместные предикаты

Кратко изложим те аспекты подхода, предложенного в [5], которые потребуются нам в данной работе. Как уже было отмечено, в качестве базовой конструкции для построения модели знаний по тексту естественного языка нами выбраны двухместные предикаты и константы-ситуации. Для обработки текста на естественном языке и извлечения из него знаний на первом этапе производится преобразование текста с помощью программной системы LogicText. В результате мы получаем набор фрагментов атомарных диаграмм моделей, иначе говоря, набор многоместных предикатов, построенных по тексту.

[1] См. также: OWL 2 Web Ontology Language Document Overview (Second Edition). W3C OWL Working Group. URL: http://www.w3.org/TR/owl2-overview.

Например, предложение «Рецензент предоставляет в НГУ рецензию до 31 мая» программной системой LogicText будет преобразовано в предикат: *Предоставить* (*предоставить_0*, *рецензент*, *НГУ*, *рецензия*, *до 31 мая*). *Предоставить_0* является константой-действием.

Задача следующего этапа – преобразование *n*-местных предикатов в двухместные. В рамках Semantic Web известен подход, когда *n*-местный предикат преобразуется в набор из n двухместных предикатов [16]. Мы же осуществляем преобразование из каждого *n*-местного предиката в набор из $n + 1$ двухместного предиката. Перед преобразованием необходимо привести полученные на первом этапе наборы атомарных диаграмм к специальному виду, а затем заменить вспомогательные константы-действия на константы-ситуации. Более подробно преобразование предикатов описано в [5]. В результате мы имеем набор из $n + 1$ двухместного предиката.

Так, приведенный выше многоместный предикат преобразуется в следующий набор:

Что сделать (s, предоставить)
Кто (s, рецензент)
Куда (s, НГУ)
Что (s, рецензия)
Когда (s, до 31 мая)

Здесь s – новая константа-ситуация, введенная специально для формального представления данного предложения.

На третьем этапе необходимо определить связи между предложениями, точнее, между константами-ситуациями. Этот шаг является ключевым для понимания общего контекста. Важно определить, являются ли ситуации тождественными, т. е. идет ли в них речь об одном и том же действии. Кроме того, в одних случаях одна константа относится к одним и тем же объектам, а в других – к совершенно разным. Таким образом, на третьем этапе выявляются эквивалентные ситуации, а также тождественные объекты.

После прохождения всех трех этапов работы мы имеем набор двухместных предикатов, построенных по фрагменту текста естественного языка, а также знаем об эквивалентности некоторых констант-ситуаций. Теперь основная задача – объединить извлеченные знания. Рассмотрим данный этап подробнее.

Интеграция знаний, содержащихся в нескольких предложениях текста естественного языка

Предположим, у нас есть ситуации s_1, …, s_5 , которые являются эквивалентными. Эквивалентность ситуаций s_i и s_j задается с помощью отношения **тождественности** $Id(s_i, s_j)$. Для каждой из этих ситуаций описаны фрагменты атомарных диаграмм. В таком случае для интеграции знаний создается новая ситуация s_6. К ней добавляются все знания, которые соответствуют ситуациям s_1, …, s_5, при этом сами фрагменты диаграмм s_1, …, s_5 не изменяются. Такой подход позволяет нам объединять знания и при этом сохранять исходные ситуации и данные для дальнейшего отслеживания изменений и обработки ризонерами [12; 17]. В результате мы получаем новую ситуацию s_6, включающую в себя знания из нескольких предложений текста естественного языка.

Помимо отношения тождественности, между ситуациями может возникать связь **включения** $Include(S_i, S_j)$. Такое отношение мы используем, когда ситуация s_i является частью более общей ситуации s_j. В этом случае при объединении знаний идет уточнение информации.

Например, мы имеем два предложения: «Студент находится в новом корпусе НГУ» и «Новый корпус НГУ находится в Академгородке». Первое предложение имеет меньшую продолжительность по времени и общность, чем второе предложение. Значит, ситуация, связанная с первым предложением, включается в ситуацию, связанную со вторым предложением. В результате объединения знаний и логического вывода мы получим предложение «Студент находится в Академгородке».

Формально представить описанные выше связи между ситуациями можно следующим образом:

Аксиомы для связи Identity

$Id(s_1, s_2)$: ситуация s_1 тождественна ситуации s_2.

Аксиомы отношения эквивалентности:

$Id(s, s)$,

$\big(Id(s_1, s_2) \rightarrow Id(s_2, s_1)\big)$,

$\Big(\big(Id(s_1, s_2)\ \&\ Id(s_2, s_3)\big) \rightarrow Id(s_1, s_3)\Big)$.

Аксиома конгруэнции:

$\big(\big(Id(s_1, s_2)\ \&\ P(s_1, x)\big) \rightarrow P(s_2, x)\big)$.

Аксиомы для связи Include

$Include(s_1, s_2)$: $s_1 \subseteq s_2$, s_1 – частное, s_2 – общее.

Аксиомы частичного порядка:

$Include(s, s)$,

$\Big(\big(Include(s_1, s_2)\ \&\ Include(s_2, s_1)\big) \rightarrow Id(s_1, s_2)\Big)$,

$\Big(\big(Include(s_1, s_2)\ \&\ Include(s_2, s_3)\big) \rightarrow Include(s_1, s_3)\Big)$.

Аксиома наследования:

$\Big(\big(Include(s_1, s_2)\ \&\ P(s_1, x)\big) \rightarrow P(s_2, x)\Big)$.

В ходе исследований было выявлено, что введение констант-ситуаций, а также отношений тождественности и включения между ними недостаточно для полноценной и семантически верной интеграции знаний. Объекты могут менять свои свойства, в том числе и с течением времени. Нам важно учитывать, что какие-то события происходят одновременно, какие-то последовательно или параллельно. Во избежание противоречий нужно понимать, какое событие произошло раньше, а также какой «временной объем» оно имело (сколько длилось, с какими событиями пересеклось). Для каждого такого события необходимо описывать набор ситуаций. По этой причине нам необходимо ввести модель, в которую, помимо прочего, будут введены временные отношения между ситуациями.

Такая модель должна включать в себя: двухместные предикаты, первым аргументом которых является определенная ситуация, набор используемых ситуаций и отношения между ситуациями (например, «раньше», «позже», «часть по времени», «часть по протяженности» и т. д.)

Вероятно, для определения временных отношений между ситуациями потребуется обработка дат каким-либо внешним источником (например, чтобы выявить, какая ситуация произошла раньше). Для этого можно использовать оракулы – веб-сервисы, объекты из внешнего мира.

Отдельно стоит отметить, что при интеграции знаний мы рассматриваем множество тех ситуаций, которые относятся к определенному событию, обстоятельству. Объем (или протяженность) каждой ситуации включает протяженность рассматриваемого события.

Итак, для того чтобы учитывать связи между ситуациями относительно времени, введены дополнительные отношения.

Для определения связей между ситуациями, которые выполняются последовательно, мы используем двухместное отношение ***Раньше*** (или ***before***). $Before(s_1, s_2)$ – отношение между ситуациями, когда ситуация s_1 происходит раньше ситуации s_2. Мы можем понять, что ситуации происходят последовательно, если в предложении четко указано время или присутствуют специальные слова, указывающие на это, например: *перед* (этим), *до* (этого), *раньше*, *вчера* и подобные. Обратный случай можно обозначить двухместным отношением ***позже* (*after*).** $After(s_1, s_2)$ – отношение между ситуациями, когда ситуация s_1 происходит позже ситуации s_2.

Таким образом, $Before(s_1, s_2) \Leftrightarrow After(s_2, s_1)$.

Рассмотрим данную связь между ситуациями на примере. Возьмем предложения «Комплект документов собирается на факультете» и «После комплект документов хранится в деле Обучающегося». Обратим внимание, что сущности «комплект документов» и «дело Обучающегося» будут рассматриваться как целые константы, а не как отдельные слова. После выполнения первого шага описанного выше алгоритма и приведения многоместных предикатов к специальному виду получим следующее:

1) *Собираться(s_1, комплект документов, на факультете)*

2) *Храниться(s_2, комплект документов, в деле Обучающегося, после)*

Этому соответствуют следующие наборы двухместных предикатов:

1) *Что (s_1, комплект документов)*

Что делает(s_1, собирается)

Где (s_1, на факультете)

2) *Что (s_2, комплект документов)*

Что делает(s_2, хранится)

Где (s_2, в деле Обучающегося)

Когда(s_2, после)

Схематично это можно представить следующим образом (рис. 2).

Рис. 2. Схема разбора предложения
Fig. 2. The scheme of analysis of the sentence

Две полученных ситуации мы можем объединить согласно общему алгоритму интеграции. Его описание представлено ниже.

1. На первом шаге мы создаем общую ситуацию s, включающую в себя подситуации s_1 и s_2 (подситуаций может быть неограниченное количество). Другими словами, имеем отношения $Include(s_1, s)$ и $Include(s_2, s)$.

2. Проверяем тождественность ситуаций $Id(s_1, s_2)$ (для трех и более подситуаций делаем это последовательно, поскольку для отношения Id соблюдается транзитивность). Если удается выявить объекты, которые совпадают по смыслу, выносим их в общую ситуацию s (т. е. для этих объектов меняем текущую ситуацию на общую ситуацию).

3. Остальные объекты остаются в своих подситуациях, подситуации связываем соответствующим отношением.

4. Объединяем все полученные предикаты и отношения, строим по ним окончательное объединенное предложение.

Применим алгоритм к нашему примеру.

1. Создаем общую ситуацию s, включающую в себя ситуации s_1 и s_2, получаем связи между ситуациями $Include(s_1, s)$ и $Include(s_2, s)$. Схематично это можно представить следующим образом (рис. 3).

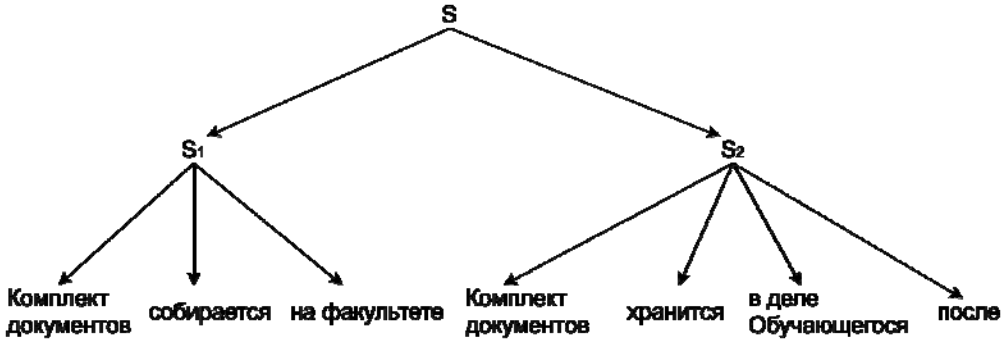


Рис. 3. Добавление общей ситуации
Fig. 3. The addition of the general situation

2. Проверяем тождественность s_1 и s_2 и выносим эквивалентные объекты в общую ситуацию s. В нашем случае такими объектами являются константы «комплект документов» и, соответственно, предикаты *Что(s_1, комплект документов)* и *Что (s_2, комплект документов)*. Выносим тождественные объекты в общую ситуацию s путем замены первого аргумента в соответствующих двухместных предикатах, получаем предикат *Что(s, комплект документов)*. Остальные предикаты остаются прежними:

Что делает(s_1, собирается)

Где (s_1, на факультете)

Что делает(s_2, хранится)

Где (s_2, в деле Обучающегося)

Когда(s_2, после)

Стоит отметить следующий момент: чтобы отнести тождественные объекты к общей ситуации, нужно их объединить в один. Если слова одинаковые, проблем при объединении не возникает. Но бывают случаи, когда встречаются слова, синонимичные, эквивалентные по смыслу или находящиеся в отношении «общее – частное». Например, если в одном предложении используется слово *студент*, а в другом – *бакалавр*. Сейчас для простоты будем приводить объекты к тому виду, в котором они встретились нам впервые. Если сначала мы рассмотрели предложение, где объектом был «студент», потом предложение, где объектом был «бакалавр», то, объединив эти слова, получим «студент» и вынесем его в общую ситуацию. В дальнейшем такие случаи нужно обрабатывать отдельно, возможно, в полуавтоматическом режиме (привлекая пользователя или эксперта).

Результат второго шага схематично представлен на рис. 4.

3. Далее необходимо связать подситуации s_1 и s_2 каким-либо отношением. Поскольку в предложении встречается ключевое слово *после*, будем связывать их временным отношением $After(s_2, s_1)$.

4. Выписываем все полученные предикаты и отношения:

$Include(s_1, s)$

$Include(s_2, s)$

Что(s, комплект документов)

Что делает(s_1 *, собирается)*

Где (s_1 *, на факультете)*

Что делает(s_2 *, хранится)*

Где (s_2 *, в деле Обучающегося)*

Когда(s_2 *, после)*

$After(s_2, s_1)$.

Для составления конечного предложения сначала выписываются общие объекты (из общей ситуации), потом к ним добавляются объекты из подситуаций. Поскольку подситуации связаны временным отношением, выписываем их в соответствующем порядке (рис. 5).

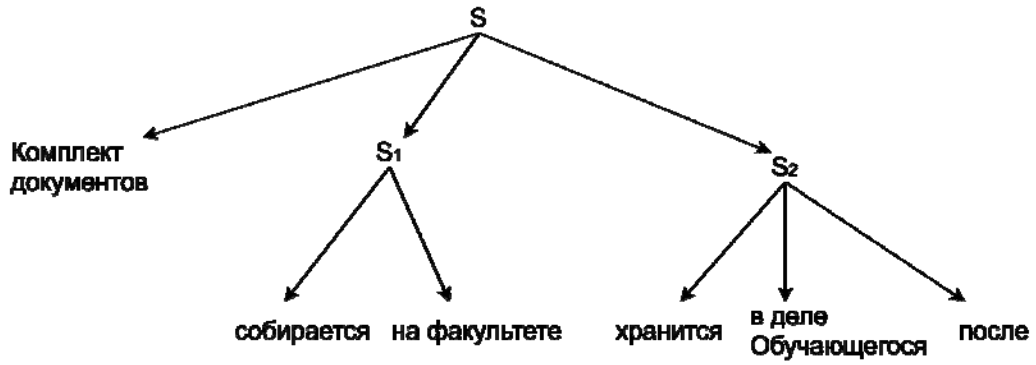


Рис. 4. Вынесение эквивалентных объектов в общую ситуацию
Fig. 4. Moving of the identical objects in the general situation

Рис. 5. Полученное предложение
Fig. 5. The resulting sentence

Мы рассмотрели пример интеграции знаний для ситуаций, которые происходят последовательно.

Для определения связей между ситуациями, которые происходят одновременно, предлагается использовать двухместное отношение ***Одновременно*** (или *SameTime*). В данном случае ключевыми словами могут быть «в то же время», «одновременно» и подобные. Если в предложениях явно не указано время действия или отсутствуют ключевые слова, применяется принцип пресуппозиции. Другими словами, сначала выявляется пресуппозиция, что две ситуации s_1 и s_2 связаны отношением $SameTime(s_1, s_2)$, затем задается вопрос пользователю о верности данной пресуппозиции. Если возможности общения с пользователем нет, формируется условное утверждение: фрагмент атомарной диаграммы верен, если верна данная пресуппозиция. В противном случае формируется другой фрагмент атомарной диаграммы, поскольку нам важно рассмотреть все возможные варианты интеграции знаний.

Для ситуаций, связанных отношением *SameTime*, объединение знаний происходит по описанному выше алгоритму. Так, из предложений «Для поступления поступающий подает заявление о приеме» и «Для поступления поступающий подает документ, удостоверяющий об-

разование соответствующего уровня» мы получим «Для поступления поступающий подает заявление о приеме и документ, удостоверяющий образование соответствующего уровня».

Добавление моментов времени во фрагменты атомарных диаграмм

Некоторые ситуации могут иметь условия, меняющиеся во времени, т. е. в момент времени t_1 утверждение является истинным, а в момент времени t_2, отличный от t_1, это же утверждение ложно. Поэтому кроме временных отношений между ситуациями следует вводить аргумент t – время. Если в утверждение добавлять момент времени, то оно становится «абсолютно» истинным или ложным, т. е. не зависящим от ситуации и времени.

В рамках данного подхода предложена конструкция, позволяющая добавить время в многоместный предикат и преобразовать его в набор двухместных.

Сначала многоместные предикаты вида $P(t, c_1, \dots, c_n)$, как это описано выше, преобразуются в набор атомарных предложений $\{R(s,P),\ Q_0(s,t), Q_1(s,c_1), \dots, Q_n(s,c_n)\}$, где s – соответствующая константа-ситуация. Выше были рассмотрены примеры предикатов $Q_i(s,c_i)$: $Кто(s,c_i)$, $Где(s,c_i)$, $Когда(s,c_i)$. Эти двуместные предикаты (*Кто*, *Где*, *Когда*) – типы аргументов предиката $P(t, c_1, \dots\ , c_n)$, они соответствуют вопросам к действию (ситуации), задаваемым предикатом $P(t, c_1, \dots, c_n)$.

Таким образом, при преобразовании многоместного предиката в набор двухместных происходит:

- добавление символов предикатов в качестве новых элементов и
- добавление констант s в качестве новых элементов.

Для того чтобы не добавлять имена предикатов в качестве новых элементов модели, мы рассматриваем также следующий способ преобразования многоместных предикатов в двухместные. Многоместный предикат вида $P(t, c_1, \dots, c_n)$ преобразуется в набор атомарных предложений $\{Q_0^P(s,t), Q_1^P(s,c_1), \dots, Q_n^P(s,c_n)\}$. Пример предиката $Q_i^P(s,c_i)$: $Купил\_Где(s,c_i)$. При таком способе преобразования многоместных предикатов в двухместные, происходит добавление только констант s в качестве новых элементов модели (и, соответственно, новых сигнатурных символов). Символы предикатов в качестве новых элементов модели не добавляются. В элементарную теорию полученной модели добавляются универсальные предложения: $\forall x_1\ \dots\ \forall x_n(P(t,x_1,\dots,x_n) \rightarrow (Q_0^P(s,t)\ \&\ Q_1^P(s,x_1)\ \&\ \dots\ \&\ Q_n^P(s,x_n)))$.

Заключение

В работе предложен подход к формализации и интеграции знаний, извлеченных из текстов естественного языка. В его основу легли теория «Смысл ⇔ Текст» И. А. Мельчука, а также теоретико-модельный подход к извлечению и представлению знаний, разработанный авторами ранее.

Разработаны методы интеграции знаний, содержащихся в нескольких предложениях естественного языка, основанные на использовании двухместных предикатов и констант-ситуаций. Предложенные методы могут быть применены для решения проблемы неполноты и неточности знаний. Также они дают возможность интегрировать как знания об объектах с неизменными свойствами, так и знания об изменяющихся во времени свойствах объектов.

В дальнейшем предложенный подход может быть дополнен использованием технологий Semantic Web, в том числе автоматических средств логического вывода, для выявления противоречий в документах естественного языка и для порождения новых знаний.

Список литературы

1. **Мельчук И. А.** Об одной лингвистической модели типа «Смысл – Текст»: уровни представления языковых высказываний // Серия литературы и языка. 1976. Т. 33, вып. 5. С. 5–33.
2. **Мельчук И. А.** Опыт теории лингвистических моделей «Смысл ⇔ Текст». М., 1999.

3. **Махасоева О. Г., Пальчунов Д. Е.** Автоматизированные методы построения атомарной диаграммы модели по тексту естественного языка // Вестник НГУ. Серия: Информационные технологии. 2014. Т. 12, № 2. С. 64–73.
4. **Махасоева О. Г., Пальчунов Д. Е.** Программная система построения атомарной диаграммы модели по тексту естественного языка. Св-во о гос. регистрации программы для ЭВМ № 2014619198, зарегистрировано 10.09.2014.
5. **Ненашева Е. О., Пальчунов Д. Е.** Разработка автоматизированных методов преобразования предложений естественного языка в бескванторные формулы логики предикатов // Вестник НГУ. Серия: Информационные технологии. 2017. Т. 15, № 3. С. 49–63.
6. **Найданов Ч. А., Пальчунов Д. Е., Сазонова П. А.** Теоретико-модельные методы интеграции знаний, извлеченных из медицинских документов // Вестник НГУ. Серия: Информационные технологии. 2015. Т. 13, № 3. С. 29–41.
7. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories // Сибирский научный медицинский журнал. 2016. Т. 36, вып. 1. С. 105–113.
8. **Пальчунов Д. Е.** Моделирование мышления и формализация рефлексии. Ч. 2: Онтологии и формализация понятий // Философия науки. 2008. № 2 (37). С. 62–99.
9. **Palchunov D. E.** Modeling of reasoning and formalization of reflection. I: Model theoretical formalization of ontology and reflection. *Philosophy of Science*, 2006, no. 4 (31), p. 86–114.
10. **Allemang D., Hendler J.** Semantic Web for the Working Ontologist. Morgan Kaufmann, 2008, 352 p.
11. **Parreiras F. S.** Semantic Web and Model-Driven Engineering. Wiley-IEEE Press, 2012, 264 p.
12. **Капустина А. И., Пальчунов Д. Е.** Разработка онтологической модели тарифов и услуг сотовой связи, основанной на логически полных определениях понятий // Вестник НГУ. Серия: Информационные технологии. 2017. Т. 15, № 2. С. 34–46.
13. **Корсун И. А., Пальчунов Д. Е.** Разработка интеллектуальной системы обработки и интеграции знаний на основе технологий семантической паутины // Вестник НГУ. Серия: Информационные технологии. 2018. Т. 16, № 3. С. 113–125.
14. **Szeredi P., Lukácsy G., Benkő T.** The Semantic Web Explained: The Technology and Mathematics behind Web 3.0. Cambridge University Press, 2014, 478 p.
15. **Meilicke C., Stuckenschmidt H.** A Reasoning-Based Support Tool for Ontology Mapping Evaluation. University of Mannheim, 2008.
16. **Gutiererz F., Dou D., Fickas S., Griffiths G.** Online Reasoning for Ontology-Based Error Detection in Text. Computer and Information Science Department University of Oregon, 2014.
17. **Noy N., Rector A.** Defining N-ary Relations on the Semantic Web. In: W3C Working Group Note (12 April 2006). URL: https://www.w3.org/TR/swbp-n-aryRelations/.

References

1. Melchuk I. A. About one linguistic model of the «Meaning ⇔ Text» type: levels of utterance representation. *Series of literature and language*, 1976, vol. 33, no. 5, p. 5–33. (in Russ.)
2. **Melchuk I. A.** Experience of the theory of the linguistic models “Meaning ⇔ Text”. Moscow, 1999. (in Russ.)
3. **Makhasoeva O. G., Palchunov D. E.** Semi-automatic methods of a construction of the atomic diagrams from natural language texts. *Vestnik NSU. Series: Information Technologies*, 2014, vol. 12, no. 2, p. 64–73. (in Russ.)
4. **Makhasoeva O. G., Palchunov D. E.** Program system for the construction of the atomic diagram of a model from natural language texts. Certificate of the State Registration of the Computer Program No. 2014619198, registered 10.09.2014.
5. **Nenasheva E. O., Palchunov D. E.** Semi-automated methods of transforming sentences from natural language into quantifier-free formulas of predicate logic. *Vestnik NSU. Series: Information Technologies*, 2017, vol. 15, no. 3, p. 49–63. (in Russ.)

6. **Naydanov Ch. A., Palchunov D. E., Sazonova P. A.** Model-theoretic methods of integration of knowledge extracted from medical documents. *Vestnik NSU. Series: Information Technologies*, 2015, vol. 13, no. 3, p. 29–41. (in Russ.)
7. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories. *The Siberian Scientific Medical Journal*, 2016, vol. 36, iss. 1, p. 105–113.
8. **Palchunov D. E.** Modeling of reasoning and formalization of reflection. Part 2: Ontologies and formalization of concepts. *Philosophy of Science*, 2008, no. 2 (37), p. 62–99. (in Russ.)
9. **Palchunov D. E.** Modeling of reasoning and formalization of reflection. I: Model theoretical formalization of ontology and reflection. *Philosophy of Science*, 2006, no. 4 (31), p. 86–114.
10. **Allemang D., Hendler J.** Semantic Web for the Working Ontologist. Morgan Kaufmann, 2008, 352 p.
11. **Parreiras F. S.** Semantic Web and Model-Driven Engineering. Wiley-IEEE Press, 2012, 264 p.
12. **Kapustina A. I., Palchunov D. E.** The development of ontological model of tariffs and services of mobile operator, based on logically complete definitions of concepts. *Vestnik NSU. Series: Information Technologies*, 2017, vol. 15, no. 2, p. 34–46. (in Russ.)
13. **Korsun I. A., Palchunov D. E.** An intellectual system for processing and integrating knowledge based on Semantic Web technologies. *Vestnik NSU. Series: Information Technologies*, 2018, vol. 16, no. 3, p. 113–125. (in Russ.)
14. **Szeredi P., Lukácsy G., Benkő T.** The Semantic Web Explained: The Technology and Mathematics behind Web 3.0. Cambridge University Press, 2014, 478 p.
15. **Meilicke C., Stuckenschmidt H.** A Reasoning-Based Support Tool for Ontology Mapping Evaluation. University of Mannheim, 2008.
16. **Gutiererz F., Dou D., Fickas S., Griffiths G.** Online Reasoning for Ontology-Based Error Detection in Text. Computer and Information Science Department University of Oregon, 2014.
17. **Noy N., Rector A.** Defining N-ary Relations on the Semantic Web. In: W3C Working Group Note (12 April 2006). URL: https://www.w3.org/TR/swbp-n-aryRelations/.

Материал поступил в редколлегию
Received
24.06.2019

Сведения об авторах / Information about the Authors

Ненашева Евгения Олеговна, студент, 2 курс магистратуры, факультет информационных технологий, Новосибирский государственный университет (ул. Пирогова, 2, Новосибирск, 630090, Россия)

Evgeniya O. Nenasheva, Student, 2 year master course, Department of Information Technology, Novosibirsk State University (2 Pirogov Str., Novosibirsk, 630090, Russian Federation)

nenasheva.zhenya@gmail.com

Пальчунов Дмитрий Евгеньевич, доктор физико-математических наук, заведующий кафедрой ОИ ФИТ, Новосибирский государственный университет (ул. Пирогова, 2, Новосибирск, 630090, Россия); ведущий научный сотрудник, Институт математики СО РАН (пр. Академика Коптюга, 4, Новосибирск, 630090, Россия)

Dmitry E. Palchunov, Doctor of Physical and Mathematical Sciences, Head of the Department of General Informatics of the Faculty of Information Technologies, Novosibirsk State University (2 Pirogov Str., Novosibirsk, 630090, Russian Federation); Leading Researcher, Sobolev Institute of Mathematics SB RAS (4 Academician Koptyug Ave., Novosibirsk, 630090, Russian Federation)

palch@math.nsc.ru

УДК 004.855.5
DOI 10.25205/1818-7900-2019-17-3-73-92

Разработка мобильного приложения визуализации технических характеристик устройств под ОС Android на базе технологий дополненной реальности и методов машинного обучения

И. С. Прокопьев [1], **Е. С. Шмаков** [2]

[1] *Новосибирский государственный университет*
Новосибирск, Россия

[2] *ООО «БНС»*
Новосибирск, Россия

Аннотация

В последнее время увеличивается количество окружающих нас электронных приборов, возрастает и их техническая сложность. В то же время инструкции к данным приборам читает только малая часть людей. Также возможны ситуации, когда инструкции или специалиста нет рядом с устройством. Поэтому часто встречаются случаи неправильного использования техники. В связи с этим закономерно встает задача быстрого распознавания типа устройства с последующим выводом информации о данном устройстве. В работе описывается процесс реализации системы распознавания типа устройства с использованием методов машинного обучения, с последующим выводом на экран его основных заранее отобранных характеристик и отображением пользователю набора вероятных инструкций для данного типа устройств.

Поскольку направлением данной работы является нахождение похожих объектов, а также их классификация на основе информации о внешнем представлении объекта, то рассматриваются алгоритмы получения признаков объекта из его изображения. В статье предлагается подход получения определяющих признаков объекта на основе его внешнего представления (контуров).

Ключевые слова

распознавание объектов, машинное обучение, дополненная реальность, контуры объекта, признаки изображения

Для цитирования

Прокопьев И. С., *Шмаков Е. С.* Разработка мобильного приложения визуализации технических характеристик устройств под ОС Android на базе технологий дополненной реальности и методов машинного обучения // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 73–92. DOI 10.25205/1818-7900-2019-17-3-73-92

Development Android Application for Visualization of Technical Characteristics Based on Augmented Reality and Machine Learning Methods

I. S. Prokopyev [1], **E. S. Shmakov** [2]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation

[2] *LLC "BNS"*
Novosibirsk, Russian Federation

Abstract

Nowadays the number of electronic devices has increased as well as their complexity. At the same time only few people are reading user guides for these devices. Moreover, in certain cases even no guides are provided by the manufacturer. Therefore there is a large number of cases when people use such devices incorrectly. This issue could be possi-

bly solved with a system automatically recognizing the type of the device. Such system could provide all necessary user information about that device. This article suggests one of possible implementations of such device type recognition system. It recognizes type of device in an unsupevised way and shows main characteristics and user guides for the recognized gadget.

Our approach for constructing such system relies on machine learning methods since greedy search for an object pattern is not efficient, as it was found out by recent scholarly works. Moreover, automatic object patterns classifiers show higher performance in this task and allow to scale the system to various kinds of input data. The algorithms that we are using for object classification are based on feature extraction from an graphical representation of the object look. This representation is usually proposed in an digital photography format. We consider our study as the first work towards automated defining characteristics of a device based on its graphical representation.

Keywords

object recognition, machine learning, augmented reality, object contours, image features

For citation

Prokopyev I. S., Shmakov E. S. Development Android Application for Visualization of Technical Characteristics Based on Augmented Reality and Machine Learning Methods. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 73–92. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-73-92

В последнее время увеличивается количество окружающих нас электронных приборов, возрастает и их техническая сложность. В то же время инструкции к данным приборам читает только малая часть людей. Возможны ситуации, когда инструкции или специалиста нет рядом с устройством, поэтому часто встречаются случаи неправильного использования техники. В связи с этим закономерно встает задача быстрого распознавания типа устройства с последующим выводом информации о данном устройстве. Но для реализации системы, хранящей информацию обо всех устройствах, требуется огромный объем данных, который обработать вручную не представляется возможным. Поиск объекта по шаблону в данной задаче является нерациональным, так как любой такой алгоритм составляется вручную с использованием перебора большого количества вариантов классификации объектов. Сложности возникают при распознавании объектов с разных ракурсов, при различном освещении и при деформации. Методы машинного обучения позволяют обучить модель с использованием различных видов входных данных, что дает высокий уровень распознавания объектов. Поэтому решением данной проблемы является использование технологий машинного обучения.

В работе описывается процесс реализации системы распознавания типа устройства, с последующим выводом на экран его основных заранее отобранных характеристик и отображением пользователю набора вероятных инструкций для данного типа устройств.

Поскольку направлением данной работы является нахождение похожих объектов, а также их классификация на основе информации о внешнем представлении объекта, то рассматриваются алгоритмы получения признаков объекта из его изображения. В статье предлагается подход к получению определяющих признаков объекта на основе его внешнего представления. При этом должна иметься возможность корректировки признаков, влияющих на результат распознавания объекта, и корректировка вклада отдельно взятых признаков.

Одним из наиболее популярных методов выделения признаков объекта на основе его внешнего представления является метод SURF. Этот метод основан на поиске ключевых точек изображения. Ключевая точка – это точка, имеющая определенные признаки, существенно отличающие ее от остальных точек изображения. Имея информацию о ключевых точках объекта, можно выполнять сравнение двух объектов или классифицировать объект, имея информацию о том, какой набор особых точек характерен для того или иного типа объектов. После получения ключевых точек изображения для каждой из них высчитывается ее дескриптор. Дескриптор – это вектор, характеризующий особую точку.

Данный алгоритм не подходит к решению задачи, поскольку, имея только ключевые точки (рис. 1), которые являются более характеристикой изображения, нежели объекта, нельзя предсказать признаки устройств. Поэтому в данной работе используется метод, основанный на использовании контурного анализа [1. С. 224].

Рис. 1. Пример работы алгоритма SURF
Fig. 1. SURF Algorithm Example

Контурный анализ представляет собой совокупность методов описания, преобразования, выделения контуров объекта на изображении. Контур в данном контексте – это граница объектов, т. е. совокупность точек (пикселей). Контуры изображения являются областями, которые хранят большое количество информации об объекте и не зависят при этом от уровня освещенности и яркости. Контурный анализ требует минимизацию количества шумов на изображении, поэтому до выделения контуров обычно применяются операции, которые уменьшают количество шумов (размытие, анализ гистограммы и т. д.). Одними из наиболее распространенных алгоритмов выделения контуров являются алгоритмы Adaptive Thresholding и Canny. Рассмотрим их подробнее.

Adaptive Thresholding часто применяют в задаче сегментации изображений. Подход является пороговым методом, при котором выбирается определенное значение яркости T, обычно полученное из анализа гистограммы изображения $f(x, y)$, затем каждый пиксель поочередно сравнивают с выбранным значением, и получившиеся два разбиения соответствуют либо принадлежности точки фону, либо объекту:

.

В алгоритме, использующем адаптивный порог, пороговое значение каждого пикселя зависит от соседних. Берется либо среднее значение окружающих пикселей, либо значение, в которое пиксели, лежащие ближе к исходному пикселю, дают больший вклад, а более удаленные – меньший. Затем для каждого пикселя из получившегося значения для области вычитают заранее заданную константу. Получившееся число является пороговым значением для области. Исходя из того, что данный алгоритм опирается на значения яркости изображения, итоговый результат становится чувствительным к перепадам яркости, теням и другим световым артефактам.

Алгоритм Canny состоит из пяти шагов.

1. Сглаживание. На данном этапе происходит фильтрация шума изображения с помощью фильтра Гаусса размером 5×5. Влияние пикселей при использовании фильтра Гаусса друг на друга обратно пропорционально квадрату расстояния между ними.

2. Нахождение градиентов для каждого пикселя изображения. На данном этапе для вычисления приближенного значения градиентов применяется оператор Собеля.

3. Подавление немаксимумов (Non-maximum suppression). Пикселями границ становятся точки, в которых в направлении вектора градиента наблюдается локальный максимум.

4. Выполнение двойной пороговой фильтрации. На данном шаге потенциальные границы определяются пороговыми значениями.

5. Трассировка области неоднозначности. Происходит подавление границ, не связанных с выделенными на шаге 4.

В результате анализа описанных подходов предложен алгоритм выделения признаков устройств на основе его контуров, а также их связи с техническими характеристиками устройств. Реализованный алгоритм является более гибким и «прозрачным», так как поддерживает настройку вклада каждого контура в получение его признаков и удаление ненужных контуров.

Поскольку не существует отдельного набора данных, подходящего под начальные условия задачи, было решено воспользоваться платформой “Open Image Dataset”. Вторым способом получения данных является использование поискового робота (так называемый скрапер) для обхода сайта производителя или поставщика техники. Полученные данные представляют собой таблицу в виде списка URL-адресов для загрузки данных, а также таблицу соответствий адресов классам изображений. Но в дальнейшем данные, полученные таким способом, не использовались, поскольку невозможно подобрать инструкции к моделям, представленным на изображениях. Поэтому в работе используется скрапер для обхода сайта – реселлера техники, который получает изображения техники вместе с его инструкцией в формате PDF.

Для того чтобы получить технические характеристики объекта, используется поисковый скрапер “Web Scraper”, который обходит сайт-реселлер, получая с каждой страницы товара нужную информацию. Для обхода сайта строится граф переходов между страницами, пагинацией сайта. Также задаются поля веб-страницы, откуда берутся параметры объекта, а также ссылка на его изображение (рис. 2). Результат обхода сайта сохраняется в таблице в формате CSV для последующего использования при кластеризации. Пример полученных данных показан на рис. 3.

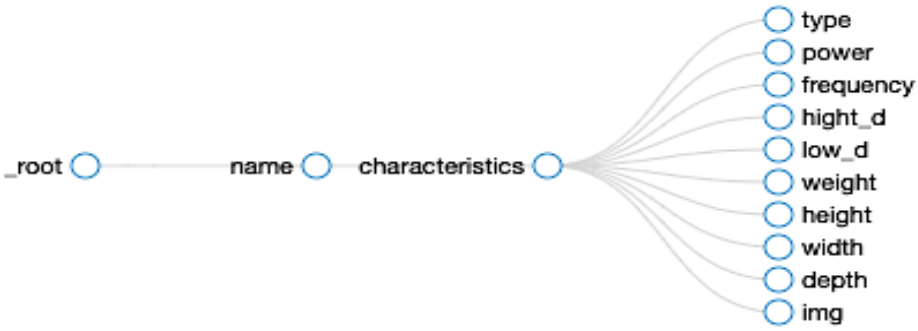


Рис. 2. Граф переходов по сайту

Fig. 2. Site’s Data Graph

name	type	power	power_m	freq	freq_m	hight_d	low_d	weight	height	width	depth
Полочные колонки Dali	полочные					21		2.6	237.0	140.0	195.0
Полочные колонки Vect	полочные	15	100	45	33000.0			9.5	310.0	170.0	220.0
Полочные колонки Wha	полочные	25	125	48	20000.0	25		9	355.0	221.0	290.0
Напольные колонки Auc	напольные		200	40	22000.0	25	120.0	27	980.0	140.0	250.0
Напольные колонки Poll	напольные	20	100	40	20000.0	25.4	165.0	19	920.0	235.0	260.0

Рис. 3. Результат выполнения скрапера

Fig. 3. Scrapper’s Search Result

Чтобы выделить контуры из исходного изображения, необходимы предварительные преобразования. Сначала изображение переводится из цветного в монохромное, соответственно значение каждого пикселя становится равным числу из множества [0, ..., 255]. Далее необходимо выделить объект на исходном изображении, но поскольку изображения взяты из каталога сайта-реселлера, объекты зачастую находятся по центру на белом фоне, без лишних объектов, поэтому задача распознавания объекта на изображении на данном этапе не рассматривается. На следующем шаге необходимо выделить границы объекта. Для решения задачи выделения границ объекта применяются алгоритмы Canny или Adaptive Thresholding (адаптивная пороговая обработка). Проанализируем их применимость к задаче.

В данной задаче перед алгоритмом Canny также было выполнено размытие по Гауссу с размером окна, равным 5. Данное значение обусловлено тем, что при бóльших значениях наблюдается снижение уровня шума, но также и потеря значимых контуров. Алгоритм Adaptive Thresholiding реализуется методом *adaptiveThreshold* из библиотеки OpenCV. Размер фильтра составляет 5 пикселей. Для того чтобы избавиться от шумов, перед выполнением алгоритма изображение обрабатывается с помощью медианного фильтра. Алгоритм его работы заключается в выборке среднего значения пикселей, попавших в окно фильтра на текущей итерации. Получившееся значение и будет выходным для рассматриваемого пикселя.

Чтобы сузить толщину линий контура и восстановить разрывы в контуре, на бинарном изображении применяются морфологические преобразования. В программной реализации, использующей алгоритм Canny, применяются операции “Erosion” и “Closing” [2. P. 137]. Erosion позволяет сузить контур. Принцип работы заключается в последовательном скольжении ядра по изображению: если все пиксели в окне равны единице, то и результирующий пиксель приравнивается к 1, иначе – к 0. Операция “Closing” является последовательным расширением и сужением до исходного состояния контура, что позволяет избавиться от пробелов внутри контура. Программно это реализуется функцией *morphologyEx* из библиотеки OpenCV. В реализации, использующей алгоритм Adaptive Thresholding, применяется операция “Opening”, которая помогает избавиться от мелких шумов на изображении, что, в свою очередь, минимизирует появление разрывных мелких контуров.

Поскольку сравнение двух объектов будет производиться в том числе и на основании иерархии контуров и их моментов, даже незначительные изменения положения на фотографии или уровня освещенности может снизить процент распознавания, а также повлиять на классификацию устройства. К тому же, поскольку внешний вид в пределах типа распознаваемого устройства может варьироваться, необходимо сохранить только значимые детали. Для решения этой задачи был написан метод, который выполняет аппроксимацию контура. Аппроксимация контура достигается путем удаления из контура компонент, находящихся на расстоянии меньше заданного. Также контуры, состоящие из количества точек меньше заданного, не включаются в получившийся список. Иерархия контуров при этом перестраивается с учетом удаленных контуров. Результат выполнения можно видеть на рис. 4, где справа находится объект с оптимизированным списком контуров, а слева – без оптимизации. На различных изображениях при использовании данного алгоритма количество контуров уменьшается в 2–5 раз, существенно не меняя представление объекта.

При использовании алгоритма Adaptive Thresholding наблюдается более сильное акцентирование мелких деталей, но также большие разрывы контуров (рис. 5). Границы объектов распознаются как отдельные контуры. С учетом этого в конечной реализации используется выделение контуров на основе алгоритма Canny.

Сравнение схожести объектов, полученных с фотографии пользователя, и исходной выборки, опирается на сравнение контуров. Для того чтобы определить положение контуров относительно друг друга, используется «иерархия контуров». Библиотека OpenCV имеет встроенный интерфейс для выделения контуров из предварительно обработанных изображений, реализующийся с помощью метода *cv2.findContours*(*input_image*, *mode*, *method*), где:

1) *input_image* – исходное черно-белое изображение;

2) *mode* – режим поиска контура. В данном случае используется извлечение контуров вместе с полной иерархией (RETR_TREE);

3) *method* – метод контурной аппроксимации. Используется метод, при котором каждый контур хранит только угловые точки (CHAIN_APPROX_SIMPLE). Использование такого метода обусловлено тем, что слишком большое количество точек может увеличить скорость сравнения и привести к снижению результатов.

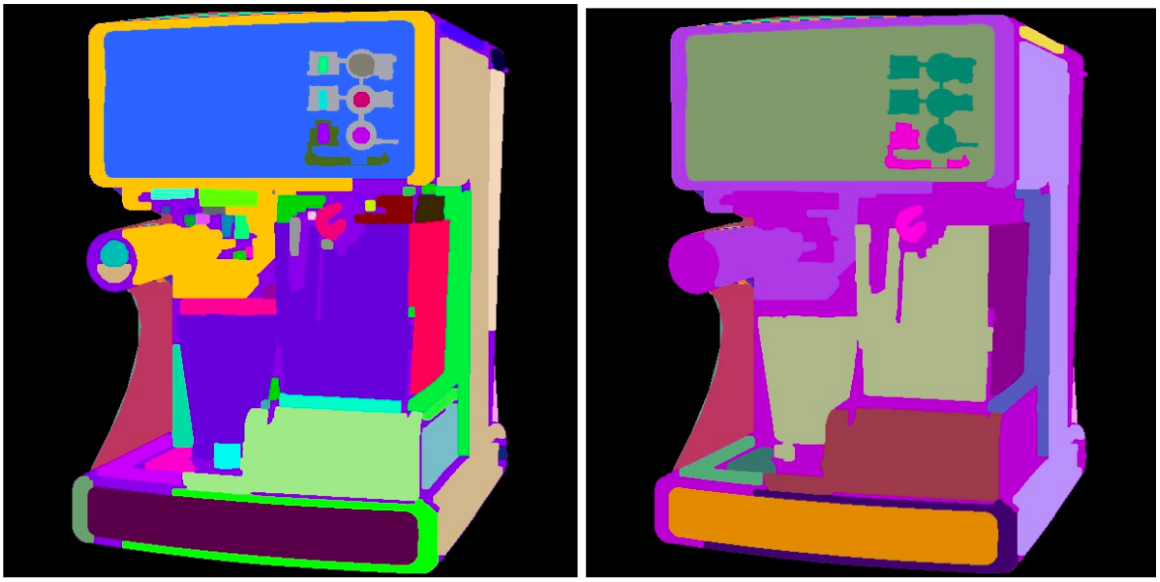


Рис. 4. Выделение контуров без применения (слева) и с применением (справа) оптимизации контуров

Fig. 4. Contours with (right) and without (left) Optimization

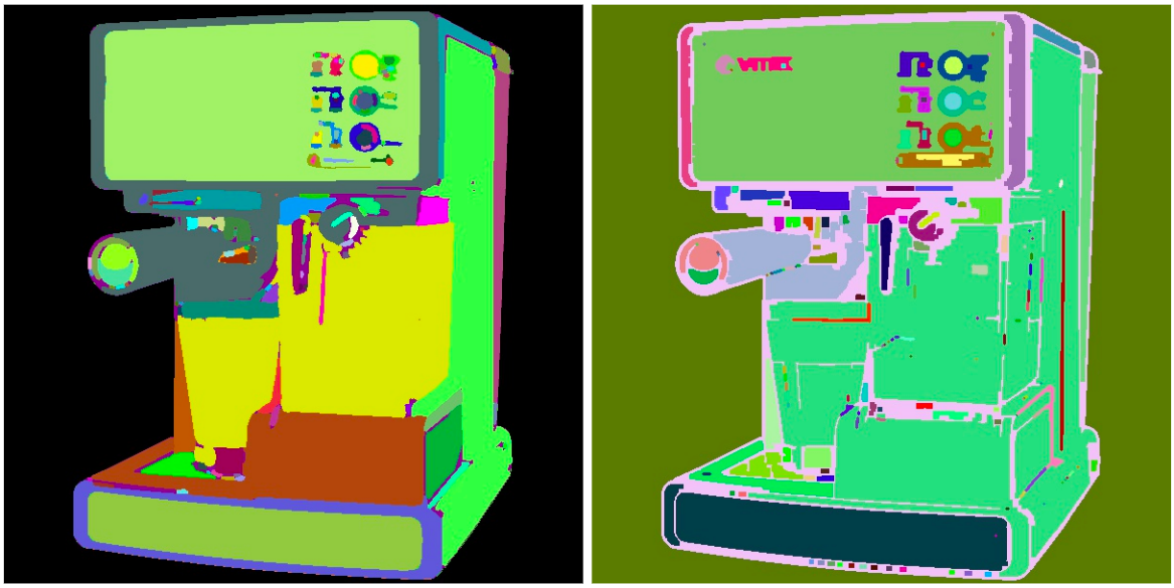


Рис. 5. Результат выделения границ методами Canny (справа) и Adaptive Thresholding (слева)

Fig. 5. Canny (right) and Adaptive Thresholding (left) Algorithms Results

Для каждого контура существует родительский контур и дочерние контуры. Структура контуров, которую предоставляет библиотека OpenCV, выглядит следующим образом:

$$\mathbf{K = [k1, k2, k3, k4]},$$

где

k1 – индекс следующего контура на текущем слое;
k2 – индекс предыдущего контура на текущем слое;
k3 – индекс первого дочернего контура;
k4 – индекс родительского контура.

Таким образом, иерархией контуров является граф (рис. 6).

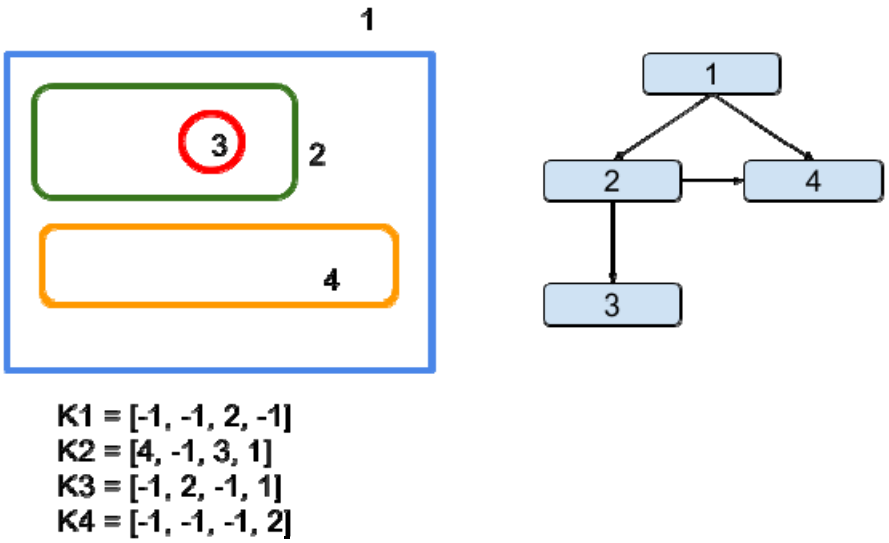


Рис. 6. Пример структуры контуров
Fig. 6. Contours Structure Example

Для оптимизации времени сравнения и для большей прозрачности самой операции сравнения, рассматривание контуров происходит от большего к меньшему, т. е. от родительского к дочернему. Соответственно при больших отклонениях мер родительских контуров операция завершается неудачей, и дочерние контуры сравниваться уже не будут. Чтобы сравнивать контуры между собой, необходимо выделить количественные коэффициенты контуров, а также задать сами правила сравнения. В условиях текущей задачи точное соответствие контуров при сравнении не требуется, поскольку тогда даже небольшие различия, возникающие из-за разных ракурсов, уровня освещения и большей предметности, например, моделей техники, может приводить к ухудшению качества распознавания.

Сравнение получившихся контуров происходит с помощью сравнения их моментов, что позволяет не учитывать их размер и угол поворота в пространстве. Момент – это численная характеристика контура, момент контура складывается из моментов его точек. Различают центральные, нормализованные и инвариантные моменты. Момент точки (x, y) порядка $(p + q)$ определяется как

,

где $I(x, y)$ – интенсивность пикселя с координатой (x, y); p – порядок x; q – порядок y. Порядок – степень, в которую возводят компоненты суммы. В случае с контурами момент, в кото-

ром $p = q = 0$, равен числу точек в контуре (поскольку рассматриваемое на данном шаге изображение бинарное, то интенсивность для точек контура равна 1).

Но моменты, вычисленные по такой формуле, зависят от положения в пространстве, поэтому используются центральные моменты, которые лишены этого недостатка, а также являются инвариантными относительно масштабирования. В вычислении центральных моментов применяются центроиды $(\bar{x}, \bar{y})$, где $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$. В таком случае формула принимает вид

$$\mu_{p,q} = \sum_x \sum_y I(x,y)(x-\bar{x})^p (y-\bar{y})^p.$$

Для независимости моментов от масштабирования моменты нормализуются:

$$\eta_{ij} = \mu_{i,j} / \mu_{00}^{(i+j)/2+1}.$$

Данный тип моментов не применяется при сравнении моментов, так как он не является инвариантным относительно поворотов. Поэтому для этих целей применяются Hu-моменты [3]. Инвариантный момент является линейной комбинацией центральных моментов. Hu-моменты – это набор из семи моментов, первые шесть инвариантны относительно сдвига, масштабирования и поворота, последний момент меняет знак при зеркальном отражении:

$$h_0 = \eta_{20} + \eta_{02},$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$

$$h_3 = (\eta_{30} + 3\eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$

$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2) + 3(\eta_{21} - \eta_{03}) \times \\ \times (3(\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2),$$

$$h_5 = (\eta_{21} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{12} + \eta_{03})^2 + (4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{12} + \eta_{03})),$$

$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2) + (\eta_{30} - 3\eta_{12}) \times \\ \times (\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{12} + \eta_{03})^2).$$

В работах [4; 5] приводится метод сравнения контуров на основе встроенных функций сравнения контуров (которые используют Hu-моменты) из библиотеки OpenCV, но неприменимость этого подхода в данной задаче обусловливается тем, что получаемый объект сравнивается не с одним объектом, а участвует в классификации, из чего следует необходимость хранения множества характеристик для каждого объекта выборки.

Hu-моменты вычисляются функцией *moment.HuMoments*() из библиотеки OpenCV. Для сравнения меры схожести моментов применяется несколько метрик. Зачастую применяется метрика вида $M(A,B) = \sum_{i=0}^{6} |H_i^B - H_i^A|$, где H – логарифм Hu-момента по основанию 10, взятый со знаком этого момента.

Использование данной метрики обусловлено также тем, что при составлении таблицы признаков для кластеризации необходимо хранить данные о каждом контуре, используя которые можно сравнивать контуры между собой. Таким образом, для каждого контура хранится алгебраическая сумма его моментов, имея которые и используя свойства суммы $\sum_{i=0}^{I}(a_i + b_i) = \sum_{i=0}^{I} a_i + \sum_{i=0}^{I} b_i$, можно вычислить меру сходства между контурами.

Поскольку категории, извлеченные с сайта, находятся в разном представлении (текстовая, численная информация), то необходимо нормализовать признаки, т. е. привести их к одному виду, в одну форму. Рассмотрим признаки, полученные с сайта, для выборки аудиоколонок:

1) тип колонок (напольные и т. д.);
2) материал корпуса (дерево, металл и т. д.).
3) мощность;
4) частотный диапазон;

5) габариты;
6) вес;
7) диаметр высокочастотных излучателей;
8) диаметр низкочастотных излучателей.

Выбор признаков обусловлен наличием связи между внешним видом устройства и его характеристиками. Например, имея данные о габаритах устройства и информацию о габаритах, полученную из анализа изображения, можно предсказать габариты объекта, полученного с камеры смартфона при распознавании.

Поскольку в полученной в результате обхода сайта таблице данные представления в столбцах в текстовой и числовой информации, необходимо привести данные в столбцах только к одному виду, опустив единицы изменения, предлоги, а также разбить столбцы, содержащие интервалы на два – минимальные и максимальные значения (рис. 7). Также в таблицу вносится информация о контурах, расположенная в порядке уменьшения периметра контуров.

Type	power	power_ma	freq	freq_m	hight_d	low_d	weight	height	width	depth	contour_0	contour_1
Полочные					21		2.6	237.0	140.0	195.0	16.663920881268623	-0.5297403
Полочные	15	100	45	33000.0			9.5	310.0	170.0	220.0	7.3507366452908975	55.7952324
Полочные	25	125	48	20000.0	25		9	355.0	221.0	290.0	-2.64724015162624	-0.8186475
Напольные		200	40	22000.0	25	120.0	27	980.0	140.0	250.0		-28.011966
Напольные	20	100	40	20000.0	25.4	165.0	19	920.0	235.0	260.0	-5.347270012066903	-3.8756198

Рис. 7. Данные после операции преобразования
Fig. 7. Data after Transformations

Поскольку в дальнейшем данные будут использоваться в алгоритме классификации, а в большинстве методов классификации используются евклидово или метрические пространства, то данные представляются в виде векторов. Метрикой является евклидова метрика ($\sqrt{m}$, где $m = \sum_{k=1}^{n}(p_k - q_k)^2$, $p = (p_1,...,p_n)$, $q = (q_1,...,q_n)$ – векторы размерностью n). Категориальные данные преобразуются в числовые. В таких случаях можно сопоставить с каждой категорией числовое представление, например: “catagoria_a” = 1, “catagoria_b” = 2 и т. д. Но проблема данного подхода заключается в том, что числовое представление категорий создает евклидовое представление. На нем становятся доступны операции, не имеющие смысла в категориальных признаках, такие как вычитание, умножение и т. д. Для того чтобы избежать этого, используется подход, в котором каждая категория заменяется на отдельный признак. На позиции, соответствующие численному значению, ставится 1, иначе – 0 (рис. 8).

contour_29	contour_30	type_floor	type_shelving
15.217872473771653	-14.894815648454125	0	1
13.692011951099424	3.790743824951145	0	1
27.824993381208337	27.77607981130654	1	0

Рис. 8. Результат выполнение OneHotEncoder
Fig. 8. OneHotEncoder Execution Result

В выборке встречаются недостающие данные, которые заменяются средним значением признака. Поскольку значения признаков могут сильно варьироваться – от пары десятков до нескольких десятков тысяч (например, в столбцах, отвечающих за моменты), разные признаки дают разный вклад при определении объекта. Данная проблема решается нормализацией значений признаков. В результате нормализации значение каждого признака принимает значение в интервале [–1, 1].

Поскольку для каждого класса (в условиях задачи классов техники) существует свое число внешних определяющих признаков (которое может не совпадать между классами), в качестве проверки работоспособности алгоритма решается задача бинарной классификации аудиоколонок.

Для того чтобы выделить признаки из контуров, необходимо их проанализировать. Для каждого класса коэффициенты, описанные ниже, могут отличаться и настраиваться исходя из выборки. Рассмотрим изображение из полученной выборки и его контуры (рис. 9).

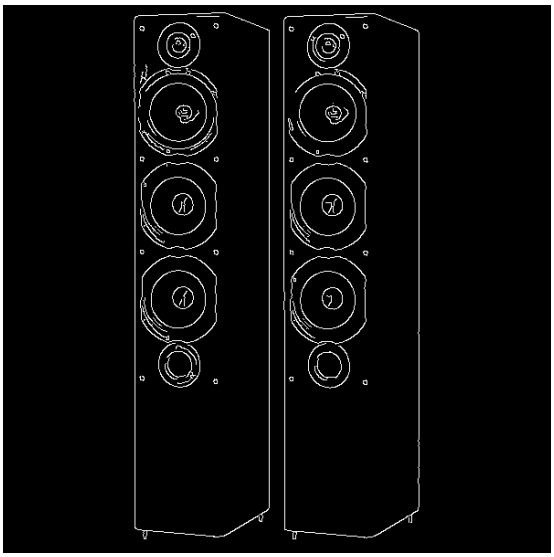


Рис. 9. Контуры объекта
Fig. 9. Object's Contours

На изображении присутствуют незначительные контуры либо помехи, которые при дальнейшей кластеризации и сравнении объектов могут вносить погрешности. Для того чтобы минимизировать эти эффекты, применяется фильтрация по периметру, а также по площади контура. Поскольку площадь объекта может меняться, данная фильтрация нужна прежде всего, чтобы явно выделить внешние грани объектов, а также большие ограниченные области внутри самих объектов. Рассмотрим распределение контуров по площади и периметру.

Общее количество контуров рассматриваемого объекта – 134. Преобладают контуры, площадь которых близка к 0 (рис. 10), такие контуры не являются определяющими для объекта и, скорее всего, являются шумом. Но поскольку это может быть площадь незамкнутых контуров, то также следует рассмотреть периметр контуров объекта.

Больше всего контуров, периметры которых близки к 0 (рис. 11). Также есть контуры, длина которых больше тысячи. Можно предположить, что это грани объекта и грани динамиков – определяющие признаки. В данном случае из набора контуров удаляются контуры, длина которых меньше 50 и площадь которых меньше 1 000. Данные числа получены экспериментально. На рис. 12 показаны оставшиеся контуры, по моментам которых будет происходить последующее сравнение объектов.

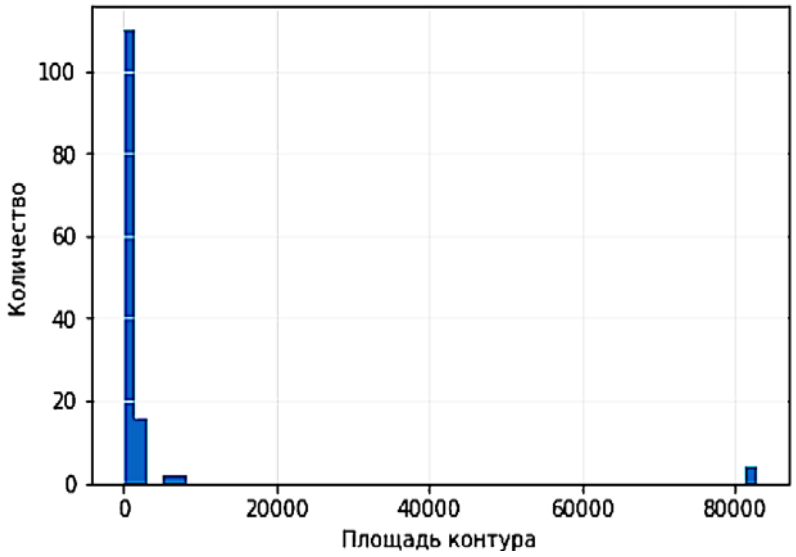


Рис. 10. Распределение контуров по площади
Fig. 10. Areas of Contours

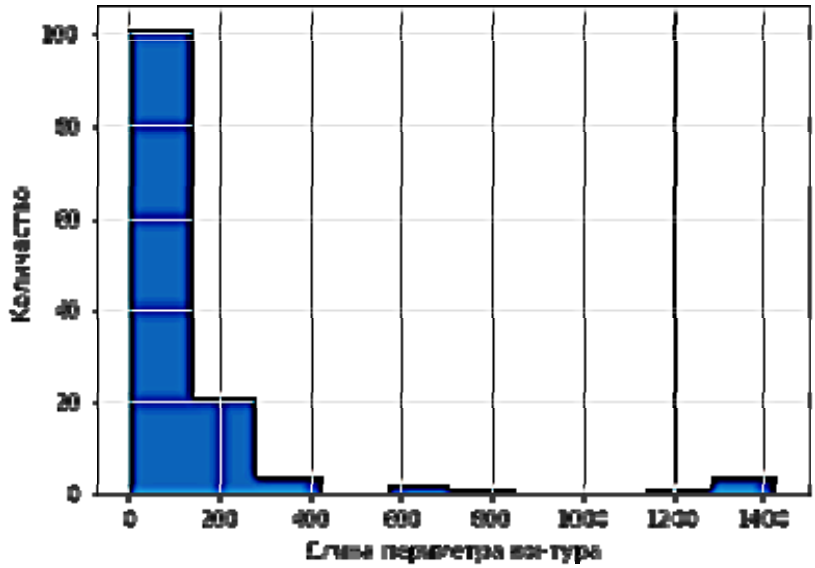


Рис. 11. Распределение контуров по периметру
Fig. 11. Perimeters of Contours

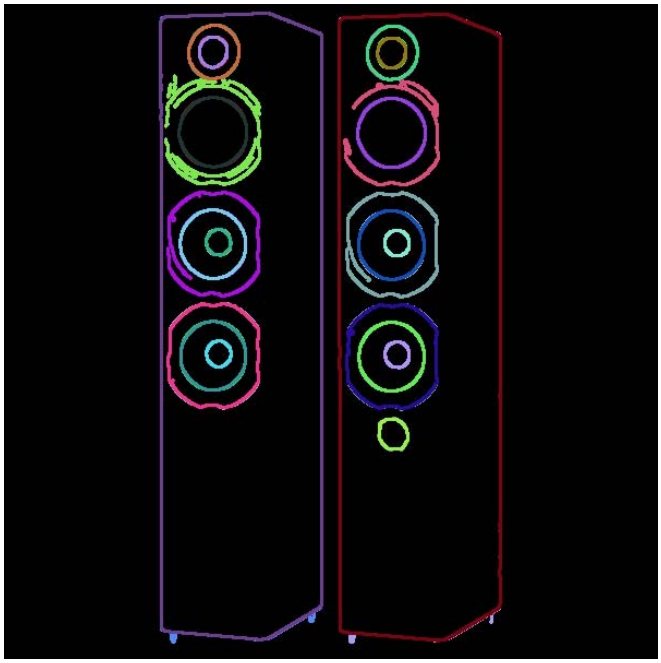


Рис. 12. Выделенные признаки изображения
Fig. 12. Object's Features

Примем гипотезу, что наиболее точно определяют объект его наибольшие компоненты. Для того чтобы построить таблицу признаков, выделим первые N контуров на основе их периметра, в таблицу признаков записываются соответствующие им моменты.

В качестве обоснования состоятельности подхода связи внешнего представления изображения с его характеристиками решается задача бинарной кластеризации. Набор данных включает в себя объекты двух типов – напольные и полочные аудиоколонки. В качестве алгоритма кластеризации используется алгоритм K-means (рис. 13). Принцип работы данного алгоритма заключается в перерасчете центров масс кластеров, изначально заданных произвольным образом, до тех пор, пока с каждой итерацией внутрикластерное расстояние не перестанет изменяться. В качестве метрики используется евклидово расстояние. Точность кластеризации составила 0,95.

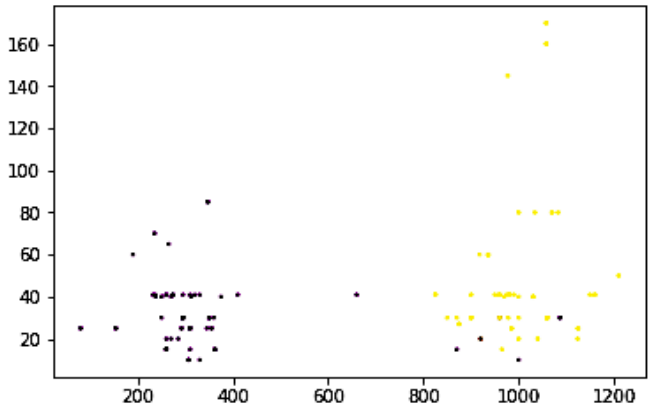


Рис. 13. Кластеризация с помощью метода k-средних
Fig. 13. Clustering with Using K-Means Algorithm

Алгоритм работы распознавания характеристик объекта состоит из следующих шагов:

1) по полученной выборке с отсутствующими данными строится таблица признаков;
2) таблица признаков нормализуется (все признаки приводятся к одному виду);
3) происходит заполнение недостающих данных исходя из значений данного признака для других объектов класса;
4) при получении объекта с неполными данными заполняются отсутствующие данные, и объект классифицируется.

Как уже было сказано, обучающая выборка состоит из объектов, принадлежащих двум классам – напольные и полочные аудиоколонки. Поскольку при фотографировании объекта модели на вход подается вектор, имеющий только часть, полученную из анализа изображения, требуется предсказать первую часть вектора, включающую в себя характеристики устройства. Вычисление всего вектора неизвестных признаков целиком одной моделью является нетривиальной задачей и может давать плохой результат, поэтому для каждого вычисляемого признака создается своя модель. Соответственно, количество моделей равно количеству неизвестных признаков. Результатом же выполнения будет класс – численная характеристика неизвестного параметра. Например, известно, что мощность устройства может быть одним значением из множества [50Вт, 100Вт, 300Вт]. В таком случае каждое значение является отдельным классом, и результатом классификации будет одно из этих значений. Такой подход также позволяет выбирать только определенные известные признаки для получения значения неизвестного признака и предугадывать значения, используя заранее вычисленные значения. В рассматриваемой выборке создается 10 моделей для получения неизвестных десяти характеристик. Но, поскольку данный подход является операцией над категориальными признаками, возможные значения которых должны быть заранее известны, он не подходит под признаки конечного вида, например габариты и т. д. Для того чтобы вычислить значения таких признаков, решается задача регрессии [6. С. 18]. Для решения данной задачи при классификации были использованы следующие алгоритмы:

1) дерево решений, точность 0,96;
2) *k* ближайших соседей, точность составила 0,79;
3) метод опорных векторов, точность 0,67.

Точность рассчитывается следующей формулой:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

В данном случае TP, TN – количество верно распознанных объектов первого или второго класса, FP, FN – количество неверно распознанных объектов первого или второго класса. Для того чтобы узнать точность предсказания каждого класса в отдельности, нужно использовать понятия полноты (recall) – доля объектов класса, из всех объектов класса, которую нашел алгоритм, и точности (precision) – доля объектов, которую алгоритм назвал положительными и которые действительно таковыми являются, поэтому используется *F*-метрика:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}.$$

F-метрика является средним гармоническим величин precision и recall, где $\beta = 1$. С использованием данной метрики распознавание напольных колонок составило 0,92, а полочных – 0,93.

При распознавании массы колонок выборка была разбита на классы до 10 кг, от 10 до 20 и выше 20 кг. Такое разбиение обосновывается на частоте встречаемости данных масс. При данном разбиении доля распознавания составила 0,74.

Как уже было сказано, для предсказания некатегориальных признаков (таких как длина и т. д.) используется регрессия. Поскольку данные в разных классах могут сильно отличаться, для предсказания параметра используются выборки, включающие только свой класс (в данном случае, например, отдельно класс напольных колонок и отдельно полочных). В качестве алгоритмов регрессии выбраны алгоритмы линейной регрессии, регрессия на ос-

нове случайного леса k ближайших соседей. Для вычисления каждого из неизвестных признаков из исходного списка признаков выбираются те, которые имеют зависимость от искомого признака. Например, при поиске габаритов такими данными являются те признаки, которые задают моменты первых контуров, вес, высота отдельных известных компонентов объекта.

После первого прогона алгоритмов результаты выглядят следующим образом:

1) линейная регрессия – 0,6;
2) случайный лес (150 деревьев) – 0,76;
3) k ближайших соседей (6 соседей) – 0,36.

В данной задаче в качестве метрики используется коэффициент детерминации (R-квадрат):

$$R^2 = 1 - \frac{V(y|x)}{V(y)} = 1 - \frac{\sigma^2}{\sigma_y^2}.$$

Данная метрика показывает, насколько условная дисперсия модели отличается от реальной дисперсии модели.

Поскольку второй алгоритм дал наиболее точный результат, модель, его реализующая, была выбрана в качестве основной для дальнейшего обучения. Также остальные алгоритмы при изменении параметров давали не сильно отличающиеся результаты.

После преобразований наилучший полученный результат стал равен 0,82. Данный результат достигнут на конфигурации, состоящей из 200 деревьев; количество выделяемых признаков, равное количеству выбранных заранее; максимальная глубина дерева = 20 (так как в данных возможны шумовые выбросы, что при большой глубине давало бы ненужные взаимосвязи).

Рассмотрим применение алгоритма к другим обучающим выборкам. Для этого была извлечена выборка, состоящая из обычных и двухстворчатых холодильников (рис. 14). Характеристиками объекта были выбраны:

1) тип;
2) вес;
3) количество камер (от 2 до 4);
4) ширина;
5) высота;
6) глубина.

Рис. 14. Пример данных выборки холодильников

Fig. 14. Samples of Fridges Data

В качестве пороговых значений для минимальной площади конура было выбрано значение $S = 1\,000$, для периметра – $P = 500$. Точность классификации объекта составила:

1) дерево решений, точность 0,94;
2) k ближайших соседей, точность 0,73;
3) метод опорных векторов, точность 0,68.

Результат распознавания количества камер 0,75. Для получения результата брались характеристики, отвечающие за ширину объекта, а также значения первых пяти контуров (данное количество получено опытным путем). Из полученных результатов видно, что алгоритм показывает хорошие результаты распознавания характеристик для различных выборок объектов. Качество зависит от анализа признаков контуров, таких как значения коэффициентов при выделении контуров, выбор минимальных значений для площади и периметра контуров.

Разработка системы

Алгоритм работы системы состоит из следующих шагов (рис. 15):

1) пользователь открывает приложение;
2) при фотографировании объекта изображение отсылается на сервер;
3) сервер классифицирует объект;
4) клиенту возвращается класс устройства, данные о нем и список возможных для загрузки инструкций;
5) на экране телефона показывается информация о найденном объекте с применением дополненной реальности;
6) при нажатии на список инструкций на сервер передается запрос, результатом которого является выбранная инструкция в формате PDF;
7) при получении мобильным приложением инструкции она отображается на экране смартфона.

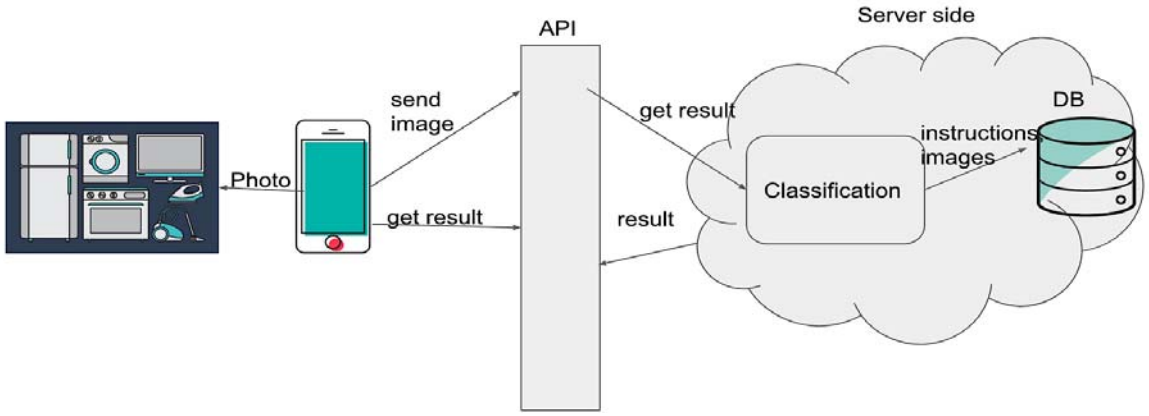


Рис. 15. Схема работы приложения
Fig. 15. Application Workflow Scheme

Так как распознавание непосредственно на устройстве является ресурсоемким, то было принято решение реализовать архитектуру «тонкий клиент», когда устройство ответственно только за получение и отправку данных. При старте приложения открывается экран, где показывается изображение с камеры устройства. При наведении смартфона на объект серверу передается текущее изображение. Для того чтобы не учитывать контуры фона, изображение обрезается (выделяется квадрат со стороной, равной трети исходного изображения). Затем клиентское приложение получает список вероятных характеристик устройства, а также иден-

тификаторы, присущие инструкциям устройств данного класса. При выборе одной из инструкций на сервер передается запрос, результатом которого является выбранная инструкция.

Мобильное приложение написано под OS “Android” на языке Kotlin. При написании приложения используются принципы ООП. Приложение спроектировано с использованием подхода “Clean Architecture” [7]. Этот подход является одним из популярных архитектурных решений на данный момент при проектировании мобильных приложений из-за простоты обучения, четкой интерпретации и большого количества обучающих источников. Также множество современных библиотек приспособлены для использования именно данного решения. Clean Architecture позволяет добиться таких важных моментов:

1) масштабируемость;
2) независимость от фреймворков;
3) хорошая тестируемость;
4) независимость от реализации пользовательского интерфейса (UI);
5) независимость от выбора базы данных.

Подход заключается в разбиении приложения по следующим уровням (рис. 16).

1. Слой представления. На данном слое логика связывается с интерфейсом приложения. Фрагменты и Активности, которые поставляет фреймворк Android, не имеют собственной логики, а ответственны только за отображение данных, полученных от презентера, – компоненты приложения, связывающей логику отображения информации с бизнес-логикой приложения, и передачи команд, таких как ввод текста пользователем или нажатие на кнопки презентеру. Презентеры на этом слое связываются с интеракторами из слоя бизнес-логики. Интерактор – это логика, разбитая на пользовательские сценарии, например: интерактор для авторизации на сервере, интерактор для отправления данных и т. д. Зачастую в этом месте происходит смена потока выполнения с интерфейсного на фоновый.

2. Слой бизнес-логики. Данный слой является модулем, лишенным зависимостей от фреймворка Android. На данном слое находится вся логика приложения. Логика разбита на интеракторы. Интеракторы связаны со слоем данных (Data-слой).

3. Слой данных. На этом слое находятся источники данных (база данных, кэш и т. д.), также на данном слое происходит соединение с сервером.

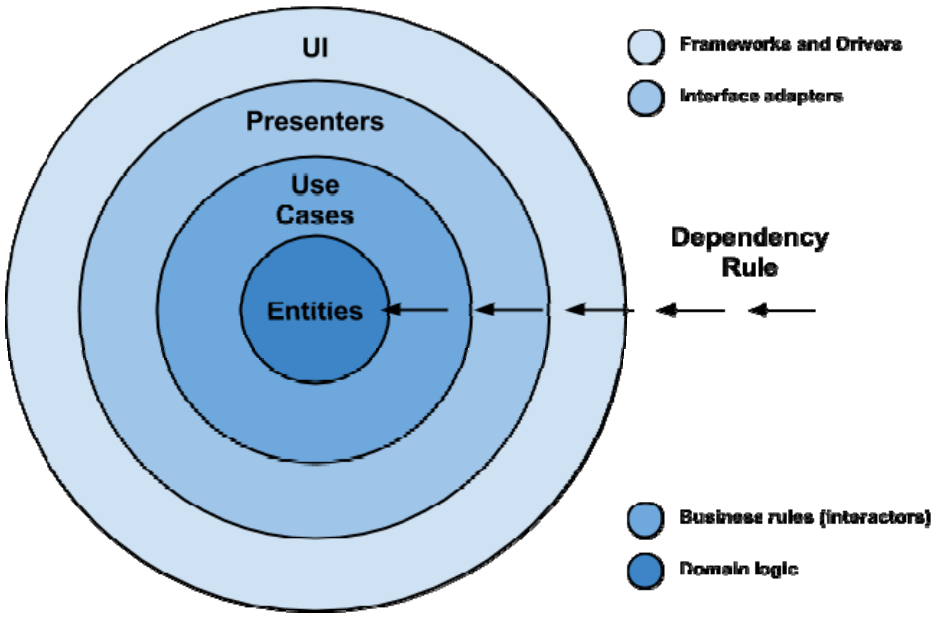


Рис. 16. Слои Clean Architecture
Fig. 16. Clean Architecture Layers

При включении приложения происходит регистрация клиента на сервере – клиентское приложение получает токен, по которому в дальнейшем происходит его аутентификация на сервере. Взаимодействие с сервером происходит по технологии REST, для соединения с сервером используется библиотека Retrofit. На главном экране приложения отображается полученное с камеры устройства изображение. При наведении камеры на распознаваемый объект полученное изображение отсылается на сервер для дальнейшей классификации. После того как сервер вычислил значение признаков для классифицируемого объекта, клиентское приложение извлекает их из полученного файла в формате JSON и отображает на экране устройства. Также во всплывающем окне появляется список инструкций, связанный с данным кластером устройств. При выборе нужной инструкции на сервер посылается GET-запрос с ее идентификатором. Результатом выполнения запроса является инструкция в формате PDF, которая по окончании загрузки сохраняется в память устройства и отображается на экране. Для отображения информации на экране используется библиотека ARCore.

Когда фреймворк ARCore распознал поверхность (вертикальную или горизонтальную), через заранее заданный промежуток времени выполняется отправка на сервер изображения с камеры. Если серверное приложение распознало один из типов устройств, то клиентское приложение получает информацию об этом устройстве, которая показывается поверх распознанного устройства (рис. 17) в виде списка.

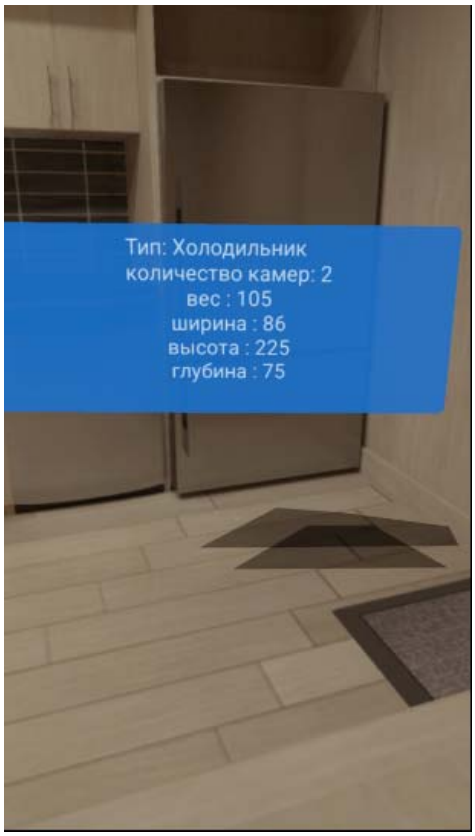


Рис. 17. Отображение характеристик устройства на примере холодильника

Fig. 17. Displaying of Fridge Characteristics

Сервер хранит заранее обученную модель, на вход которой при запросе передается информация, полученная из анализа переданного клиентом изображения. Поскольку полученное изображение цветное, то сначала оно переводится в черно-белое, затем из него выделя-

ются контуры и их моменты по алгоритму Canny, описанному ранее. После успешной классификации и вычисления характеристик входящего устройства в базе данных инструкций ищутся все ассоциированные с данным классом устройств инструкции. Затем идентификаторы инструкций передаются клиентскому приложению. При получении запроса от клиента на скачивание определенной инструкции в базе данных ищется искомая инструкция по идентификатору и отсылается клиенту. Серверная компонента приложения связывается с приложением-клиентом с помощью архитектурного подхода REST API. Запрос, реализующий данный подход, выглядит следующим образом:

1) маршрут отправки (адрес, по которому направляется запрос);
2) тип метода (GET, POST, PUT);
3) заголовки запроса;
4) тело запроса (данные).

Метод GET используется для получения с сервера определенных данных, результат на запрос данных отправляется приложению-клиенту (рис. 18).

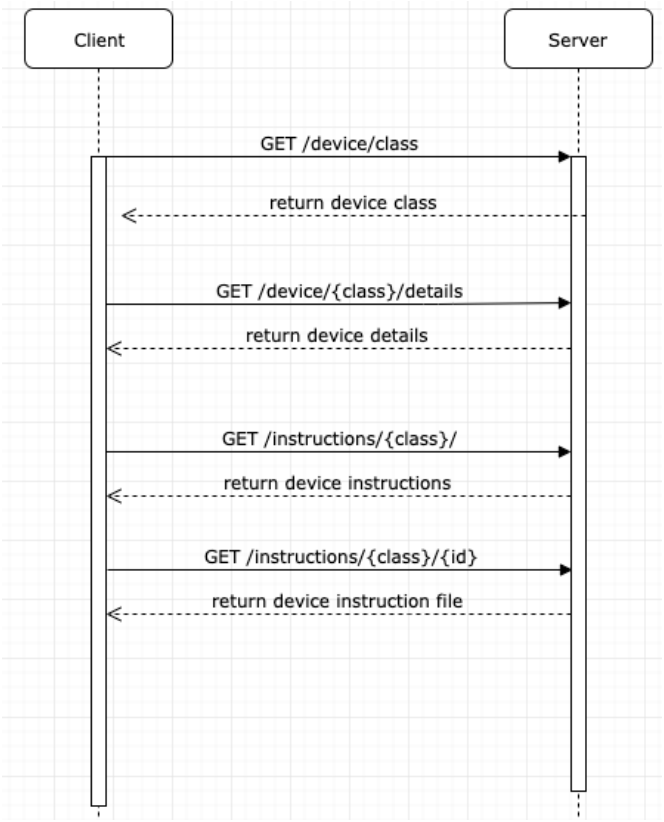


Рис. 18. Схема взаимодействия клиента и сервера
Fig. 18. Client-Server Connection Scheme

Запрос **GET /device/class/** отвечает за вычисление класса устройства, изображение которого присылает клиент в теле запроса. Результатом выполнения запроса является класс девайса. Запрос **GET /device/{class}/details** возвращает характеристики устройства. Запрос **GET /instructions/{class}/** запускает поиск инструкций (краткой инструкции и списка доступных для загрузки инструкций) для конкретного типа устройств. Загрузка определенной инструкции производится по запросу **GET /instructions/{class}/id**.

Заключение

Разработанный подход получения признаков объекта показал хорошую точность на исходной выборке. Данный подход позволяет связывать отдельные части объекта (ограниченные контуром) с каким-либо набором его признаков, что позволяет рассматривать изображение объекта как набор составляющих его частей, что в дальнейшем можно использовать задаче кластеризации объектов, зная характеристики отдельных компонент объектов.

Список литературы

1. **Фурман Я. А., Кревецкий А. В., Передреев А. К., Роженцов А. А., Хафизов Р. Г., Егошина И. Л., Леухин А. Н.** Введение в контурный анализ и его приложение к обработке изображений и сигналов. М.: Физматлит, 2002. 592 с.
2. **Garcia G. B., Suarez O. D., Aranda J. L. E.** Learning Image Processing with OpenCV. Birmingham, Packt Publ., 2015, 319 p.
3. **Ming-Kuei Hu**. Visual Pattern Recognition by Moments Invariants. *IRE Transactions on Information Theory*, 1962, vol. 8, no. 2, p. 179–187.
4. **Belongie S., Malik J., Puzicha J.** Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, vol. 24, no. 4, p. 509–522. DOI 10.1109/34.993558
5. **Коробов Д. В., Патин М. В.** Метод распознавания шрифта текста с изображения // Молодой ученый. 2016. № 12. С. 161–165.
6. **Вапник В. Н.** Восстановление зависимостей по эмпирическим данным. М.: Наука, 1979. 448 с.
7. **Robert C. Martin.** The Clean Architecture. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 15.02.2019)

References

1. **Furman Ya. A., Krevetskiy A. V., Peredreev A. K., Rozhentsov A. A., Khafizov R. G., Egoshina I. L., Leukhin A. N.** Vvedenie v konturnyj analiz i ego prilozhenie k obrabotke izobrazhenij i signalov. Moscow, Fizmatlit Publ., 2002, 492 p. (in Russ.)
2. **Garcia G. B., Suarez O. D., Aranda J. L. E.** Learning Image Processing with OpenCV. Birmingham, Packt Publ., 2015, 319 p.
3. **Ming-Kuei Hu**. Visual Pattern Recognition by Moments Invariants. *IRE Transactions on Information Theory*, 1962, vol. 8, no. 2, p. 179–187.
4. **Belongie S., Malik J., Puzicha J.** Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, vol. 24, no. 4, p. 509–522. DOI 10.1109/34.993558
5. **Korobov D. V., Patin M. V.** Metod raspoznavaniya shrifta teksta s izobrazheniya. *Molodoj uchenyj*, 2016, no. 12, p. 161–165. (in Russ.)
6. **Vapnik V. N.** Vosstanovlenie zavisimostej po empiricheskim dannym. Moscow, Nauka, 1979, 448 p. (in Russ.)
7. **Robert C. Martin.** The Clean Architecture. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 15.02.2019)

Материал поступил в редколлегию
Received
30.04.2019

Сведения об авторах / Information about the Authors

Прокопьев Иван Сергеевич, магистрант факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Ivan S. Prokopyev, Master's Student, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

i.prokopev3@g.nsu.ru

Шмаков Евгений Сергеевич, Software Architect / Tech lead, ООО "БНС" (ул. Фрунзе, 49, Новосибирск, Россия)

Evgeny S. Shmakov, Software Architect / Tech lead, LLC "BNS" (49 Frunze St., Novosibirsk, Russian Federation)

sjs-master@yandex.ru

УДК 004.054
DOI 10.25205/1818-7900-2019-17-3-93-110

Методика автоматического тестирования развивающегося веб-приложения

А. В. Ткачев, **Д. В. Иртегов**

Новосибирский государственный университет
Новосибирск, *Россия*

Аннотация

Статья посвящена методике автоматизированного тестирования системы автоматической оценки заданий по программированию NSUts. При разработке методики главным приоритетом было параллельное тестирование старой и новой версий приложения так, чтобы одни и те же или минимально модифицированные тесты проходили на двух версиях системы с различными архитектурами. Мы надеемся, что наш опыт будет полезен при выстраивании процесса разработки других приложений с длительным жизненным циклом.

Чтобы тестировать не только серверную, но и клиентскую часть веб-приложения, мы предлагаем использовать инструменты типа Selenium WebDriver для симуляции действий пользователей, посылая команды настоящим браузерам. В методике применяется известный шаблон проектирования Page Object и рассматривается ряд приемов, позволяющих снизить хрупкость разрабатываемых тестов и упростить их адаптацию для работы с новой версией системы.

В статье также описано применение данной методики для организации тестирования системы NSUts и проведен анализ ее эффективности. Анализ показал, что оценочное покрытие кода данными тестами достаточно высоко, и потому методику можно считать эффективной и применять на схожих веб-приложениях.

Ключевые слова

тестирование веб-приложений, Selenium WebDriver, функциональное тестирование

Для цитирования

Ткачев А. В., *Иртегов Д. В.* Методика автоматического тестирования развивающегося веб-приложения // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 93–110. DOI 10.25205/1818-7900-2019-17-3-93-110

Developing Web-Application's Automated Testing Technique

A. V. Tkachev, **D. V. Irtegov**

Novosibirsk State University
Novosibirsk, *Russian Federation*

Abstract

The article is devoted to the technique of automated testing of NSUts – automatic assessment system for programming tasks developed at NSU. The main priority for this technique is to test both the old and the new versions of the application, so that the same or minimally modified tests could be executed on two versions of the system with different architectures. This could be useful while organizing the development process for other applications with a long life cycle.

To test not only the server but also the client side of the web application, we suggest using tools like Selenium WebDriver to simulate user actions by sending commands to real browsers. We use the well-known Page Object design pattern to handle differences in HTML layout and functionality, and describe a number of ways to make developed tests less fragile and easily adapt those to work with the new version of the system.

The article also describes the use of this technique to organize automated testing of the NSUts system and analyzes its effectiveness. The analysis shows that the estimated code coverage by these tests is quite high, and therefore the technique can be considered effective and applied to other similar web applications.

Keywords
web-application testing, Selenium WebDriver, functional testing
For citation
Tkachev A. V., Irtegov D. V. Developing Web-Application's Automated Testing Technique. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 93–110. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-93-110

Введение

При разработке системы автоматической оценки заданий по программированию NSUts авторы столкнулись с необходимостью улучшить процедуры контроля качества. Необходимость была связана, главным образом, с планом перевода системы на новую архитектуру и стек технологий. При этом у заказчиков и пользователей возникал резонный вопрос: можно ли доверять новому коду в той же степени, что и старой системе, у которой был накоплен большой (и преимущественно позитивный) опыт эксплуатации? Чтобы снять этот вопрос, было принято решение организовать автоматизированное тестирование старой и новой систем. Методике организации этого тестирования и посвящена данная работа. Мы надеемся, что сама методика и соображения, исходя из которых мы ее создавали, могут быть полезны при решении аналогичных задач, в первую очередь при построении систем поддержки образовательного процесса.

Разработку программного обеспечения можно описать как формализацию требований. В начальный момент требования заказчика доступны лишь в виде неформального описания, даже не всегда в виде единого документа на естественном языке. На основе этих описаний разрабатывается описание системы на машинно-читаемом языке или языках. Машинно-читаемость сама по себе налагает достаточно высокие требования на уровень формализации. В свете этого рассуждения контроль качества программного обеспечения можно описать как повторную формализацию тех же требований, как правило, на другом машинно-читаемом языке (например, на входном языке разрабатываемой системы) в надежде на то, что вероятность дважды совершить одну и ту ошибку существенно ниже.

В зависимости от характера задач формализацию целесообразно выполнять на разных этапах и в разных видах. В некоторых приложениях требования просты и стабильны, а стоимость возможной ошибки очень велика, поэтому целесообразно вложиться в формализацию самих требований, после чего можно применять инструменты для автоматической генерации кода и тестов к нему, а иногда и автоматической верификации.

В других ситуациях требования сложны или просто велики по объему и меняются часто, так что формализация на раннем этапе оказывается экономически нецелесообразной. В этом случае часто прибегают к ручному тестированию, когда формализацию требований в итоге выполняет тестировщик. В нашем случае занимать студентов ручным тестированием было политически невозможно, поэтому автоматическое тестирование было сочтено наиболее приемлемым вариантом.

Для системы NSUts уже были разработаны комплекты автоматизированных тестов на основе Selenium, но попытки применить их для тестирования новой системы выявили их хрупкость, в первую очередь чувствительность к изменениям верстки HTML. Поскольку новый стек технологий (генерация HTML на стороне клиента при помощи фреймворка React) предполагал полное изменение верстки, это делало старые тесты практически непригодными. Поэтому было решено разработать методику, позволяющую составить автоматические тесты таким образом, чтобы они работали как с новой, так и со старой версией системы, и требовали при этом минимальных изменений для запуска после внесения модификаций.

Обзор предметной области

В НГУ давно существует, активно развивается и используется система для автоматической оценки заданий по программированию NSUts. Она используется в учебном процессе,

для проведения школьных и вузовских олимпиад по программированию и тренировок сборной НГУ. Система достаточно сложна по сравнению с типичными студенческими проектами (1 406 файлов, 148 115 строк кода) и содержит около двадцати крупных функциональных модулей: собственно оценка заданий, построение рейтинга по различным правилам, возможность задавать вопросы и т. д. В 2010 г. для системы было разработано техническое задание в соответствии с требованиями ГОСТ 19.201-78, которое заняло 49 страниц. Поскольку система развивается, появляются новые требования, и этот документ потерял актуальность. Ему на замену в 2018 г. было составлено новое описание функциональности, которое содержит 37 пунктов, соответствующих отдельным страницам системы. Для каждой из этих страниц описано в среднем по 5 функций. Сам документ занимает более 50 страниц понятного человеку текста (рис. 1).

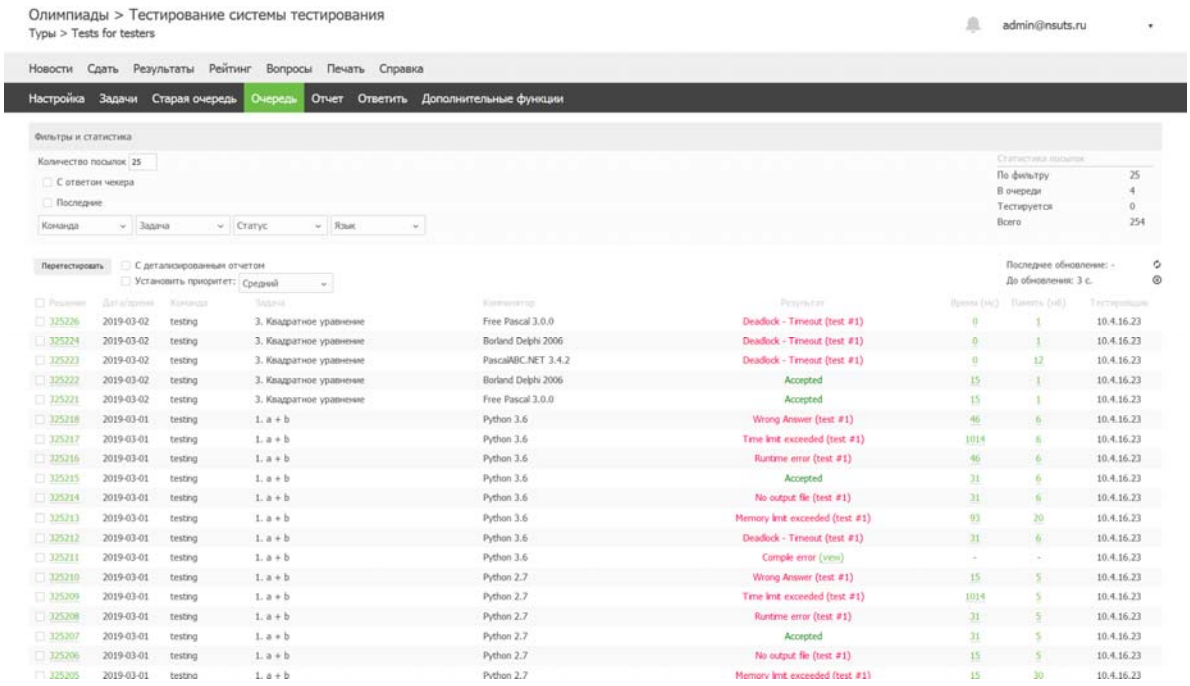


Рис. 1. Внешний вид одного из модулей системы

Fig. 1. Interface of One of the System's Modules

Ежегодно в системе проводится несколько различных олимпиад. Кроме того, на протяжении всего учебного года она используется преподавателями и студентами для сдачи заданий по курсам «Программирование на языке высокого уровня» и для тренировки олимпиадных команд НГУ. Каждый год на Открытой Всесибирской олимпиаде им. И. В. Поттосина представляется тур со специальной задачей с уникальными правилами.

Правила проведения некоторых олимпиад – например, регионального этапа Всероссийской олимпиады по информатике (РОИ) – меняются за несколько месяцев до их проведения, что требует оперативного внесения изменений в код системы. Иногда это требует существенной доработки, поскольку изменения затрагивают множество модулей сразу. Например, обновление правил РОИ в 2019 г. привело к изменению 972 строк в 14 файлах.

Обзор существующих решений

Изначально требования заказчика выражены неформально – даже если они подробно описаны в документах типа технического задания, они все еще описаны на естественном языке,

по которому невозможно сгенерировать конечный продукт. Потому при контроле качества, как и при разработке, эти требования рано или поздно формализуются в программный код или в действия, которые выполняет тестировщик. В зависимости от того, насколько сложны требования и как часто они изменяются, экономически выгодно формализовать их так или иначе (рис. 2).

Рассмотрим основные применяемые на практике подходы к контролю качества:

- ручное тестирование – формализация в ходе выполнения описанных неформально сценариев;
- формальная верификация – требования описываются на формальном языке, что позволяет затем автоматически генерировать тесты или верифицировать саму программу на соответствие этим требованиям;
- автоматическое тестирование – формализация требований в виде программного кода (тестовых сценариев).

Рис. 2. Схема разных подходов к формализации требований.
Сверху вниз: ручное тестирование, формальная верификация, автоматическое тестирование

Fig. 2. Different Approaches to Requirements Formalization.
From top to bottom: manual testing, formal verification, automated testing

Ручное тестирование

Если требования достаточно сложны или меняются часто, может оказаться очень эффективным простой подход ручного тестирования: человек, который достаточно хорошо знаком с неформально выраженными требованиями (иногда не в исходной их форме, а в форме плана тестирования), по сути формализует их в процессе выполнения различных сценариев в приложении либо может записать их в виде последовательности действий, которую должен выполнить другой специально обученный человек. Данная технология широко используется при разработке приложений поддержки бизнеса, учебного процесса, веб-приложений и др.

В процессе отладки нового кода, разработчики в той или иной форме выполняют отдельные операции по ручному тестированию. Но при этом ошибка в коде и пропуск ее при тести-

ровании могут быть причинно связанными и не могут считаться независимыми в теоретико-вероятностном смысле. Поэтому главный принцип, изложенный во введении, – что при самом программировании и при контроле качества мы выполняем формализацию требований, по возможности независимо, – не выполняется. Поэтому полагаться на тестирование самими разработчиками для контроля качества невозможно. В промышленности тестированием обычно занимаются другие люди. При этом ручное тестирование составляет значительную долю общих трудозатрат проекта, и во многих случаях эту долю воспринимают как проблему и так или иначе стараются снизить [1; 2].

Одно из усовершенствований идеи ручного тестирования привело к созданию подхода «Capture and Replay», при котором компьютер записывает действия пользователя и затем может их воспроизводить. Этот подход особенно популярен при тестировании приложений с графическим и веб-интерфейсом. Построенные таким образом тесты довольно хрупки, т. е. перестают работать при незначительных изменениях в приложении. Во многих случаях даже небольшие изменения функциональности или визуального дизайна требуют полной перестройки всего набора тестов. Встречаются также попытки генерации тестов по журналам доступа реальных пользователей [3; 4]. При обсуждении данного подхода обычно не рассматривают вопрос, насколько правомерна генерация тестов не на основе требований.

Ключевым недостатком Capture and Replay является сам принцип работы записывающих действия пользователей продуктов. Человек выполняет некоторые действия осмысленно, а программа механически записывает его действия. При взаимодействии с различными элементами на странице программа запоминает какие-то отличительные свойства этих элементов, чтобы в дальнейшем найти эти же самые элементы снова. Такими свойствами могут быть, например, текстовые надписи, значение атрибута id или XPath-выражение. id – это уникальная строковая метка, привязанная разработчиками веб-приложения, а язык XPath позволяет указать на конкретный элемент в иерархии элементов, описанных на языке разметки HTML. В листинге 1 представлен пример (рис. 3).

```
<div id="header" class="panel">
    <div class="title"><a href="select_olympiad.cgi">Выбор олимпиады</a></div>
    <div class="right">
        <div class="dropdown">
            <span><span>admin@nsuts.ru</span></span>
            <div class="links">
                <a href="edit_profile.cgi">Профиль</a>
                <a href="login.cgi?logout=1">Выйти</a>
            </div>
        </div>
    </div>
</div>
```

Рис. 3. Фрагмент HTML-кода типовой страницы NSUts

Fig. 3. Fragment of Typical NSUts Page HTML-Code

Для этого примера автоматические инструменты могли бы записать id элемента “header”, при нажатии на ссылки – их текст, например, «Выйти». Поскольку далеко не каждый элемент может иметь id, а текст может повторяться по несколько раз, многие инструменты просто записывают соответствующее XPath-выражение. Ссылке «Профиль» соответствует следующее выражение: /div/div[2]/div/div/a[1].

Однако к приложениям со сложным внутренним состоянием, в которых выполнение одной и той же последовательности действий приводит к разным результатам, данный подход оказывается неприменим. Например, при создании в NSUts олимпиады в базе появляется новая запись, и пользователь в списке видит новый пункт. При записи сценария человек нажимает на такой пункт, и инструмент запомнит какое-то его свойство. Если создать новую

олимпиаду, эти свойства уже будут другими, а записанный сценарий будет обращаться к старым. С другой стороны, если у олимпиады будет такое же название, то неизвестно, старую или новую выберет инструмент. Получается, что для приложений подобного рода невозможно составить адекватный сценарий, не описывая некоторую логику приложения, чего запись действий пользователя не предполагает.

Формальная верификация

На другой стороне спектра находится формальная верификация. Формализация требований происходит в процессе их описания на специальном языке, по которому в дальнейшем можно сгенерировать код или верифицировать, что написанный код соответствует описанным требованиям. Этот подход больше соответствует предметным областям, в которых требования к функциональности могут быть относительно простыми, но требование к надежности и безопасности стоит во главе угла. Примерами таких областей могут быть авионика [5; 6], медицина [7], автомобильная промышленность [8], разработка распределенных вычислительных программ [9] и аппаратного оборудования в соответствии с индустриальными стандартами [10].

Для решения таких задач ведется разработка специальных технологий и инструментов, но эти инструменты приспособлены именно к языкам, операционным системам, протоколам и др., которые применяются именно в соответствующих областях (например, к языку Ada). В нашей же задаче требования сложны и нестабильны, и потому данный подход является неприменимым. Поэтому даже перевод проекта на технологии и языки, для которых есть средства формальной верификации, не мог бы решить стоящую перед нами проблему.

Автоматическое тестирование при помощи тестовых сценариев

С учетом изложенных ранее соображений для нашей задачи наиболее разумным подходом представляется автоматическое тестирование, при котором формализация происходит при написании программного кода тестовых сценариев. По оценкам [1; 2], начальное написание тестирующих скриптов занимает больше времени по сравнению с Capture and Replay, но дальнейшая их доработка после внесения изменений в тестируемую систему оказывается дешевле, и уже после 2–3 обновлений общие трудозатраты на поддержку Capture and Replay становятся выше.

Часто встречаются различные попытки генерации тестов: по различным графам, соответствующим приложению [11], или с помощью автоматических инструментов по самому приложению [12; 13]. Идея тестирования в принципе заключается в независимой проверке тех же требований, а генерация тестов по коду приложения сводит эту идею на нет. Кроме того, поддерживать автоматически сгенерированный код, как правило, гораздо сложнее, нежели написанный людьми. Поэтому можно предположить, что по мере развития системы также окажется, что поддержка написанных вручную скриптов – наиболее выгодный по трудозатратам вариант.

Также в рассмотренной литературе встречается описание генерации тестов по модели в виде конечного автомата [14]. В этом случае страницы описываются группами состояний, а функциональность – переходами между этими состояниями. Система автоматически обходит все возможные пути в данном автомате, проверяя, что все требования выполняются после каждого перехода.

По сути, это иной способ записать требования формально: не в виде программного кода, а в виде диаграммы состояний конечного автомата. Дискуссия о сравнительных достоинствах разработки программ в виде текстов и в виде диаграмм состояний продолжается уже много десятилетий. На взгляд авторов, ограниченный успех CASE-инструментов, ориентированных на диаграммы, свидетельствует о том, что все-таки писать и поддерживать программы в форме текста людям почему-то удобнее. Кроме того, авторы работ по данной технологии тестирования предлагают инструменты только для языка Java, но не для языков, используемых в нашем проекте.

Проектирование методики

Как было показано ранее, наиболее подходящим подходом для нашего проекта является формализация требований в виде тестирующих скриптов. Для того чтобы упростить эту формализацию, можно собрать и систематизировать имеющиеся требования в виде некоторого документа. Таким документом может быть, например, техническое задание, но в целом достаточно будет и простого, но подробного описания функциональности.

Инструменты для тестирования веб-приложений можно разбить на две категории:

- те, что «притворяются» браузерами и отправляют только HTTP-запросы;
- настоящие браузеры или их эмуляторы, исполняющие также клиентский Javascript-код.

Первые применимы для тестирования серверной части веб-приложения, но если на клиентской стороне также присутствует нетривиальная логика, которую мы хотим проверить, то нужно использовать вторые. В нашем случае даже старая версия системы реализовала ряд нетривиальных функций (например, вопросы и ответы, новости) при помощи Javascript, а новая версия системы полностью основана на генерации страниц средствами Javascript, что делает тестирование на уровне запросов HTTP полностью неприменимым.

Ко второй категории можно отнести headless-браузеры (которые не отрисовывают страницу при работе) и другие инструменты, предоставляющие API для работы с каким-то конкретным браузерным движком. Такие версии браузеров, а тем более сторонние инструменты на их основе, имеют ряд отличий (иногда недокументированных) от обычных браузеров, и их использование может привести к неожиданным результатам при тестировании.

Кроме того, headless-браузеры предоставляют API, несовместимые между собой. В NSUts заявлена поддержка Firefox и Chrome / Chromium, поэтому разработка тестов при помощи headless-браузера потребовала бы работы с двумя разными API.

Наиболее популярный инструмент, решающий эти проблемы – это Selenium [1]: он позволяет обращаться к реальным браузерам и имеет при этом общий для них всех API, что упрощает написание тестирующих скриптов.

В прошлом в нашем проекте уже предпринимались попытки разработать тестовые сценарии с использованием инструмента Selenium WebDriver, который будет подробнее рассматриваться далее. Сценарии были разработаны на языке Perl, для унификации с основным кодом системы. Как отмечено в работе [2], поддержка тестов является трудоемкой и дорогостоящей задачей, поскольку должна выполняться вручную [15]. Из-за нехватки ресурсов и некоторых специфических проблем, которые будут обсуждаться далее, поддержка этих скриптов проводилась в недостаточном объеме, поэтому они отстали от реальной системы и оказались неприменимы.

Кроме того, продолжали развиваться браузеры и Selenium, и получилось так, что библиотека, использованная для реализации скриптов в нашем проекте, была снята с поддержки, а доступные для новой версии Selenium библиотеки для Perl неофициальны и несовместимы со старыми [2]. Поскольку в новой архитектуре системы мы также отказываемся от Perl, было решено было решено писать новые тесты на Python. Предполагается, что применение методики позволит сделать процесс более системным, а тесты – более поддерживаемыми.

Особенности применения методики

Обновление архитектуры в NSUts

Проведенное командой разработчиков NSUts исследование [16] показало, что прототипы системы, использующие технологии AJAX / AJAJ, создают гораздо меньшую нагрузку на сервер, что критично при большой нагрузке. Например, во время интернет-тура Открытой Всесибирской олимпиады по программированию количество одновременно активных поль-

[1] URL: https://www.seleniumhq.org/

[2] См.: https://metacpan.org/release/Test-WWW-Selenium и https://metacpan.org/release/Selenium-Remote-Driver

зователей превосходило тысячу. При этом активность многих этих пользователей заключалась в частом (иногда раз в несколько секунд) обновлении страницы рейтинга, что приближало систему к исчерпанию мощности процессоров и ряда других параметров. Поэтому в настоящее время система находится в процессе поэтапного перехода на новую архитектуру. Системой постоянно пользуются студенты и преподаватели, и потому необходимо избежать негативных последствий при внесении изменений: нужно осуществить этот переход так, чтобы с точки зрения пользователей система продолжила работать как прежде.

Таким образом, перед нами стоит следующая задача: написать тесты так, чтобы они не только поддерживались в будущем, но и позволяли убедиться, что модули, разработанные на замену оригинальным, имеют ту же функциональность. Для этого хотелось бы составить тесты так, чтобы их можно было запускать на обеих версиях системы, т. е. тестировать именно функциональность системы, а не ее реализацию. Понятно, что тесту в любом случае придется учитывать различия из-за особенностей реализации, поэтому в ходе применения методики нужно было придумать способ, с помощью которого адаптация написанного для исходной системы теста к работе с новой версией будет требовать минимального написания нового кода.

Использование Selenium

Selenium WebDriver позволяет взаимодействовать со страницей – передавать нажатия мыши и клавиш клавиатуры элементам страницы. Для этого нужно иметь возможность указать, с каким элементом (например, ссылкой или кнопкой) надо взаимодействовать. В Selenium есть 8 способов найти элемент:

- по HTML-атрибуту id;
- по HTML-атрибуту name, который обычно имеют только различные поля ввода;
- по XPath-выражению, соответствующему элементу;
- по тексту ссылки;
- по частичному тексту ссылки;
- по имени HTML-тега;
- по классу элемента;
- по CSS-селектору.

XPath – это место элемента в структуре HTML-документа, поэтому он, по определению, существует у любого элемента страницы. Но при малейших изменениях в верстке структура документа меняется, и XPath может перестать соответствовать тому элементу, которому он соответствовал раньше. Остальные способы применимы не для всех элементов (например, не все элементы являются ссылками) или являются недостаточно точными (например, на странице может быть много элементов с одинаковым именем HTML-тега).

Поиск элементов по id – метке, которую привязывают сами разработчики, – оказывается наиболее точным и наименее хрупким из доступных способов указать конкретный элемент. Атрибут id можно оставить прежним даже после полного изменения верстки страницы. Это позволяет устранить целый класс различий между двумя реализациями системы: можно «семантически связывать» элементы, которые выполняют одну и ту же функцию в разных реализациях, просто указав для них одинаковый id.

Главный недостаток этого способа в нашей ситуации заключается в том, что при разработке старого кода не было действующих соглашений по стилю кодирования, требовавших расставлять эти метки. Поэтому большинство элементов, с которыми нужно взаимодействовать тестирующим скриптам, этих меток не имели. Внесение изменений в старый код, особенно при отсутствии тестов, является нежелательным, поскольку может привести к созданию ошибок. Однако использование id имеет значительные преимущества и вдобавок является достаточно безопасным изменением, поскольку добавление атрибута id не влияет на функциональность.

Первоначально ставилось требование, чтобы тесты проходили на старом коде системы без каких-либо изменений этого кода. Преимущества, которые давала расстановка id, были сочтены настолько существенными, что это требование было ослаблено.

Другая проблема состоит в том, что в системах типа NSUts содержимое страниц в основном состоит из списков различных сущностей: олимпиады, задачи, решения. Из-за этого на странице присутствует множество однотипных элементов, и потому в id элементов добавляется переменное значение (чаще всего, первичный ключ сущности, рис. 4):

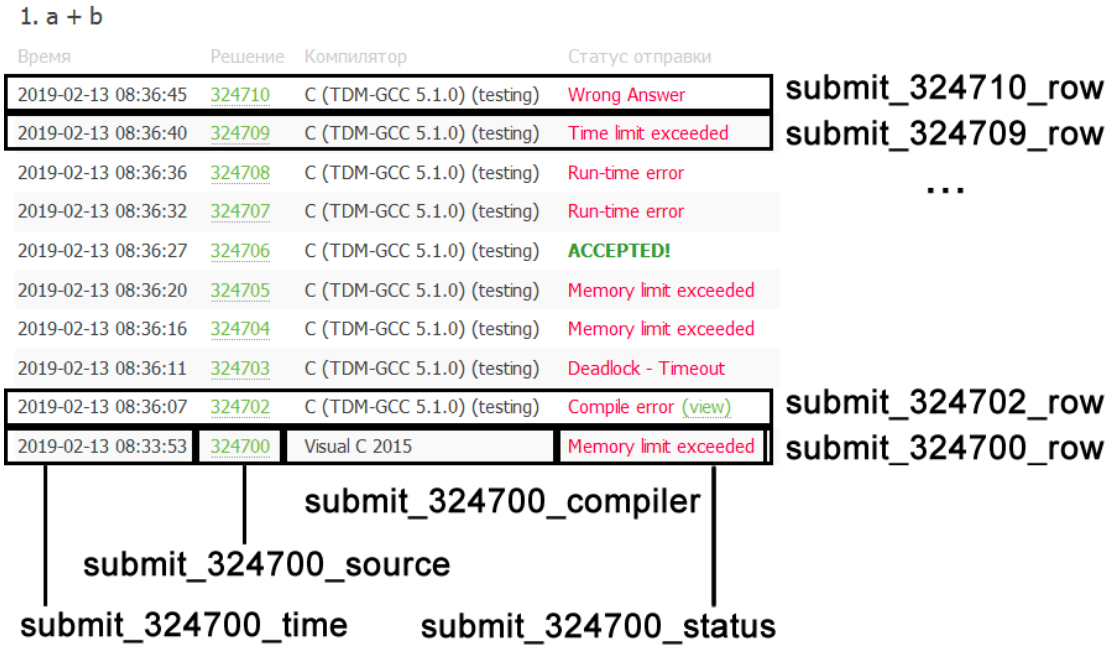


Рис. 4. id однотипных элементов
Fig. 4. Similar Elements' ids

Тестирующему скрипту нужно взаимодействовать с конкретным элементом, а для этого необходимо выяснить, какое переменное значение ему соответствует. Предлагается при создании сущностей генерировать уникальные значения – например, при отправке текста программы на проверку в системе NSUts можно добавить комментарий с текущим временем и случайно сгенерированной строкой. Затем придется перебрать все доступные сущности в поисках этого значения и таким образом вычислить искомый id.

Шаблон проектирования Page Object

Чтобы минимизировать изменения, необходимые для адаптации тестов к новым версиям старых модулей, был разработан промежуточный программный интерфейс для взаимодействия с приложением. Для разработки этого интерфейса использован шаблон проектирования, известный как Page Object. Он состоит в том, что каждой странице приложения в соответствие ставится класс (в нашем случае – класс языка Python), а доступная на странице функциональность описывается методами этого класса. Это позволяет описывать в тестовых сценариях логику взаимодействия в терминах высокоуровневого функционального описания системы, а не дублировать множество команд, например, нажатия мыши по конкретным элементам страницы. Код, использующий библиотеки для управления браузером, скрыт внутри классов – наследников описания страницы. Исследования [17] показывают, что ввиду указанных причин применение этого шаблона упрощает поддержку тестовых скриптов.

Поскольку функциональность старой и новой версий модулей должна совпадать, можно сделать классы страниц абстрактными и предоставлять для них по две реализации. Применение полиморфизма позволит передавать экземпляры классов для разных реализаций в код тестовых сценариев, написанный для работы с интерфейсом их базового класса. Код самих тестовых сценариев при этом может вообще остаться без изменений.

```
...

PRINT_PAGE_SUBMIT_TOO_LARGE = u"Превышено ограничение на размер файла!"
TEST_PRINT_LARGE_SUBMIT = "T" * (262144 + 1)

class TestPrintPage(NsutsTest):
    def setUp(self):
        NsutsTest.setUp(self)
        self.tour_id = PageAdminTour.make_test_tour(self, OLYMPIAD_ID, do_logout=False)

    def test_1_submit_on_print(self):
        ...

    def test_2_submit_large(self):
        self.page.open()
        submit_successful, message = self.page.submit_text(TEST_PRINT_LARGE_SUBMIT)
        self.assertFalse(submit_successful)
        self.assertEqual(message, PRINT_PAGE_SUBMIT_TOO_LARGE)

    def test_3_admin_mark_printed(self):
        ...

    def tearDown(self):
        PageAdminTour.remove_test_tour(self, OLYMPIAD_ID, self.tour_id)
        NsutsTest.tearDown(self)
```

Рис. 5. Фрагмент тестового сценария, использующего методы базового класса страницы
Fig. 5. Fragment of Test Script, which Uses Methods of Page's Base Class

```
...

class PrintPageFormPerl(PrintPageForm):
    def __init__(self, engine):
        PrintPageForm.__init__(self, engine)

    def input_text(self, code):
        ...

    def submit(self):
        self.engine.remember_page()
        self.engine.driver.find_element_by_xpath("//*[@type='submit']").click()
        self.engine.wait_for_page_to_change()

        # waiting for redirect to happen
        self.engine.remember_page()
        self.engine.wait_for_page_to_change()

        return self.engine.driver.find_element_by_id("print_form_message").get_attribute('innerHTML')

```

Рис. 6. Фрагмент реализации класса страницы, использующий Selenium
Fig. 6. Fragment of Page Class Implementation with Selenium Usage

```
...

class PrintPageFormReact(PrintPageForm):
    def __init__(self, engine):
        PrintPageForm.__init__(self, engine)

    def input_text(self, code):
        ...

    def submit(self):
        self.engine.driver.find_element_by_id("print_form_submit_button").click()

        # wait for ajax request
        time.sleep(2)

        return self.engine.driver.find_element_by_id("print_form_message").get_attribute('innerHTML')

```

Рис. 7. Фрагмент реализации класса страницы, использующий Selenium
Fig. 7. Fragment of Page Class Implementation with Selenium Usage

С тех пор, как систему начали переводить на новую архитектуру, прошло уже несколько лет, и за это время появились новые требования. Было решено реализовывать новую функциональность только в новых версиях модулей, чтобы способствовать переходу на них. Влияние новой функциональности нужно учитывать наравне с различиями в реализации модулей. Это делается путем написания разной реализации одних и тех же методов в соответствующих Page Object. Совпадающую функциональность можно описывать в базовом классе. Как было сказано, во многих простых случаях достаточно использовать одинаковые id элементов, чтобы устранить различия для тестирующих скриптов.

Параметризация тестов

В системе может быть множество параметров, которые влияют на функциональность отдельных ее элементов. Например, в NSUts бывают разные правила олимпиад, настройки тура, различный набор привилегий у пользователей. При этом необязательно писать разные тесты: помимо разных реализаций для страниц тест может и принимать другие параметры, и учитывать их. Это, однако, может привести к комбинаторному взрыву, при котором из-за различных комбинаций значений всех параметров получается слишком большое число возможных вариантов, и потому сильно возрастает время прохождения тестов. Были обнаружены исследования [18], в которых описаны методики сокращения числа тестов, обеспечивающие при этом приемлемый уровень кодового покрытия.

В настоящий момент наши тестовые скрипты работают с небольшим числом параметров, но в дальнейшем рассматривается возможность добавления новых и применения указанной методики для эффективного управления временем прохождения тестов.

Организация тестового стенда

Для запуска тестов и проверки написанных разработчиками новых версий страниц был подготовлен тестовый стенд. Разработчики могут запускать тесты и на собственных машинах, однако можно сэкономить их время, проделав настройку один раз. В стенде используются Selenium Server, последние версии Google Chrome и Mozilla Firefox, соответствующие им Selenium-драйвера и библиотека для работы с Selenium. Все эти компоненты периодически обновляются, и чтобы тесты работали, необходимо использовать согласованный набор компонент. На машинах разработчиков может не быть всех браузеров или установлены старые версии компонент, из-за чего тесты, которые запускаются в одном месте, перестают работать в другом. Чтобы этого избежать, можно использовать стенд, который поддерживается в актуальном состоянии одним из разработчиков.

Компоненты тестового стенда

В тестовый стенд должны входить:

- экземпляр тестируемой системы;
- одна или несколько машин, на которых запускаются браузеры и Selenium Server;
- машина, на которой запускаются тестирующие скрипты, управляющие браузерами через Selenium Server.

Все эти компоненты могут располагаться на разных компьютерах.

В нашем случае тестируемая система – NSUts – состоит из двух частей: веб-сервера, работающего на Debian, и набора тестирующих клиентов, работающих на Windows. На последних установлены компиляторы разных версий, и запущен процесс, который общается с веб-сервером NSUts, получает решения участников, компилирует и запускает их проверку на наборах входных данных, после чего сообщает веб-серверу, успешно ли решение прошло проверку. Схема работы изображена на рис. 8.

Рис. 8. Схема работы системы NSUts

Fig. 8. Scheme Illustrating NSUts Workflow

В целях верификации работы веб-сервера NSUts не обязательно разворачивать полноценные тестирующие клиенты – достаточно, чтобы сервер получил запрос на выдачу решения, а затем получил отчет о его проверке. Поэтому в наших тестирующих скриптах реализованы «поддельные» тестирующие клиенты, которые скачивают решение и выдают заранее согласованный ответ. Также это сокращает время, необходимое для тестирования веб-сервера, поскольку «поддельные» клиенты могут сообщить о результатах проверки мгновенно, а при использовании настоящих клиентов приходилось бы ожидать, пока этот процесс действительно произойдет.

Запуск браузеров на разных машинах позволяет сократить время выполнения тестов, запуская их параллельно. Пока что нам это не потребовалось, и потому для тестового стенда достаточно одной машины, на которой установлены все необходимые браузеры. Поскольку большинство пользователей NSUts пользуются ОС Microsoft Windows, необходимо запускать браузеры именно в этой ОС. Веб-сервер NSUts работает на Debian, и его можно развернуть

либо на отдельной машине, либо внутри виртуальной машины. Нами был выбран второй вариант: виртуальная машина работает на той же машине, на которой запускаются браузеры. Таким образом, тестовый стенд представлен всего одним компьютером.

Запуск тестирующих скриптов происходит на машине разработчика. Для этого ему требуется установить дистрибутив языка, на котором реализованы скрипты (в нашем случае Python) и библиотеку для работы с Selenium на этом языке. Данный подход сопряжен с серьезной проблемой: в Selenium Server не реализована аутентификация, поэтому размещение его на публичном или даже локальном в рамках сети НГУ IP-адресе привело бы к тому, что скрипты (не обязательно разработанные нами) мог бы запускать кто угодно. Для решения этой проблемы, удаленный доступ к Selenium Server осуществляется через ssh-туннель.

Рабочий цикл тестового стенда

Тестовый стенд подразумевает однократную первоначальную настройку, т. е. развертывание тестируемой системы и установку браузеров, Selenium и драйверов, а затем многократный запуск тестов, перед которым выполняется обновление тестируемой системы. По ряду причин (часть из которых описана ниже) очистка от результатов предыдущих тестов осуществляется как отдельная операция и должна активироваться вручную.

Для удобства развертывания системы как для тестирования, так и для разработки, был создан набор скриптов, упрощающий данный процесс. Пользователю нужно загрузить эти скрипты из специального репозитория, и по мере их работы указывать желаемые настройки. Скрипты сами устанавливают необходимые пакеты с помощью менеджера пакетов, создают необходимые директории и файлы (например, конфигурацию для веб-сервера apache), создают и заполняют таблицы в базе данных, загружают код веб-приложения из отдельного репозитория. Поскольку развертывание системы проводится один раз при создании стенда, необходимость пользовательского ввода считается некритическим недостатком.

«Боевая» система также не переразвертывается каждый раз как минимум в том смысле, что данные (учетные записи пользователей, олимпиады, задачи, сданные решения и т. д.) должны сохраняться, даже когда вносятся изменения в схему базы данных. Поэтому в репозиторий добавляются специальные скрипты, которые инкрементально обновляют развернутую систему, например SQL-запросы для изменения схемы БД. Тестируемая система обновляется точно так же: перед очередным запуском тестов из репозитория загружаются изменения и запускаются все новые скрипты. В отличие от набора скриптов для развертывания системы данные скрипты не требуют пользовательского ввода, поэтому процедура обновления происходит полностью автоматически.

Чтобы запустить тестирующие скрипты, разработчики получают их из отдельного репозитория и указывают в конфигурационном файле, по какому адресу доступны Selenium Server и веб-сервер NSUts. Наличие конфигурационного файла позволяет легко использовать машину с Selenium Server и браузерами для тестирования реальной системы. Схема работы стенда показана на рис. 9.

Применение методики для тестирования системы NSUts

В настоящее время по указанной методике разработаны тесты для четырех модулей системы NSUts, и два из этих модулей были успешно переведены на новую архитектуру. После минимальных изменений в коде классов страниц новые модули были протестированы. В одном случае изменения не потребовались вовсе, несмотря на значительные различия во внешнем виде пользовательского интерфейса и в структуре HTML (табл. 1). Сейчас указанные модули уже используются вместо их старых аналогов, и готовятся новые версии модулей, для которых уже написаны тесты.

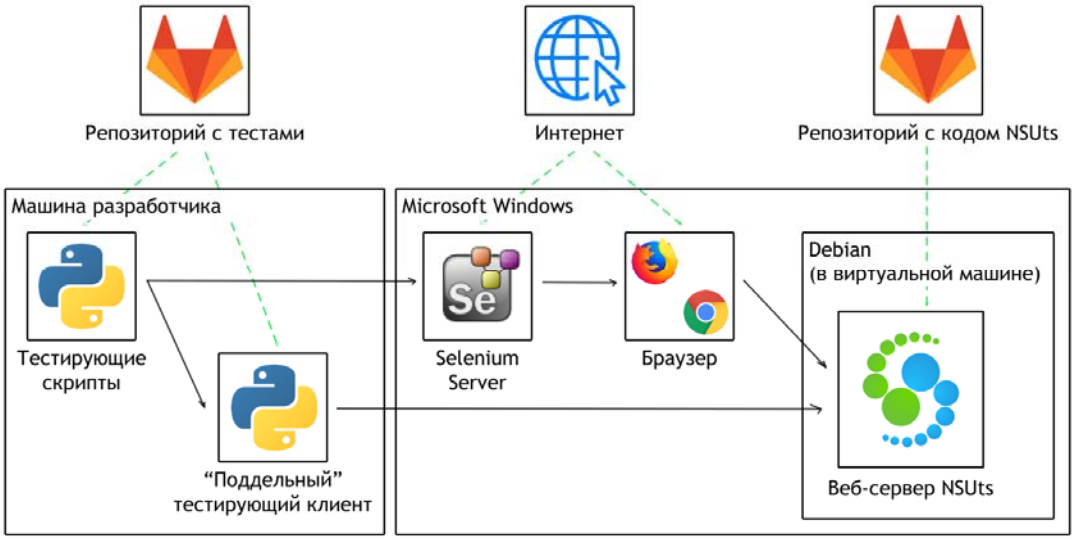


Рис. 9. Схема работы тестового стенда
Fig. 9. Test Stand Workflow

Таблица 1

Объем кода тестов для страниц системы
и изменений для адаптации тестов к новой версии системы

Table 1

Amount of Tests Code for System Pages
and Changes to Adapt Tests to Work with the New Version of the System

Страница	Число строк	
	общего кода	добавленных
Новости	399	–
Отправить	315	–
Печать	241	95
Результаты	392	0

Изменения не потребовались на странице со статической информацией (результаты отправленных на тестирование решений), т. е. новым представлением тех же данных, и нескольким доступным действиям, не ведущим к их изменениям (например, просмотр текста отправленного решения или ошибки компиляции). Здесь очень сильно пригодилось использование совпадающих значений атрибута id.

Однако на странице с возможностью отправки информации (печать текста на принтере) потребовалось сделать отдельные, хотя и очень схожие, реализации класса страницы для разных модулей. Дело в том, что страницы работают по-разному, хотя для пользователя поведение системы выглядит практически неотличимым. Так, в одном случае при отправке формы происходит несколько перенаправлений, а в другом – асинхронный запрос, и это различие необходимо отразить в коде соответствующих странице классов.

Анализ эффективности методики

Одной из самых популярных метрик, используемых для оценки качества тестов, является процент покрытия ими исходного кода. К сожалению, данную метрику сложно измерить без

специализированных инструментов. В нашем случае используется Perl в старой архитектуре и PHP + Javascript в новой, и не было обнаружено инструмента, который поддерживал бы все три языка.

Поэтому для оценки покрытия был использован метод Монте-Карло, заключающийся в следующем: в файле выбирается случайная строка, она изменяется с целью изменить логику программы, и запускаются тесты. Изменения в строку вносятся вручную, чтобы новая строка была корректна синтаксически, но некорректна семантически (например, замена условия на противоположное). Если тесты проходят успешно, то строка считается непокрытой, и наоборот. Не все строки в файле влияют на логику программы (например, комментарии, пустые строки или строки с различными скобками), и потому такие строки не учитываются. Номера строк генерируются псевдослучайным образом. Если генератор выдает номер строки, которая уже была протестирована, она все равно учитывается в выборке. По полученной выборке можно построить график, показывающий, как изменяется процент покрытия с увеличением размера выборки. То, что процент покрытия прекращает существенно изменяться, говорит о том, что этот метод действительно позволяет оценить реальный процент покрытия.

Подсчет был совершен для двух модулей, имеющих реализацию на обеих архитектурах. Далее представлены графики сходимости покрытия (рис. 10) и приведены получившиеся результаты (табл. 2).

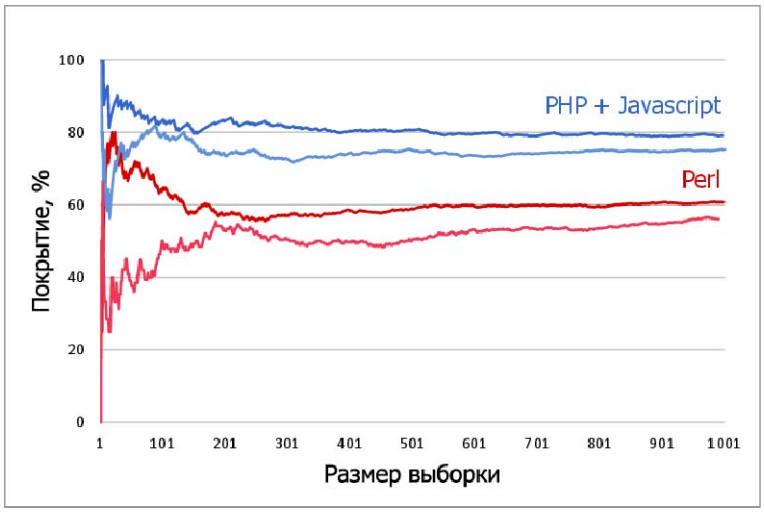


Рис. 10. Сходимость покрытия страниц «Печать» и «Результаты» (на Perl и PHP + Javascript)
*Fig. 10. C*overage Convergence for “Print” and “Report” Pages (in Perl and in PHP + Javascript)

Таблица 2

Оценка покрытия кода модулей тестами, %

Table 2

Evaluated Code Coverage, %

Страница	Perl	PHP + Javascript
Печать	60,88	79,1
Результаты	56,05	75,2

Из табл. 2 видно, что тест покрывает код на новой архитектуре даже лучше, чем на старой. Это позволяет предположить, что в старой системе существует значительная доля «мертвого» кода. Покрытие неполное, поскольку на страницах присутствуют элементы, ко-

торые не влияют на функциональность, например заголовки и другой текст, предназначенный для пользователя. Некоторые элементы просто не обладают функциональностью (например, дата отправки в таблицах на указанных страницах), и потому тесты на них написаны не были. Соответствующие им строки действительно не покрыты тестами, ведь их изменение или отсутствие не отражается на прохождении тестов.

Строки, изменение которых привело к провалу тестов, реально соответствуют тестируемой функциональности, поэтому методику можно считать применимой и, учитывая достаточно высокий процент покрытия, эффективной. При необходимости после получения оценки покрытия тесты можно доработать, и протестировать изменения в менее важных или недостающих, если таковые имеются, элементах и функциональности.

Заключение

В работе представлена методика автоматического тестирования развивающихся веб-приложений. Поскольку мы разрабатывали методику для приложения со сложными и часто меняющимися требованиями, она основывается на написании тестирующих сценариев. А так как мы разрабатывали ее для веб-приложения, она предлагает использование Selenium WebDriver. Важной особенностью методики является применение полиморфизма при разработке скриптов, позволяющее написать тест так, чтобы он мог проверять несколько разных реализаций одной и той же функциональности. Адаптация скрипта к новой реализации может вообще не потребовать написания нового кода, если приложение соответствующим образом адаптировано к тестированию. В нашем случае главным методом адаптации является расстановка HTML-тегов id у управляющих элементов интерфейса приложения.

Методика используется при организации тестирования системы NSUts, и спроектированные по ней тесты существенно упрощают задачу разработчиков, позволяя быстро и надежно убедиться, что переводимые ими на новую архитектуру модули имеют ту же функциональность, что и их старые аналоги. Показано, что оценочное покрытие кода данными тестами достаточно высоко.

Список литературы / References

1. **Leotta M.**, **Clerissi D.**, **Ricca F.**, **Tonella P.** Approaches and Tools for Automated End-to-End Web Testing. *Advances in Computers*, 2016, vol. 101, p. 193–237. DOI 10.1016/bs.adcom.2015.11.007
2. **Leotta M.**, **Clerissi D.**, **Ricca F.**, **Tonella P.** Capture-replay vs. programmable web testing: An empirical assessment during test case evolution. In: Proceedings – Working Conference on Reverse Engineering, 2013, p. 272–281. DOI 10.1109/wcre.2013.6671302
3. **Шмейлин Б. З.** Современные технологии тестирования Web приложений // Системы и средства информ. 2009. Доп. выпуск. С. 138–147.
   Shmeilin B. Z. Sovremennye tekhnologii testirovaniya Web prilozhenii [Modern technologies of web-applications testing]. *Sistemy i sredstva inform*., 2009, add. issue, p. 138–147. (in Russ.)
4. **Chen C.**, **Chen Y.**, **Miao H.**, **Wang H.** Usage-pattern based statistical web testing and reliability measurement. *Procedia Computer Science*, 2013, p. 140. DOI 10.1016/j.procs.2013.09.020
5. **Muñoz C.**, **Narkawicz A.**, **Dutle A.** From formal requirements to highly assured software for unmanned aircraft systems. *Lecture Notes in Computer Science*, 2018, vol. 10951, p. 647–652. DOI 10.1007/978-3-319-95582-7_38

6. **Denman W.**, **Zaki M. H.**, **Tahar S.**, **Rodrigues L.** Towards flight control verification using automated theorem proving. *Lecture Notes in Computer Science*, 2011, vol. 6617, p. 89–100. DOI 10.1007/978-3-642-20398-5_8
7. **Harrison M. D.**, **Freitas L.**, **Drinnan M.**, **Campos J. C.**, **Masci P.**, **di Maria C.**, **Whitaker M.** Formal techniques in the safety analysis of software components of a new dialysis machine. *Science of Computer Programming*, 2019, vol. 175, p. 17–34. DOI 10.1016/j.scico.2019.02.003
8. **Nellen J.**, **Rambow T.**, **Waez M. T. B.**, **Abraham E.**, **Katoen J. P.** Formal verification of automotive Simulink controller models: empirical technical challenges, evaluation and recommendations. *Lecture Notes in Computer Science*, 2018, vol. 10951, p. 382–398. DOI 10.1007/978-3-319-95582-7_23
9. **Khanna D.**, **Sharma S.**, **Rodríguez C.**, **Purandare R.** Dynamic symbolic verification of MPI programs systems. *Lecture Notes in Computer Science*, 2018, vol. 10951, p. 466–684. DOI 10.1007/978-3-319-95582-7_28
10. **Steiner W.**, **Dutertre B.** Automated formal verification of the TTEthernet synchronization quality. *Lecture Notes in Computer Science*, 2011, vol. 6617, p. 375–390. DOI 10.1007/978-3-642-20398-5_27
11. **Liu C.-H.**, **Kung D. C.**, **Hsia P.**, **Hsu C.-T.** An object-based data flow testing approach for web applications. *International Journal of Software Engineering and Knowledge Engineering*, 2001, vol. 11, no. 2, p. 157–179. DOI 10.1109/apaq.2000.883773
12. **Choudhary S.**, **Dincturk**, **M.E.**, **Mirtaheri**, **S.M.** Building rich internet applications models: Example of a better strategy. In: Proc. of the International Conference on Web Engineering, ICWE. Springer, 2013. DOI 10.1007/978-3-642-39200-9_25
13. **Benedikt M.**, **Freire J.**, **Godefroid P.** VeriWeb: Automatically testing dynamic web sites. In: Proc. 11th Intern. WWW Conf. May, 2002.
14. **Chen J.**, **Chovanec S.** Towards Specification-Based Web Testing. Springer, 2002, vol. 2376, p. 165–171. DOI 10.1007/3-540-45745-3_15
15. **Mirzaaghaei M.** Automatic test suite evolution. In: Proc. of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering – SIGSOFT/FSE'11, 2011, p. 396–399. DOI 10.1145/2025113.2025172
16. **Боженкова Е. Н.**, **Иртегов Д. В.**, **Колбин Я. С.** Оптимизация производительности веб-интерфейса приложения NSUts средствами динамического HTML // Вестник НГУ. Серия: Информационные технологии. 2015. Т. 13, № 2. С. 13–21.
    Bozhenkova E. N., **Irtegov D. V.**, **Kolbin Ya. S.** Optimizing performance of NSUts application web-interface via dynamic HTML. *Vestnik NSU. Series: Information Technologies*, 2015, vol. 13, no. 2, p. 13–21. (in Russ.)
17. **Leotta M.**, **Clerissi D.**, **Ricca F.**, **Spadaro C.** Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study. In: IEEE 6th Int. Conf. on Software Testing, Verification and Validation Workshops, 2013, p. 108–113. DOI 10.1109/icstw.2013.19
18. **Masuda S.**, **Matsuodani T.**, **Tsuda**, **K.** A method of creating testing pattern for pair-wise method by using knowledge of parameter values. In: Procedia Computer Science, 2013, p. 521. DOI 10.1016/j.procs.2013.09.131

Материал поступил в редколлегию
Received
04.03.2019

Сведения об авторах / Information about the Authors

Ткачев Александр Витальевич, магистрант факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Alexander V. Tkachev, Master's Student, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

alexander@tkachov.ru
D-4763-2019
ORCID 0000-0001-7721-9527

Иртегов Дмитрий Валентинович, доцент, заведующий лабораторией, факультет информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Dmitry V. Irtegov, Associate Professor, Head of Laboratory, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

fat@nsu.ru
ORCID 0000-0003-1007-134X

УДК 004.912
DOI 10.25205/1818-7900-2019-17-3-111-122

Разработка методов семантического поиска в Интернете, основанных на древовидных лингвистических шаблонах

И. И. Филиппов [1], **Д. Е. Пальчунов** [1, 2], **П. А. Степанов** [1]

[1] *Новосибирский государственный университет*
Новосибирск, Россия
[2] *Институт математики им. С. Л. Соболева СО РАН*
Новосибирск, Россия

Аннотация

Статья посвящена разработке методов поиска информации в Интернете. В настоящее время задача быстрого автоматизированного извлечения данных из различных источников является чрезвычайно актуальной. Цель данной работы – создание инструментария для решения этой задачи. Идея подхода заключается в разработке полуавтоматических методов составления лингвистических шаблонов для заданного фрагмента атомарной диаграммы. Процесс построения шаблонов по данному фрагменту атомарной диаграммы выполняется в 2 этапа: 1) порождение простых шаблонов; 2) с помощью полученных простых шаблонов создание более сложных или реже встречающихся шаблонов. С помощью полученных шаблонов можно находить необходимую информацию, обрабатывая тексты естественного языка, представленные в Интернете.

Ключевые слова

лингвистические шаблоны, синтаксические деревья, фрагменты атомарных диаграмм, семантический поиск информации, извлечение знаний

Благодарности

Исследование выполнено при частичной финансовой поддержке Президиума СО РАН (проект «Инженерия интенсиональных онтологий в дедуктивных и информационных системах» Комплексной программы ФНИ СО РАН II.1)

Для цитирования

Филиппов И. И., *Пальчунов Д. Е.*, *Степанов П. А.* Разработка методов семантического поиска в Интернете, основанных на древовидных лингвистических шаблонах // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 111–122. DOI 10.25205/1818-7900-2019-17-3-111-122

Development of Methods of Semantic Search on the Internet, Based on Treelike Linguistic Templates

I. I. Filippov [1], **D. E. Palchunov** [1, 2], **P. A. Stepanov** [1]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation
[2] *Sobolev Institute of Mathematics SB RAS*
Novosibirsk, Russian Federation

Abstract

The article is devoted to development of methods of search of information on the Internet. Currently the issue of fast and automatic data extraction from different sources is very actual. The aim of this work is creation of instrument to solve this problem. The idea of the approach is to develop semi-automatic methods of linguistic templates construction for given fragment of atomic diagram. Algorithm of templates construction for the fragment of atomic diagram is performed in 2 stages. On the 1st stage generation of the simple templates is done, on the 2nd stage more seldom and

complicated templates are found, using templates from 1st part of algorithm. Processing texts in natural language on the Internet using gotten templates, needed information may be found.

Keywords

linguistic templates, syntax trees, fragments of atomic diagrams, semantic search, knowledge extraction

For citation

Filippov I. I., Palchunov D. E., Stepanov P. A. Development of Methods of Semantic Search on the Internet, Based on Treelike Linguistic Templates. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 111–122. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-111-122

Введение

В современном мире задача быстрого и автоматизированного извлечения данных является очень актуальной. Например, когда возникает необходимость найти некоторую информацию в Интернете, нужно просмотреть множество страниц, попробовать несколько вариантов запросов, проанализировать большое количество информации и выбрать что-то действительно удовлетворяющее требованиям. Для некоторых запросов подобный процесс может потребовать очень много времени и усилий. Поэтому было бы крайне полезным наличие программного инструмента, который мог бы предложить пользователю не огромное количество различных Интернет-ресурсов, а несколько, например 3–5, в которых содержалась бы нужная ему информация. Разработка такой программной системы является целью данной работы.

Предлагаемый подход основан на разработке полуавтоматических методов порождения лингвистических шаблонов для поискового запроса, формально представленного в виде некоторого фрагмента атомарной диаграммы алгебраической системы. Построение лингвистического шаблона для конкретного фрагмента атомарной диаграммы выполняется в два этапа. На первом этапе происходит поиск простых шаблонов. На втором этапе на основе построенных простых шаблонов выполняется поиск более сложных вариантов шаблонов или редко встречающихся в текстах. С помощью полученных шаблонов происходит поиск и извлечение необходимой информации путем проверки соответствия текста естественного языка лингвистическому шаблону.

В настоящее время существует множество систем, решающих подобные задачи [1–4]. Самые популярные среди них – поисковые системы, такие как Гугл, Яндекс и др. Они основаны на поиске по ключевым словам с применением различных эвристик и способны выдать нужную информацию для простых запросов, однако возникают проблемы с более сложными поисковыми запросами.

Также задачу поиска информации призваны решать различные вопросно-ответные системы [5; 6]. Их важным преимуществом является то, что они используют различные методы извлечения знаний, способны уточнять запрос, учитывать контекст, также могут использовать методы машинного обучения. Таким образом, подобные системы способны лучше отвечать на вопросы пользователя, однако у них также есть недостатки.

Существует два вида вопросно-ответных систем: системы, отвечающие на вопросы пользователя, используя внутреннюю базу данных, и системы, порождающие ответы при помощи знаний, содержащихся в Интернете [5]. В первом случае нужной информации может просто не быть в базе, либо она может быть устаревшей, а пользователю обычно не предоставляется возможность каким-либо образом расширить базу. С системами второго типа все немного сложнее. Поскольку они извлекают информацию из Интернета, они обычно пытаются различными способами свести сложные запросы к последовательности простых и далее раскручивать запрос по цепочке. При этом ответы на простые запросы получаются с помощью все тех же поисковых систем. Таким образом, ответ на запрос находится по ключевым словам.

Однако одна и та же информация с семантической точки зрения может быть представлена совершенно по-разному. Это справедливо даже для самых простых запросов. Например, для запроса «Где родился Эйнштейн?» ответ может быть представлен как «Эйнштейн родился в городе Ульм» либо, например, как «Город Ульм подарил миру Эйнштейна». Таким обра-

зом, поиск по ключевым словам «Эйнштейн» и «родился» не обнаружит второй вариант. Для сложных запросов подобные наблюдения еще более существенны.

Таким образом, цель данной работы – разработать программный инструмент, который будет выполнять поиск требуемой информации без необходимости представлять поисковый запрос в виде последовательности нескольких ключевых слов. Определенные методы и технологии семантического поиска были описаны в ряде работ [1–8]. Идея разрабатываемого подхода заключается в автоматизированном построении лингвистических шаблонов для конкретного поискового запроса пользователя. При этом пользователь может «описать» свой поисковый запрос (т. е. поставить в соответствие запросу множество лингвистических шаблонов), если готовый лингвистический шаблон, полностью соответствующий поисковой потребности, отсутствует в базе. Пользователь может контролировать процесс извлечения и порождения шаблонов, для того чтобы избежать построения ошибочных вариантов поисковых запросов. При создании лингвистических шаблонов можно рассматривать параметризованные фрагменты атомарных диаграмм [6] и для них строить семантически параметризованные лингвистические шаблоны.

Понятие параметризованного лингвистического шаблона

Методы, используемые в данной работе, основаны на понятиях фрагмента атомарной диаграммы [9; 10], параметризованного фрагмента атомарной диаграммы [6] и лингвистического шаблона [8; 11; 12]. Атомарной диаграммой модели называется множество истинных на ней атомарных предложений и отрицаний атомарных предложений [9; 10]. Атомарным предложением называется бесквантворное предложение вида $P(c_1, \ldots, c_n)$, где P – символ n-местного предиката, а $c_1, \ldots, c_n$ – символы констант. Фрагмент атомарной диаграммы может быть параметризован, в таком случае часть предикатов, входящих в данный фрагмент атомарной диаграммы, являются заранее не определенными; при реализации информационного поиска предикаты-параметры оснащаются конкретными предикатами [6]. Каждому поисковому запросу пользователя ставится в соответствие конечный фрагмент атомарной диаграммы. Каждому конечному фрагменту атомарной диаграммы соответствует один или несколько лингвистических шаблонов. Таким образом, каждый поисковый запрос пользователя формально определяется множеством лингвистических шаблонов.

Лингвистический шаблон – это параметрический набор условий на искомую структуру текста [8; 11]. Он может быть представлен несколькими способами. Один из возможных вариантов – представление лингвистического шаблона в виде аналога регулярного выражения над множеством слов [8; 11].

Рассмотрим, например, шаблон вида *NOUN(«им.п.») NOUN(«рд.п.»)*. Данный шаблон описывает последовательность из двух существительных, где первое находится в именительном падеже, а второе – в родительном. Данному шаблону удовлетворяют такие словосочетания, как «отец ребенка», «стены дома» и др. В данном шаблоне параметрически задано значение падежа; таким образом, слова в словосочетании, найденном при помощи данного шаблона, должны удовлетворять условиям, наложенным на падеж. При этом могут варьироваться другие параметры слов, такие как род, число и пр. Один шаблон может содержать несколько параметризаций. Недостатком подобных шаблонов является то, что они неустойчивы к порядку слов. Например, фраза «ребенка отец» уже не удовлетворяет данному шаблону, хотя с семантической точки зрения фразы «отец ребенка» и «ребенка отец» эквивалентны.

Другой вариант шаблона – так называемый древовидный шаблон. Древовидный шаблон – это дерево, в вершинах которого находятся слова, а ребро между вершинами обозначает, что между словами, находящимися в данных вершинах, существует связь. Пример такого шаблона для запроса «Где родился человек?» представлен на рис. 1. Древовидные шаблоны устойчивы к порядку слов, поскольку они учитывают лишь типы связей между словами, но не порядок их в предложении. Например, шаблону с рис. 1 удовлетворяют предложения

Рис. 1. Пример лингвистического шаблона
Fig. 1. Пример лингвистического шаблона

«Эйнштейн родился в Ульме», «Эйнштейн в Ульме родился», «В Ульме родился Эйнштейн» и т. д. В работе используются именно древовидные шаблоны.

Для того чтобы сформировать запрос к поисковой системе, необходимо расположить слова из шаблона в том порядке, в котором они были в изначальном предложении (можно выбрать любой, если вариантов несколько, поскольку поисковые системы не учитывают порядок), а также подставить в запрос конкретный субъект и исключить объект. Для конкретного человека запрос может выглядеть, например, так: «Пушкин родился в».

Порождение лингвистических шаблонов

Сначала опишем общий алгоритм порождения шаблонов:

1) поиск предложений, содержащих необходимую информацию;
2) извлечение шаблонов из найденных предложений;
3) расширение множества предложений и возвращение к шагу 2.

Рассмотрим подробно каждый из этих этапов.

Поиск предложений, содержащих необходимую информацию

На первом этапе необходимо найти предложения, в которых содержится информация о текущем запросе. Мы используем два основных метода получения таких предложений.

1. Национальный корпус русского языка [13; 14] – это информационно-справочная система, основанная на собрании русских текстов в электронной форме. Корпус насчитывает около 30 млн предложений различной тематики, таких как художественные тексты, газетная публицистика, технические тексты, научные тексты, тексты личной переписки и др. Корпус позволяет быстро выполнять лексико-грамматический поиск, используя различные морфологические, грамматические и семантические признаки. Также корпус позволяет получить результаты поиска в удобном для машинной обработки формате. Однако, поскольку корпус вручную обрабатывается и поддерживается, объем текстов в нем значительно меньше, чем в Интернете. Поэтому данный метод используется в основном на первом этапе, где выполняется поиск более простых предложений.

2. Поисковая система Гугл [1]. Всем известная система выполняет поиск по всей сети и предоставляет результаты в удобочитаемом для человека формате. Для автоматического извлечения предложений необходимо потратить больше ресурсов, чем для извлечения с помощью корпуса русского языка, однако множество извлеченных предложений гораздо больше по объему и разнообразию.

На первом этапе необходимо найти наиболее простые предложения, отвечающие требованиям. В основном поиск производится по ключевым словам без учета морфологических признаков. Например, для запроса «Где родился человек?» можно выполнить поиск по ключевому слову «родиться». Очевидно, что не все предложения, содержащие слово «родиться», будут содержать информацию о месте рождения человека. Поэтому полученные данные необходимо обработать вручную, чтобы отсеять ненужные варианты.

Извлечение шаблонов из найденных предложений

Далее, по найденным предложениям нужно построить лингвистические шаблоны. Для получения шаблонов используются синтаксические деревья предложений, извлеченных на первом шаге. Синтаксическое дерево (дерево разбора) – структура в виде ориентированного дерева, представляющая синтаксическую структуру предложения. Пример синтаксического

[1] www.google.com

дерева для предложения «Дельта Волги является восьмым чудом света» приведен на рис. 2. Корректно построенное дерево разбора обладает следующими свойствами:

1) слова, находящиеся на разных вершинах одного ребра, связаны в исходном предложении. Причем слово, находящееся на нижнем уровне, является зависимым. Слово, находящееся в корне дерева, не имеет зависимого.

2) слова, находящиеся в разных ветках, не имеют прямой связи друг с другом и связаны только через зависимые слова. Например, слова «восьмым» и «света» не связаны напрямую, но они оба являются характеристиками слова «чудо».

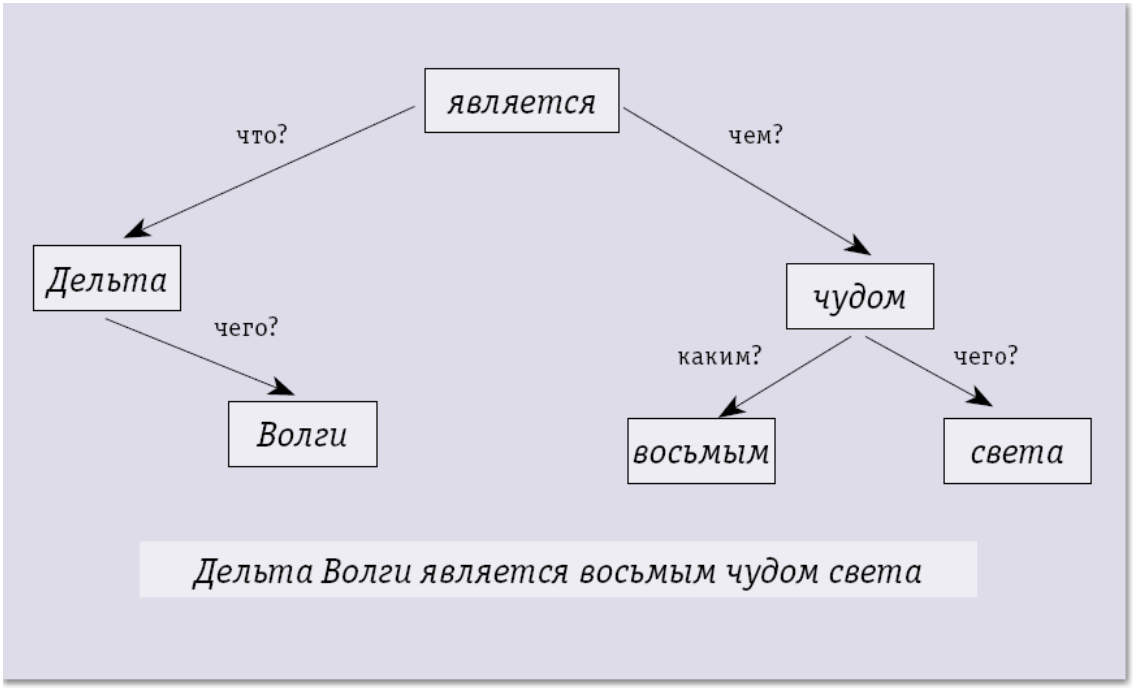


Рис. 2. Пример синтаксического дерева
Fig. 2. Syntax tree example

Таким образом, можно сделать следующие выводы. Если мы хотим выяснить, как два слова связаны в исходном предложении, нам достаточно найти минимальное поддерево, которое содержит в себе искомые слова. В данное дерево не попадут следующие категории вершин:

1) вершины, располагающиеся на более нижних уровнях, чем вершины, содержащие искомые слова. Очевидно, что данные вершины не имеет смысла включать в дерево, так как искомые слова никак (даже транзитивно) от них не зависят;

2) вершины, не являющиеся непосредственными предками искомых, но находящиеся на том же уровне или выше. Данные вершины не имеют прямой связи с искомыми, а значит, их также не имеет смысла включать в итоговое дерево;

3) вершины, находящиеся выше, чем наименьший общий предок искомых вершин. С семантической точки зрения, слова, находящиеся в таких вершинах, несут более глобальный смысл, чем исходные слова. Такая ситуация может возникнуть, если оба исходных слова находятся в придаточной части предложения, которая сама по себе зависит от других частей предложения. Очевидно, что вершины из данной категории также не нужно включать в искомое дерево.

Возвращаясь к лингвистическим шаблонам, можно заметить, что итоговое дерево и есть искомый шаблон, если в качестве искомых слов использовать *субъект* и *объект* исходного

предложения. В качестве примера рассмотрим предложение «Великолепный Коля внезапно родился в поистине огромной Москве, где его семья жила 10 лет». На рис. 3 можно увидеть дерево разбора данного предложения.

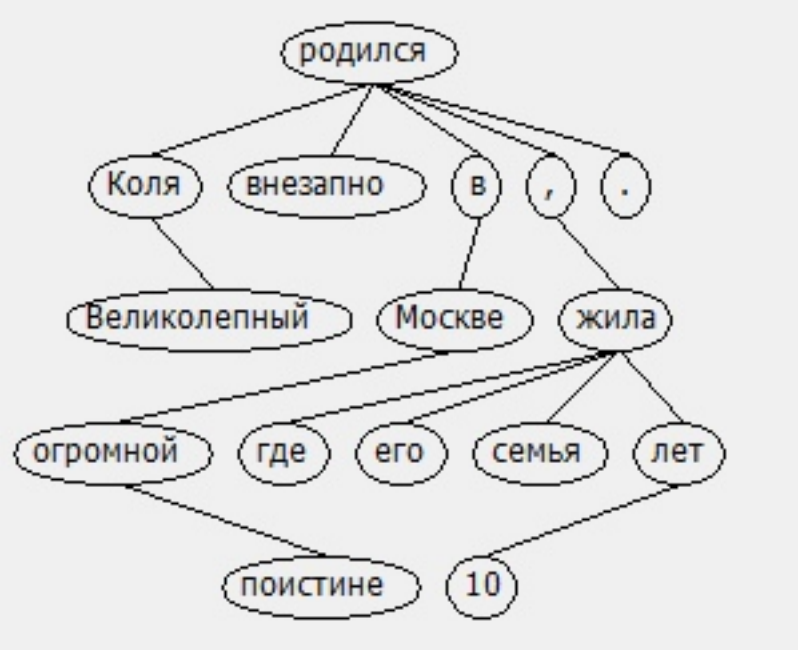


Рис. 3. Пример синтаксического дерева
Рис. 3. Syntax tree example

В данном дереве субъектом является слово «Коля», а объектом – «Москва». Причем предложение имеет достаточно сложную структуру с большим количеством дополнений. Однако если попытаться найти связь между словами «Коля» и «Москва», то в результате получится очень простой шаблон, аналогичный представленному на рис. 1.

Приведем еще два определения, которые понадобятся нам в текущей работе. Синтаксический анализ – это процесс сопоставления линейной последовательности лексем естественного языка с его формальной грамматикой. Результатом синтаксического анализа обычно является синтаксическое дерево. Синтаксический анализатор (или парсер) – инструмент, выполняющий синтаксический анализ [15; 16]. Теперь рассмотрим способы построения синтаксических деревьев. Всего существует два основных подхода.

1. Машинное обучение с учителем. Подход заключается в обучении парсера на предварительно размеченной выборке. Размеченные корпуса текстов можно получить с помощью корпуса русского языка. Этот подход достаточно прост в реализации, однако его точность оставляет желать лучшего. Модель, обученная на $^{5}/_{6}$ корпуса и протестированная на оставшейся части, показывает точность в 79,6 %. К тому же, перед тем как отдавать предложения на анализ, необходимо провести их лексический и морфологический анализ, что требует дополнительных ресурсов.

2. Метод, основанный на правилах. Основная идея заключается в создании набора правил, которые определяют, как проставлять связи в предложении. Данный подход дает наибольшую точность и строит корректные деревья разбора, если грамматика корректно описывает данное предложение. Однако данный подход очень ресурсоемок, поскольку он требует описания буквально всего русского языка.

Оба метода имеют свои плюсы и свои минусы. В своей работе мы использую парсер Solarix [17]. Это парсер с общедоступной лицензией, основанный на правилах. Грамматика языка была описана авторами парсера, поэтому данную задачу в этой работе мне решать не придется. Парсер решает следующие задачи:

1) лексический анализ – разбивка текста на предложения и слова;

2) морфологический анализ – определение морфологических признаков с учетом контекста;

3) лемматизация – приведение слов к начальной форме;

4) синтаксический анализ – определение синтаксических связей слов в предложении, построение дерева разбора.

Таким образом, парсер решает все необходимые задачи, и для построения шаблонов необходимо ввести в парсер множество полученных предложений. Существует два режима работы парсера.

1. Восходящий анализ. Идея метода заключается в разборе сначала конкретных слов, затем в связывании слов в пары, далее связывание пар с другими словами и парами и т. д., пока все слова предложения не окажутся связанными в одну структуру. Такой метод работает достаточно быстро, однако, если грамматика не полностью описывает данное предложение, либо предложение содержит неоднозначности, парсер может сделать ошибочные предположения о связи слов и в результате получить неточный результат. В целом полученные результаты достаточно корректны, и выбросы (результаты, сильно отличающиеся от среднего) встречаются довольно редко.

2. Нисходящий анализ. Идея метода заключается в выдвижении гипотез о крупномасштабной структуре предложения, далее эта структура уточняется, рекурсивно спускаясь до уровня отдельных слов. Данный метод работает очень медленно, поскольку, сделав ошибочное предположение на ранних этапах и опровергнув его на более поздних, алгоритм выполняет большое количество лишних действий. Однако данный подход может выдать все возможные варианты парсинга предложения в случае наличия неоднозначностей.

Поскольку быстрый восходящий анализ работает корректно в большинстве случаев, целесообразно использовать именно его для выполнения первичной обработки предложений. Переключаться на нисходящий анализ разумно только в случае, если восходящий анализ покажет неудовлетворительные результаты. На выборке из 50 предложений восходящий анализ показывает результат 3 секунды, тогда как нисходящий анализ требует 1,5 минуты.

В результате анализа полученных шаблонов были выявлены две категории шаблонов, которые должны быть исключены из итоговых результатов.

1. Шаблоны, содержащие только слова ОБЪЕКТ и СУБЪЕКТ. Подобные шаблоны могут возникнуть, если в полученном дереве разбора субъект и объект находятся в соседних узлах. В целом такая ситуация является корректной, но очевидно, что данные шаблоны невозможно использовать для дальнейшего поиска, следовательно, они должны быть отсеяны.

2. Шаблоны, встречающиеся менее чем в 0,5 % случаев. Поскольку парсер показывает не 100 % верные результаты, необходимо сделать его более устойчивым к ошибкам. Один из способов – отсеять шаблоны, которые встречаются очень редко, поскольку они, скорее всего, были получены в результате ошибочного анализа. Величина 0,5 % была получена чисто эвристически.

Стоит также отметить, что в полученном шаблоне на месте субъекта и объекта могут находиться не только отдельные слова, но также и словосочетания, см. пример на рис. 4. Однако весь анализ для таких случаев остается неизменным, данное предложение также удовлетворяет шаблону из рис. 1.

Расширение множества предложений

После того как полученные предложения проанализированы и получен набор шаблонов, необходимо выявить более редкие предложения, которые были упущены на первом этапе, и также извлечь из них шаблоны. Для выявления более редких предложений мы используем

Рис. 4. Субъект и объект представлены несколькими словами

Fig. 4. Subject and object are presented as several words

поисковую систему Гугл ввиду наличия большего объема и разнообразия данных. Например, на первом этапе для предиката «Где родился человек?» используются все предложения, содержащие слово «родиться». Очевидно, что большое количество предложений при этом теряется, поскольку в предложении, где идет речь о месте рождения, наличие слова «родиться» совсем необязательно.

Для того чтобы получить больше предложений, необходимо из существующих выбрать несколько таких, в которых субъект и объект широко представлены в текстах, находящихся в Интернете. Далее, сделав запрос вида «СУБЪЕКТ ОБЪЕКТ» в поисковой системе, мы получим множество различных вариантов того, как субъект и объект могут быть связаны в исходном предложении. Конечно, данные предложения могут содержать информацию, которая не относится к текущему предикату, поэтому полученные предложения также должны быть обработаны вручную.

Далее из полученных предложений извлекаются лингвистические шаблоны аналогично тому, как это было сделано на первом этапе. При желании можно повторить второй и третий этапы для получения большего количества шаблонов.

Примеры обработки предикатов

В качестве примера рассмотрим процесс получения шаблонов для вопроса «Где родился человек?». На первом этапе с помощью корпуса русского языка извлекаются около 300 шаблонов, содержащих слово «родиться». При этом поиск проводится без учета морфологических признаков, т. е. варианты со словами «родился», «родилась» и т. д. также представлены в итоговых результатах. Из полученных предложений примерно 50 содержат информацию о месте рождения человека. В результате проведенного анализа был получен набор новых лингвистических шаблонов. Вот самые популярные из них:

1) СУБЪЕКТ родился в ОБЪЕКТ: 19 вхождений в исходное множество;
2) ОБЪЕКТ СУБЪЕКТ родился: 4 вхождения;
3) СУБЪЕКТ родилась в году в ОБЪЕКТ: 3 вхождения.

Далее мы выбираем следующие пары «субъект – объект» для последующего анализа: Эйнштейн – Ульм, Ньютон – Вулсторп, Чехов – Таганрог. Проанализировав около 120 результатов запросов, мы получаем около 40 различных предложений, удовлетворяющих предикату. В результате дальнейшего анализа, помимо шаблонов, аналогичных полученным на первом этапе, также были получены принципиально новые варианты, такие как: «*ОБЪЕКТ родина СУБЪЕКТ*», «*Из уроженцев ОБЪЕКТ известен СУБЪЕКТ*», «*ОБЪЕКТ город подаривший миру СУБЪЕКТ*» и др.

Другой пример более сложного предиката: «Кто является президентом страны?». Самые популярные шаблоны, полученные на первом этапе по ключевому слову «президент»:

1) Президент СУБЪЕКТ ОБЪЕКТ;
2) ОБЪЕКТ, президент СУБЪЕКТ;
3) ОБЪЕКТ является президентом СУБЪЕКТ.

Шаблоны второго этапа, не найденные на первом:

1) Глава СУБЪЕКТ ОБЪЕКТ;
2) ОБЪЕКТ на посту главы СУБЪЕКТ.

Полученные результаты

Разработана программная система, реализующая описанный выше алгоритм. Для проверки корректности работы алгоритма была проведена серия тестов. В программную систему были введены 3 запроса: «*Место рождения человека*», «*Автор изобретения*», «*Президент страны*». Для каждого запроса протестировано 10 возможных субъектов, результаты работы оценивались экспертом. Оценка ставилась из множества {0|1}, где 1 ставилась в случае, если на найденной странице действительно содержалась искомая информация, и 0 в противном случае.

В ходе анализа полученных результатов выявлены 3 типа ошибок, т. е. все полученные ошибочные результаты можно разделить на 3 категории.

1. Ошибка первой категории возникает в случае, если предложение удовлетворяет шаблону, однако в более глобальном контексте было сказано, что это предложение ошибочно. Пример: «*Они ошибочно полагали, что лампочку изобрел Томас Эдисон*». Стоит отметить, что во всех случаях на этой же странице содержалась информация о реальном положении вещей, поэтому такие страницы были отмечены как успешный результат.

2. Ошибки второй категории возникали в случае, если предложение удовлетворяет шаблону, но все равно не отвечает на запрос. Пример такой ошибки для предиката «*Место рождения*»: «*Александр Пушкин родился в дворянской семье*». Чтобы избежать данной ошибки, необходимо учесть более широкий контекст, чем предложение. Ведь теоретически, «дворянская семья» вполне может быть названием населенного пункта. Стоит отметить, что подобные ошибки встречались достаточно редко, хотя, несомненно, они являются наиболее опасными.

3. Ошибки третьей категории появляются в случае, если в момент последнего редактирования страницы информация была еще действительна, но на текущий момент она устарела. Или у автора была некорректная информация, или же он намеренно вводит читателя в заблуждение. Пример: «*Президент США – Барак Обама*». Подобные ошибки даже сложно считать ошибкой, ведь в таком случае даже человек не сможет удостовериться, что информация действительно верная. Однако они имеют место, поэтому также были отмечены в текущей работе.

Всего для каждого запроса программа выдает 5 результатов с предполагаемым ответом. Таким образом, для каждого предиката было получено 50 результатов. Далее приведена таблица с количеством ошибок каждого типа для каждого запроса.

Количество ошибок, %
Number of Errors, %

Запрос	Категория ошибок		
	первая	вторая	третья
Место рождения	0	10	10
Автор изобретения	0	0	4
Президент страны	0	8	14

Можно заметить, что ошибок первой категории в данном случае не было, хотя они и были обнаружены в процессе тестирования. Можно предположить, что такие ошибки аналогичны ошибкам третьей категории, однако между ними есть существенная разница. В первом случае в тексте явно сказано, что данная информация не верна, в третьем же случае информация представляется корректной. Ну и логично, что в первом случае сразу же приводится корректная информация.

Что касается ошибок второй категории, можно понять, что количество таких ошибок зависит от запроса. Например, для запроса «*автор изобретения*» алгоритм не допустил подобных ошибок. Ну, действительно, сложно придумать предложение, удовлетворяющее, например, шаблону «*СУБЪЕКТ изобрел ОБЪЕКТ*» и имеющее какой-то посторонний смысл.

Всего было получено 150 результатов работы программы, из которых 127 были корректными. Таким образом, можно сказать, что на тестовой выборке точность алгоритма составляет 84,6 %, причем для каждой пары «субъект – запрос» как минимум 3 из 5 результатов были корректными. В целом полученные результаты можно считать успешными.

Заключение

В работе предложен алгоритм автоматизированного извлечения лингвистических шаблонов из текстов естественного языка. Были использованы древовидные лингвистические шаблоны в силу того, что они являются более устойчивыми к порядку слов в предложении.

Разработаны и протестированы методы построения синтаксических деревьев по предложениям естественного языка. Предложены критерии определения валидности различных лингвистических шаблонов.

Разработана и реализована программная система, позволяющая осуществлять семантический поиск и извлекать требуемые знания в текстах естественного языка, представленных в Интернете. Разработанная программная система протестирована, полученная точность 84,6 % представляется достаточно хорошей для решения рассмотренных задач и достижения поставленных целей.

Список литературы

1. **Осипов Г. С., Тихомиров И. А., Смирнов И. В.** Семантический поиск в сети Интернет средствами поисковой машины Exactus. URL: https://refdb.ru/look/3439993.html (дата обращения 18.05.2019).
2. **Нгуен Ба Нгок, Тузовский А. Ф.** Обзор подходов семантического поиска // Докл. ТУСУРа. 2010. № 2 (22), ч. 2. С. 234–237.
3. **Перминов С. В., Афанасьев С. В.** Семантический способ поиска информационных аномалий через web // Тр. СПИИРАН. 2006. Т. 3, № 1. С. 279–287.
4. **Guha R., McCool R., Miller E.** Semantic search. In: Proc. of the 12th Int. Conf. on World Wide Web, 2003, p. 700–709. DOI 10.1145/775152.775250
5. **Пальчунов Д. Е.** Поиск и извлечение знаний: порождение новых знаний на основе анализа текстов естественного языка // Философия науки. 2009. № 4 (43). С. 70–90.
6. **Деревянко Д. В., Пальчунов Д. Е.** Формальные методы разработки вопросно-ответной системы на естественном языке // Вестник НГУ. Серия: Информационные технологии. 2014. Т. 12, № 3. С. 34–47.
7. **Власов Д. Ю., Пальчунов Д. Е., Степанов П. А.** Автоматизация извлечения отношений между понятиями из текстов естественного языка // Вестник НГУ. Серия: Информационные технологии. 2010. Т. 8, № 3. С. 23–33.
8. **Пальчунов Д. Е., Степанов П. А.** Применение теоретико-модельных методов извлечения онтологических знаний в предметной области информационной безопасности // Программная инженерия. 2013. № 11. С. 8–16.
9. **Махасоева О. Г., Пальчунов Д. Е.** Автоматизированные методы построения атомарной диаграммы модели по тексту естественного языка // Вестник НГУ. Серия: Информационные технологии. 2014. Т. 12, № 2. С. 64–73.
10. **Корсун И. А., Пальчунов Д. Е.** Теоретико-модельные методы извлечения знаний о смысле понятий из текстов естественного языка // Вестник НГУ. Серия: Информационные технологии. 2016. Т. 14, № 3. С. 34–48.

11. **Степанов П. А.** Язык описания лингвистических шаблонов // Материалы Всерос. конф. с междунар. участием «Знания – Онтологии – Теории». 2013. Т. 2. С. 136–145.
12. **Степанов П. А., Пальчунов Д. Е., Мирзагитов А. А.** Методы анализа диалогов, основанные на теории речевых актов // Вестник НГУ. Серия: Информационные технологии. 2014. Т. 12, № 4. С. 102–111.
13. **Аброскин А. А.** Поиск по корпусу: проблемы и методы их решения // Национальный корпус русского языка: 2006–2008. Новые результаты и перспективы. 2009. С. 277–282.
14. **Апресян Ю. Д., Богуславский И. М., Иомдин Б. Л.** Синтаксически и семантически аннотированный корпус русского языка: современное состояние и перспективы // Национальный корпус русского языка: 2003–2005. С. 193–214.
15. **Антонова А. А.** Синтаксический анализатор для русского и английского языков // Информационно-аналитические аспекты в задачах управления: Сб. тр. ИСА РАН. М.: ЛКИ, 2007. Т. 29. С. 329–337.
16. Парсим русский язык // Хабрахабр. 2012. 20 июля. URL: https://habr.com/ru/post/148124/ (дата обращения 18.05.2019).
17. Парсер – морфологический и синтаксический анализатор русскоязычных текстов // Инструменты для NLP разработчика: лексика, морфология, синтаксис русского языка. 2019. URL: http://solarix.ru/parser.shtml (дата обращения 18.05.2019).

References

1. **Osipov G. S., Tikhomirov I. A., Smirnov I. V.** Semanticheskii poisk v seti Internet sredstvami poiskovoi mashiny Exactus [Semantic search on the Internet by means of a search engine Exactus]. URL: https://refdb.ru/look/3439993.html (accessed 18.05.2019). (in Russ.)
2. **Ba Ngoc Nguyen, Tuzovsky A. F.** A review of semantic search approaches. *Proc. of TUSUR University*, 2010, № 2 (22), part 2, p. 234–237. (in Russ.)
3. **Perminov S. V., Afanasyev S. V.** Semantic Search for Information Anomalies through Web Technique. *SPIIRAS Proc*., 2006, vol. 3, no. 1. (in Russ.)
4. **Guha R., McCool R., Miller E.** Semantic search. In: Proc. of the 12th Int. Conf. on World Wide Web, 2003, p. 700–709. DOI 10.1145/775152.775250
5. **Palchunov D. E.** Knowledge retrieval and elicitation: generation of new knowledge based on analysis of natural language texts. *Filosofiya nauki*, 2009, no. 4 (43), p. 70–90. (in Russ.)
6. **Derevyanko D. V., Palchunov D. E.** Formal methods of development of the question-answering system on natural language. *Vestnik NSU. Series: Information Technology*, 2014, vol. 12, no. 3, p. 34–47. (in Russ.)
7. **Vlasov D. Yu., Palchunov D. E., Stepanov P. A.** Automating the extraction of relations between concepts from natural language texts. *Vestnik NSU. Series: Information Technology*, 2010, vol. 8, no. 3, p. 23–33. (in Russ.)
8. **Palchunov D. E., Stepanov P. A.** The use of model-theoretic methods for extracting ontological knowledge in the domain of information security. *Programnaya ingeneriya*, 2013, no. 11, p. 8–16. (in Russ.)
9. **Makhasoeva O. G., Palchunov D. E.** Semi-automatic methods of a construction of the atomic diagrams from natural language texts. *Vestnik NSU. Series: Information Technology*, 2013, vol. 12, no. 2, p. 64–73. (in Russ.)
10. **Korsun I. A., Palchunov D. E.** Model-Theoretic Methods of Extraction of Knowledge on the Meaning of Concepts from the Natural Language Texts. *Vestnik NSU. Series: Information Technology*, 2016, vol. 14, no. 3, p. 34–48. (in Russ.)
11. **Stepanov P. A.** Linguistic patterns description language. In: All-Russian Conference with international participation “Knowledge – Ontology – Theory”. 2013, vol. 2, p. 136–145. (in Russ.)

12. **Stepanov P. A., Palchunov D. E., Mirzagitov A. A.** Methods of analysis of dialogues based on the theory of speech acts. *Vestnik NSU. Series: Information Technology*, 2014, vol. 12, no. 4, p. 102–111. (in Russ.)
13. **Abroskin A. A.** Poisk po korpusu: problem I metody ikh resheniya [Search in corpus: problems and methods of their solutions]. In: Natsional'nyi korpus russkogo yazyka: 2006–2008. Novye rezul'taty i perspektivy. 2009, p. 277–282. (in Russ.)
14. **Apresyan Yu. D., Boguslavskii I. M., Iomdin B. L.** Sintaksicheski i semanticheski annotirovannyi korpus russkogo yazyka: sovremennoe sostoyanie i perspektivy [Syntactically and semantically annotated corpus of the Russian language: current state and prospects]. In: Natsional'nyi korpus russkogo yazyka: 2003–2005. p. 193–214. (in Russ.)
15. **Antonova A. A.** Syntax analyzer for the Russian and English languages. In: Information and analytical aspects in problems of management. Moscow, LKI Publ., 2007, vol. 29, p. 329–337. (in Russ.)
16. Parsim russkii yazyk [Parse Russian language]. *Habrahabr*, 2012, June, 20. URL: https://habr.com/ru/post/148124/ (accessed 18.05.2019). (in Russ.)
17. Parser – morfologicheskii i sintaksicheskii analizator russkoyazychnykh tekstov [Parser – morphological and syntactic analyzer of Russian texts]. In: Instrumenty dlya NLP razrabotchika: leksika, morfologiya, sintaksis russkogo yazyka, 2019. URL: http://solarix.ru/parser.shtml (accessed 18.05.2019). (in Russ.)

Материал поступил в редколлегию
Received
05.06.2019

Сведения об авторах / Information about the Authors

Филиппов Иннокентий Игоревич, студент, Новосибирский государственный университет (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Innokentiy I. Filippov, Student, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

i.filippov1@g.nsu.ru

Пальчунов Дмитрий Евгеньевич, доктор физико-математических наук, ведущий научный сотрудник, Институт математики СО РАН (пр. Академика Коптюга, 4, Новосибирск, 630090, Россия)

Dmitry E. Palchunov, Doctor of Physical and Mathematical Sciences, Leading Researcher, Sobolev Institute of Mathematics SB RAS (4 Academician Koptyug Ave., Novosibirsk, 630090, Russian Federation)

palch@math.nsc.ru

Степанов Павел Андреевич, ассистент, Новосибирский государственный университет (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Pavel A. Stepanov, Assistant, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

stefan.nsk@gmail.com

УДК 004.652.5
DOI 10.25205/1818-7900-2019-17-3-123-134

Обзор объектно-ориентированной парадигмы в приложении к разработке баз данных

Е. А. Яценко

Сургутский государственный университет
Сургут, Россия

Аннотация

В статье приводится обзор проектов, технологий, программных продуктов, разрабатываемых для реализации идей объектно-ориентированного подхода к проектированию баз данных. В 1980-е гг. велось множество проектов, посвященных идее ООБД, многие специалисты ожидали, что в ближайшее время реляционные базы данных будут вытеснены объектно-ориентированными. Несмотря на впечатляющее количество проектов, проводимых как коллективами ученых, так и коммерческими компаниями, ориентированными на практическое внедрение, не появилось четкой формулировки объектно-ориентированной модели данных, каждый коллектив представлял свое видение применения принципов ООП к проектированию баз данных. Отсутствие универсальной модели данных с проработанным математическим аппаратом (как в случае реляционных баз данных) и сегодня является основной проблемой распространения ООСУБД. Однако применение реляционных СУБД порождает ряд проблем, которые наиболее остро ощущаются в таких областях, как системы автоматизированного проектирования, автоматизированное производство, системы, основанные на знаниях, и др. ООБД позволяют объединить программный код и данные, избежать различий между представлениями информации в базе и прикладной программе, вследствие чего интерес к ним проявляют современные разработчики. Современный рынок программного инструментария представлен рядом ООСУБД, но ни одна из них не может конкурировать с крупнейшими поставщиками СУБД.

Ключевые слова

объектно-ориентированный подход, объектно-ориентированная база данных, объектно-ориентированная СУБД, модель данных

Для цитирования

Яценко Е. А. Обзор объектно-ориентированной парадигмы в приложении к разработке баз данных // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 123–134. DOI 10.25205/1818-7900-2019-17-3-123-134

Overview of Object-Oriented Paradigm in an Appendix to the Development of Databases

E. A. Yatsenko

Surgut State University
Surgut, Russian Federation

Abstract

The article provides an overview of projects, technologies, software products developed to implement the ideas of an object-oriented approach to database design. In the 80s of the 20th century, there were many projects devoted to the idea of OODB, many experts expected that in the near future relational databases would be crowded out with object-oriented ones. Despite the impressive number of projects conducted by both teams of scientists and commercial companies focused on practical implementation, there was no clear formulation of an object-oriented data model, each team presented its own vision of applying object-oriented concepts to database design. The absence of a universal data

model, with a well-developed mathematical apparatus (as in the case of relational databases), is still the main problem in the distribution of an OODBMS. However, the use of relational DBMS raises a lot of problems that are most acutely felt in areas such as computer-aided design, computer-aided production, knowledge-based systems, and others. OODB allow to combine the program code and data, to avoid differences between the representations of information in the database and the application program, as a result of which modern developers show interest in them. There are a lot of OODBMS, but they cannot compete with the largest storage organization systems.

Keywords

object-oriented approach, object-oriented database, object-oriented database, data model

For citation

Yatsenko E. A. Overview of Object-Oriented Paradigm in an Appendix to the Development of Databases. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 123–134. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-123-134

Объектно-ориентированная база данных (ООБД) – база данных, данные в которой структурированы в соответствии с моделью данных, основанной на понятиях и принципах объектно-ориентированного проектирования (ООП).

В 1980-е гг. велось множество проектов, посвященных идее ООБД, многие специалисты были уверены, что в ближайшие 2–3 года реляционные базы данных будут вытеснены объектно-ориентированными. Сергей Дмитриевич Кузнецов среди проектов, оказавших наибольшее влияние на развитие ООБД, называет Encore [1] (Брауновский университет, Провиденс, Род-Айленд, США); Cactis [2] (Университет Колорадо, США), Thor [3] (Массачусетский технологический институт, США); Exodus [4] (Висконсинский университет, США); Pisa [5] (университеты Глазго и Сент-Э́ндрюсм, Шотландия).

Большой вклад в развитие объектно-ориентированной парадигмы сделала группа исследовательского института OGI (Oregon Graduate Institute). OGI был уникальным, частным, последипломным исследовательским университетом в округе Вашингтон, штат Орегон, в западной части Портленда, с 1963 по 2001 г., с 1989 г. именовался Орегонским институтом высших учебных заведений. Результатом деятельности OGI в области объектно-ориентированных баз данных стала ООСУБД Gemstone [6; 7]. В качестве базового языка GemStone использовался Smalltalk [8; 9]. Данный язык разработан в 1970-х гг. на основе уже существовавших в то время концепций, в нем была введена новая терминология, являющаяся наиболее характерной для современных языков программирования. Smalltalk основан на идее посылки сообщений, реализует возможность программирования с динамической типизацией, оказал большое влияние на развитие таких объектно-ориентированных языков, как Java, Ruby, Objective-C. Многие революционные идеи появились в сообществе Smalltalk. К ним можно отнести рефакторинг, шаблоны проектирования (применительно к ПО), карты «класс – обязанности – взаимодействие» и экстремальное программирование в целом. GemStone была одной из первых коммерчески доступных ООСУБД.

Проект корпорации MCC (Microelectronics and Computer Technology Corporation) ознаменован появлением таких СУБД, как Itasca [10] и UniSQL [11; 12]. Во французском исследовательском центре INRIA [1] осуществлялся проект Altair, в рамках которого была создана O2. Компанией «Hewlett Packard» была разработана IRIS, DEC выпустили систему Trellis. В виде завершенных коммерческих систем появились G-Base, упомянутая ранее, Gemstone, Statice, Vbase.

К первым СУБД, реализующим принципы объектно-ориентированного подхода, Сергей Дмитриевич Кузнецов сформулировал два основных замечания. Первое сводится к невозможности практического применения указанного спектра систем ввиду таких характеристик, как ненадежность, отсутствие поддержки безопасности данных, низкая производительность и др. Второе замечание касается отсутствия единого языка, коим, по мнению Сергея Дмит-

[1] Inria, a major actor in the European and international research arena. INRIA: официальный сайт. URL: https://www.inria.fr/

риевича, должен был быть C++. Необходимость изучать новый язык, а у каждой такой СУБД он был свой, понижала коммерческий интерес [5].

Дальнейшее развитие методологии проектирования баз данных находило отражение в так называемых «Манифестах». Первый манифест был издан в 1989 г. Его авторами были преимущественно ученые [13], это была первая попытка собрать имеющиеся разноплановые наработки в области применения ОО-подхода к проектированию БД в единое определение ООСУБД. Был приведен перечень основных характеристик, которыми должна обладать СУБД, чтобы быть классифицированной как ООСУБД. Характеристики подразделялись на три группы: обязательные, необязательные и открытые. Обязательными характеристиками должна обладать любая СУБД, относящаяся к классу объектно-ориентированных, необязательные характеристики могут быть как присущи данным системам, так и нет. Открытые характеристики реализуются разработчиком в соответствии с его собственными соображениями, унификация на этом уровне не предполагается. К обязательным, например, были отнесены такие характеристики, как идентифицируемость объектов, инкапсуляция, классы; множественное наследование – это уже пример необязательной характеристики, к открытым относится, например, парадигма программирования. Будущее развитие СУБД авторы манифеста видели в ООСУБД, поддерживающей традиционные возможности.

Второй манифест появился в 1990 г. [14]. В противовес первому авторы второго манифеста инженеры – разработчики баз данных, ориентированных на применение языка SQL. В данном документе дается классификация СУБД по поколениям: ранее созданные реляционные, иерархические и сетевые СУБД – первое поколение, используемые на текущий момент реляционные СУБД – второе, СУБД третьего поколения – те, что соответствуют предложенным в документе требованиям. Требования включают три принципа и 13 предложений. В соответствии с первым принципом СУБД третьего поколения должны предоставлять средства хранения и манипулирования объектами, структура которых может включать более широкий спектр типов данных, таких как тексты, например, а также средства создания правил для элементов данных, как ссылочная целостность в реляционных БД и более сложные. Второй принцип состоит в необходимости поддержки возможностей СУБД второго поколения. Третий принцип гласит об открытости СУБД, т. е. о поддержке интеграции с прикладными программами, разработанными на различных платформах.

Летом 1991 г. была задумана и впоследствии образована Object Database Management Group (ODMG) [13; 15], в 1998 г. была переименована в Object Data Management Group, чтобы отобразить распространение своей деятельности не только на объектно-ориентированные базы данных, но и технологии ORM (Object-Relational Mapping) [2]. ORM – объектно-реляционное отображение, или преобразование, – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Основной целью ODMG было разработать комплекс спецификаций, которые позволят разрабатывать портируемые приложения для объектной базы данных и ORM-приложений. В период с 1993 по 2001 г. ODMG опубликовал пять изменений в своей спецификации. Последней была ODMG версии 3.0, после чего группа распалась.

Предлагаемая ODMG архитектура показана на рис. 1. В этой архитектуре определяются способ хранения данных и разные виды пользовательского доступа к этому «хранилищу данных». Единое хранилище данных доступно из языка определения данных, языка запросов и ряда языков манипулирования данными. На рис. 1 ODL означает Object Definition Language (язык определения объектов), OQL – Object Query Language (язык объектных запросов) [16] и OML – Object Manipulation Language (язык манипулирования объектами).

[2] Stack Exchange Inc. Is ObjectDB production ready? URL: https://stackoverflow.com/questions/5291950/is-objectdb-production-ready

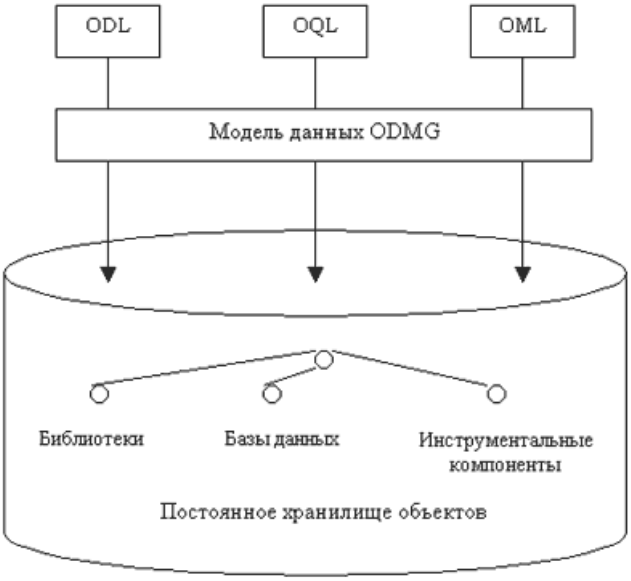


Рис. 1. Архитектура ODMG
Fig. 1. ODMG architecture

После публикации второго манифеста на рынок вышел целый новый класс СУБД, именуемый объектно-реляционными СУБД, среди них Informix Universal Server [17], Oracle 8, DB 2 Universal Database.

В марте 1995 г. была впервые официально опубликована статья Хью Дарвена и Кристофера Дейта, названная авторами «Третьим манифестом» [18]. Авторы статьи ищут прочные основы для будущего управления данными. При этом Хью Дарвен и Кристофер Дейт считают, во-первых, что такие основы не способен обеспечить язык баз данных SQL, во-вторых, любые такие основы должны твердо корениться в реляционной модели данных, впервые представленной миру Э. Ф. Коддом в 1969 г. При этом авторы сознают ценность поддержки горячо обсуждаемых возможностей объектно-ориентированного подхода, называя их «ортогональными реляционной модели». И в силу этого реляционная модель не нуждается в каких-либо расширениях, коррекции или категоризации, а самое главное, в каких-либо искажениях, для того чтобы внедрить указанные возможности в какой-либо язык баз данных, способный представлять те основы, которые мы ищем. Основу Манифеста представляет описание такого языка: «Допустим, что такой язык имеется и называется D». Авторы приводят предписания и запреты языка D. Некоторые из этих предписаний проистекают из реляционной модели данных и названы РМ-предписаниями. Предписания, которые не происходят из реляционной модели, названы остальными ортогональными предписаниями – ОО-предписаниями. Подобным же образом категоризованы и запреты для языка D.

Несмотря на впечатляющее количество проектов, проводимых как коллективами ученых, так и коммерческими компаниями, ориентированными на практическое внедрение, не появилось четкой формулировки объектно-ориентированной модели данных, каждый коллектив представлял свое видение применения принципов ООП к проектированию баз данных.

Отсутствие универсальной модели данных, с проработанным математическим аппаратом (как в случае реляционных баз данных) и сегодня является основной проблемой распространения ООСУБД.

А. М. Эльдарханов предлагает рассматривать проблематику разработки ООСУБД с позиции разделения по трем уровням представления данных. Так же, как и в классической модели структуры базы данных ANSI / SPARC [19; 20], представлены три уровня описания данных.

В обеих названных иерархиях уровни располагаются в порядке убывания степени абстракции. Самый нижний уровень, наиболее приближенный к физическим процессам, реализующим задачи хранения и управления данными, в обеих моделях является физический уровень. Над физическим уровнем А. М. Эльдарханов разместил уровень модели данных, далее следует уровень формальной математической модели [21].

Для создания формальной математической модели существуют два подхода. Первый подход предполагает последовательное исполнение ряда действий с целью получения заданного результата аналогично функциональному программированию и называется функциональным. В литературе представлены два метода данного подхода: алгебра select-project-join (SPJ) и многосортное исчисление предикатов высшего порядка [22]. Язык SQL реализует алгебру SPJ [23].

Второй подход создания формальной математической модели – дедуктивный. Дедуктивный подход основан на использовании логического вывода, запросы к базам данных формулируются в виде аксиом и правил. Кристофер Дейт в своем фундаментальном труде приводил следующий пример (листинг 1, 2)

Листинг 1

GOOD_SUPPLIER (s, st, sc) ⇐ S (s, sn, st, sc) AND st > 15

На листинге 1 представлено определение предиката «GOOD_SUPPLIER» на языке Datalog. На листинге 2 приведены примеры запросов к предикату «GOOD_SUPPLIER».

Листинг 2

1. Определить всех добросовестных поставщиков. ?
   ⇐GOOD_SUPPLIER (s, st, sc)
2. Определить всех добросовестных поставщиков из Парижа. ?
   ⇐ GOOD_SUPPLIER (s, st, Paris)
3. Является ли поставщик si добросовестным? ?
   ⇐ GOOD_SUPPLIER (S1, st, sc)

Средний уровень представления данных – уровень модели данных, представлен двумя аспектами: поведенческим и структурным. Структурный аспект представляет два уровня описания: уровень данных (рис. 2) и уровень схемы (рис. 3).

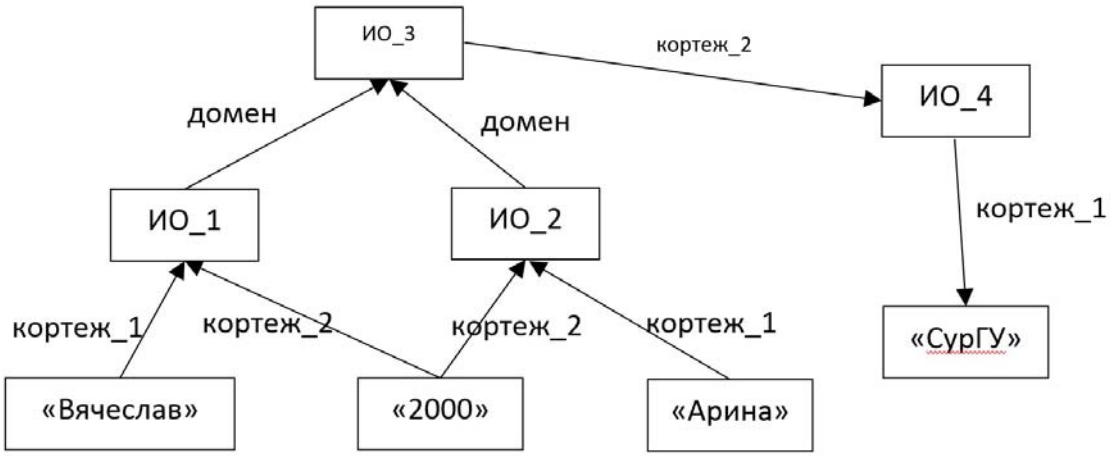


Рис. 2. Описание модели данных на уровне данных. Предметная область: студенты вуза

Fig. 2. Description of the data model at the data level. Subject area: university students

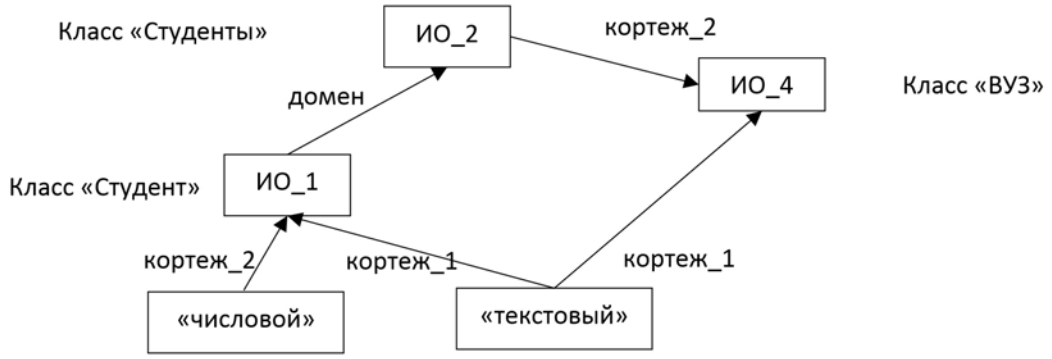


Рис. 3. Описание модели данных на уровне схемы. Предметная область: студенты вуза
Fig. 3. Description of the data model at the schema level. Subject area:university students

Важнейшим понятием объектной модели данных, характерным для любой ООСУБД, является идентификатор объекта. В нашем примере на рис. 1. ИО_1 и ИО_2 – идентификаторы двух объектов, моделирующих 2-х студентов; ИО_4 – идентификатор объекта, моделирующего вуз.

Над объектами в объектной модели данных определяются составные операции. На нашей схеме составные операции именованы русскоязычными аналогами обозначений функций в англоязычной литературе и языках программирования. Операция «домен» (set) создает множество однотипных объектов, как в нашем примере, результатом операции «домен» является новый объект с идентификатором ИО_3, являющийся набором двух студентов с идентификаторами ИО_1 и ИО_2.

Операция «кортеж» (tuple) создает кортеж в общем случае разнородных объектов, номер в обозначениях на схеме, следующий за словом «кортеж» означает, какой это по счету операнд операции составления кортежа.

База данных на рис. 1 представлена графом, вершины которого – объекты, а ребра – отношения вида «быть i-м операндом операции с данным результатом».

На рис. 3. Представлен граф уровня схемы для тех же объектов, что и на рис. 2. В ходе анализа текущего уровня развития ООБД, Эльдарханов приходит к выводам: «Ограничения на корректность объектов теории являются… внешними средствами по отношению к теории, в основном выраженными в форме указания конкретных классов некорректных объектов с обозначением пути разрешения проблемы… важной является также проблема реализации в данной модели сочетания функционального и логического программирования… Полной формализации включения функций и отношений в объектную модель пока не создано» [21].

Несмотря на наличие проблем применения объектно-ориентированного подхода к проектированию баз данных, объектно-ориентированные СУБД являются альтернативными средствами при решении такого класса задач, где применение реляционных СУБД обнаруживает ограниченность модели.

В англоязычной литературе встречается термин «The object-relational impedance mismatch», данный термин применяется для обозначения концептуальных и технических трудностей, возникающих при взаимодействии прикладных программ или комплексов прикладных программ, разработанных на объектно-ориентированных языках, с реляционными базами данных.

К одному из ряда подобных осложнений относится отсутствие возможности в реляционных СУБД работы с агрегатом значений атрибутов сущности как с единым целым экземпля-

ром сущности, моделирующим объект предметной области. Например, информацию об экземпляре сущности «Книга» можно получить из базы данных, схема которой представлена на рис. 4, не обратившись к целостному объекту, а собрав необходимые сведения из соответствующих таблиц посредством конструкции запросов на языке SQL с применением оператора «JOIN».

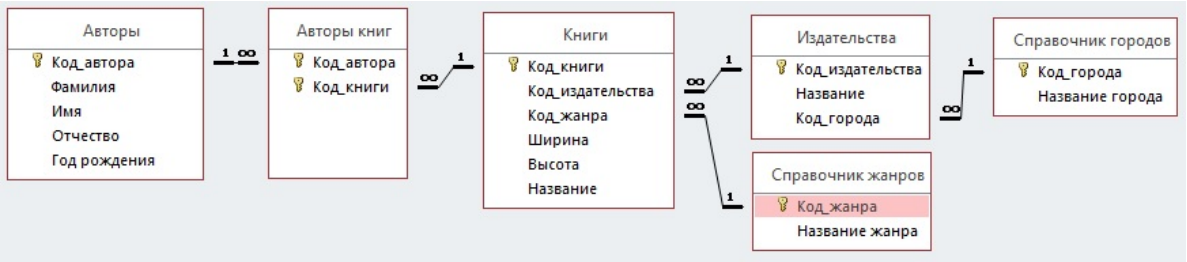


Рис. 4. Схема данных базы «Книги» в реляционной СУБД MS Access
Fig. 4. Scheme of the database “Books”, relational database management system MS Access

Конструкция запроса, которая может быть применена, представлена на листинге 3.

Листинг 3

```
SELECT Авторы.Фамилия, Авторы.Имя, Авторы.Отчество, Книги.Название, [Справочник
жанров].[Название жанра], Издательства.Название, [Справочник городов].[Название го-
рода] FROM [Справочник жанров] INNER JOIN ((([Справочник городов] INNER JOIN Из-
дательства ON [Справочник городов].Код_города = Издательства.Код_города) INNER
JOIN Книги ON Издательства.Код_издательства = Книги.Код_издательства) INNER JOIN
(Авторы INNER JOIN [Авторы книг] ON Авторы.Код_автора = [Авторы
книг].Код_автора) ON Книги.Код_книги = [Авторы книг].Код_книги) ON [Справочник
жанров].Код_жанра = Книги.Код_жанра;
```

Проблемы применения реляционных СУБД наиболее остро ощущаются в таких областях, как системы автоматизированного проектирования, автоматизированное производство, системы автоматизированного управления предприятием, системы, основанные на знаниях, мультимедийные системы.

ООБД позволяют объединить программный код и данные, избежать различий между представлениями информации в базе и прикладной программе, вследствие чего интерес к ним проявляют современные разработчики.

Среди современных объектно-ориентированных СУБД в первую очередь следует упомянуть о Caché [3], продукте компании InterSystems, существует и успешно развивается на протяжении 21-го года. ООСУБД Cache используется при проектировании автоматизированных систем управления здравоохранением, банковских и финансовых услуг, государственного управления и других секторов.

Вызывает интерес ConceptBase – дедуктивная и объектно-ориентированная система управления базами данных, разработанная в Университете Аахена и Университете Скуве. ConceptBase используется для концептуального моделирования и метамоделирования в области разработки программного обеспечения [24].

[3] Inter Systems: официальный сайт. URL: https://www.intersystems.com/products/cache/

В связи с активным развитием глобальных сетей и как следствие технологий разработки кросс-платформенных приложений возрос интерес к системе GemStone, в частности языку Smalltalk как инструменту создания JavaScript, реализующих задачи коммерческих приложений и веб-страниц. GemStone/S на сегодняшний день является одним из лучших примеров ООСУБД, которая хорошо подходит для масштабируемых, высокопроизводительных, многоуровневых распределенных систем [25].

Примером современной встраиваемой ООСУБД является ObjectDatabase++ (ODBPP), разработчик – Ekky Software [4]. В данной системе для хранения информации об экземпляре книги будет использоваться одна запись в противопоставление рассмотренному выше примеру структурирования данной информации в реляционной СУБД. Такой способ представления данных позволяет оперировать целым объектом и уменьшает необходимость объединения данных из разных таблиц; сокращает количество требуемых операций блокировки, чтения и записи, что способствует эффективной работоспособности системы при больших объемах данных.

ObjectDB – это объектная СУБД для Java. Она может использоваться в режиме клиент-сервер и во встроенном (в процессе) режиме. В то время как основные СУБД, представленные на рынке, предоставляют API (application program interface). Например, в состав MySQL входит библиотека, написанная на языке C, позволяющая обращаться к данным из любой программы, разработанной на этом языке [26]. ObjectDB не предоставляет собственный, для работы с этой СУБД используется один из двух стандартных API Java: Java Persistence API (JPA) или Java Data Objects (JDO). Спецификация JPA предоставляет возможность сохранять в удобном виде Java-объекты в базе данных [5]. Объекты данных Java (JDO) [27] представляют собой классы языка программирования Java; для них нет необходимости реализовывать определенные интерфейсы или расширяться из специальных классов. Оба API-интерфейса встроены в ObjectDB [6], поэтому промежуточное программное обеспечение ORM не требуется [7] [28].

ObjectDB является кросс-платформенной и может использоваться в различных операционных системах с Java SE 5 или выше. Он может быть интегрирован в веб-приложения Java EE и Spring и развернут в контейнерах сервлетов (Tomcat, Jetty), а также на серверах приложений Java EE (GlassFish, JBoss) [29]. Она была протестирована на различных JVM, включая HotSpot, JRockit и IBM J9 [8].

Максимальный размер базы данных составляет 128 ТБ (131 072 ГБ). Количество объектов в базе данных неограниченно (кроме размера базы данных).

Все устойчивые типы JPA и JDO поддерживаются ObjectDB, включая определяемые пользователем классы сущностей, определяемые пользователем встраиваемые классы, стандартные коллекции Java, базовые типы данных (примитивные значения, значения обертки, String, Date, Time, Timestamp) и любые другие сериализуемые классы.

Каждый объект в базе данных имеет уникальный идентификатор. ObjectDB поддерживает как традиционные идентификаторы базы данных объектов, так и механизмы реляционных баз данных, такие как первичные ключи, включая составные первичные ключи и создание и присвоение автоматического значения как часть поддержки JPA, которая в основном представляет собой API для РСУБД.

Поддерживаются два языка запросов. Язык запросов JDO (JDOQL), основанный на синтаксисе Java, и язык запросов JPA (JPQL), основанный на синтаксисе SQL. Также поддерживаются запросы по критериям JPA 2.

[4] Eccy Software. URL: http://www.ekkysoftware.com/

[5] Java Persistence API FAQ. Oracle Corporation. URL: https://www.oracle.com/technetwork/java/javaee/persistence-jsp-136066.html

[6] ObjectDB Software. URL: http://www.objectdb.com/

[7] JPA Performance Benchmark. URL: http://www.jpab.org

[8] ObjectDB. Object Database Features. URL: http://www.objectdb.com/object/db/database/features

Автоматическая эволюция схемы ObjectDB обрабатывает большинство изменений классов прозрачно, включая добавление и удаление постоянных полей, изменение типов постоянных полей и изменение иерархии классов. Также поддерживается переименование устойчивых классов и постоянных полей.

Современный рынок программного инструментария представлен рядом ООСУБД, но, если посмотреть на рейтинги используемых СУБД, как, например, результаты ежегодного опроса, проводимого сайтом «Stack Overflow» [9], где в 2018 г. приняли участие свыше 100 000 разработчиков программного обеспечения (см. таблицу), ни одну ООСУБД мы не встретим.

Самые популярные СУБД, по версии опроса «Stack Overflow»
The most popular DBMSs, according to the Stack Overflow survey

№ п/п	Название системы организации хранения данных	Количество опрошенных, применяющих данный инструментарий, %
1	MySQL	58,7
2	SQL Server	41,2
3	PostgreSQL	32,9
4	MongoDB	25,9
5	SQLite	19,7
6	Redis	18,0
7	Elasticsearch	14,1
8	MariaDB	13,4
9	Oracle	11,1
10	Microsoft Azure (Tables, CosmosDB, SQL, etc)	7,9
11	Google Cloud Storage	5,5
12	Memcached	5,5

В качестве итога можно привести слова Ричарда Энга: «Хотя ООСУБД и не получили широкого распространения, всё же объектные базы данных имеют свои нишевые рынки» [25].

Список литературы

1. **Mitchell G., Zdonik S. B.** Umeshwar Dayal. Optimization of Object-Oriented Queries: Problems and Approachers. *Advances in Object-Oriented Database Systems. Nato ASI Subseries F*, 2013, vol. 130, p. 119–146.
2. **Hudson S. E., King R.** The Efficient Support of Functionally-Defined Data in Cactis. In: Dittrich K. R., Dayal U., Buchmann A. P. On Object-Oriented Database Systems. Springer Science & Business Media, 2012, p. 341–355.
3. **Liskov B., Day M., Shrira L.** Distributed Object Management in Thor. In: Ozsuaet T. et al. (eds.). Distributed Object Management. San Mateo, CA, Morgan Kaufmann, 1994, p. 79–91.
4. **Thearle R. W.** A survey of object oriented database systems. In: Tagg R., Mabon J. (eds.). Object Management. Routledge, 2019.
5. **Кузнецов С. Д.** Три манифеста баз данных: ретроспектива и перспективы // Базы данных и информационные технологии XXI века: Докл. Междунар. науч. конф., 2003. URL: http://citforum.ru/database/articles/manifests/

[9] Developer Survey Results. 2018. URL:https://insights.stackoverflow.com/survey/2018/

6. **Stein J., Maier D.** Associative Support in GemStone. In: Dittrich K. R., Dayal U., Buchmann A. P. (eds.). On Object-Oriented Database Systems. Springer Science & Business Media, 2012, p. 323–338.
7. **Prabhu C. S. R.** Object-Oriented Database Systems: Approaches and Architectures. PHI Learning Pvt. Ltd., 2005, 232 p.
8. **Meuter W. de.** (ed.). Advances in Smalltalk: 14th International Smaltalk Conference, ISC 2006. Prague, Springer Science & Business Media, 2007, 158 p.
9. **Liu Ch.** Smalltalk, Objects, and Design. iUniverse, 2000, 312 p.
10. **Salmons J., Babitsky T.** International OOP Directory. COOT, Incorporated, 1992.
11. **D'Andrea A., Janus P.** UniSQL's next-generation object-relational database management system. *ACM SIGMOD Record*, 1996, vol. 25, no. 3 (September).
12. **Кузнецов С. Д.** Объектно-реляционные базы данных: прошедший этап или недооцененные возможности? URL: http://citforum.ru/database/articles/ordbms10/4.shtml
13. **Thearle R. W.** A survey of object oriented database systems. In: Tagg R., Mabon J. (eds.). Object Management. Routledge, 2019.
14. **Hugh D., Date C. J.** The Third Manifesto. *ACM SIGMOD Record*, 1995, vol. 24, no. 1 (March).
15. **Barry D. K., Duhl J.** Object Storage Fact Books: Object DBMSs and Object-Relational Mapping. Barry & Associates, Inc., 2001.
16. **Surhone L. M., Tennoe M. T., Henssonow S. F.** Object Query Language. URL: https://www.books.ru/books/object-query-language-1467950/ © Books.ru
17. **Галатенко В., Гвоздев А.** Типы и структуры данных в INFORMIX-Universal Server. Изд-во «Открытые системы», 1997. URL: https://www.osp.ru/news/articles/1997/0602/13031546
18. **Дейт К., Дарвен Х.** Основы будущих систем баз данных. Третий манифест. М.: Янус-К, 2004.
19. ANSI/X3/SPARC Study Group on Data Base Management Systems: Interim Report. FDT. *ACM SIGMOD bulletin*, 1975, vol. 7, no. 2.
20. **Jardine D. A.** The ANSI/SPARC DBMS Model. North-Holland Pub. Co., 1977.
21. **Эльдарханов А. М.** Обзор моделей данных объектно-ориентированных СУБД // Тр. ИСП РАН. 2011. Т. 21. С. 205–226.
22. **Beeri C.** A formal approach to object-oriented databases. *Data & Knowledge Engineering*, 1990, vol. 5, p. 353–382.
23. **Дейт К. Дж.** Введение в системы баз данных: Пер. с англ. 8-е изд. М.: Вильямс, 2005. 1328 с.: ил.
24. **Jeusfeld M.** ConceptBase.cc – A System for Metamodeling and Method Engineering. URL: http://conceptbase.sourceforge.net/
25. **Eng R. K.** How learning Smalltalk can improve your skills as a programmer. 2016. URL: https://medium.com/smalltalk-talk/how-learning-smalltalk-can-improve-your-skills-as-a-programmer-fc5b2f56685
26. **Дюбуа П.** MySQL. 3-е изд. М.: Вильямс, 2006. 344 с.
27. **Jordan D., Russell C**. Java Data Objects. 1st ed. O'Reilly Media, 2003, April 22, 384 p.
28. **Srinivasan K.** Create applications using ObjectDb and JPA in NetBeans. URL: http://www.javabeat.net/2011/02/create-applications-using-objectdb-and-jpa-in-netbeans
29. **Anghel L.** Integrate ObjectDB into Your JPA-based Java Web App. URL: http://www. developer.com/java/web/integrate-objectdb-into-your-jpa-based-java-web-app.html

References

1. **Mitchell G., Zdonik S. B.** Umeshwar Dayal. Optimization of Object-Oriented Queries: Problems and Approachers. *Advances in Object-Oriented Database Systems. Nato ASI Subseries F*, 2013, vol. 130, p. 119–146.

2. **Hudson S. E., King R.** The Efficient Support of Functionally-Defined Data in Cactis. In: Dittrich K. R., Dayal U., Buchmann A. P. On Object-Oriented Database Systems. Springer Science & Business Media, 2012, p. 341–355.
3. **Liskov B., Day M., Shrira L.** Distributed Object Management in Thor. In: Ozsuaet T. et al. (eds.). Distributed Object Management. San Mateo, CA, Morgan Kaufmann, 1994, p. 79–91.
4. **Thearle R. W.** A survey of object oriented database systems. In: Tagg R., Mabon J. (eds.). Object Management. Routledge, 2019.
5. **Kuznetsov S. D.** Three database manifest: retrospective and perspective. In: Databases and information technologies XXI century: report of the international scientific conference, 2003. URL: http://citforum.ru/database/articles/manifests/
6. **Stein J., Maier D.** Associative Support in GemStone. In: Dittrich K. R., Dayal U., Buchmann A. P. (eds.). On Object-Oriented Database Systems. Springer Science & Business Media, 2012, p. 323–338.
7. **Prabhu C. S. R.** Object-Oriented Database Systems: Approaches and Architectures. PHI Learning Pvt. Ltd., 2005, 232 p.
8. **Meuter W. de.** (ed.). Advances in Smalltalk: 14th International Smaltalk Conference, ISC 2006. Prague, Springer Science & Business Media, 2007, 158 p.
9. **Liu Ch.** Smalltalk, Objects, and Design. iUniverse, 2000, 312 p.
10. **Salmons J., Babitsky T.** International OOP Directory. COOT, Incorporated, 1992.
11. **D'Andrea A., Janus P.** UniSQL's next-generation object-relational database management system. *ACM SIGMOD Record*, 1996, vol. 25, no. 3 (September).
12. **Kuznetsov S. D.**Object-relational databases: past stage or underestimated opportunities? URL: http://citforum.ru/database/articles/ordbms10/4.shtml
13. **Thearle R. W.** A survey of object oriented database systems. In: Tagg R., Mabon J. (eds.). Object Management. Routledge, 2019.
14. **Hugh D., Date C. J.** The Third Manifesto. *ACM SIGMOD Record*, 1995, vol. 24, no. 1 (March).
15. **Barry D. K., Duhl J.** Object Storage Fact Books: Object DBMSs and Object-Relational Mapping. Barry & Associates, Inc., 2001.
16. **Surhone L. M., Tennoe M. T., Henssonow S. F.** Object Query Language. URL: https://www.books.ru/books/object-query-language-1467950
17. **Galatenko V., Gvozdev A.** Types and data structures in INFORMIX-Universal server.Open System Publishing House, 1997. URL: https://www.osp.ru/news/articles/1997/0602/ 13031546
18. **Date C. J., Hugh Darwen**. Databases, Types, and the Relational Model. The Third Manifesto. Addison Wesley; 3 edition (February 24, 2006).
19. ANSI/X3/SPARC Study Group on Data Base Management Systems: Interim Report. FDT. *ACM SIGMOD bulletin*, 1975, vol. 7, no. 2.
20. **Jardine D. A.** The ANSI/SPARC DBMS Model. North-Holland Pub. Co., 1977.
21. **Eldarkhanov A. M.** An Overview of Data Models of Object-Oriented DBMS. In: Proceedings of the Institute for System Programming RAS. 2011, vol. 21, p. 205–226.
22. **Beeri C.** A formal approach to object-oriented databases. *Data & Knowledge Engineering*, 1990, vol. 5, p. 353–382.
23. **Date C. J.** An Introduction to Database Systems. Publisher: Pearson; 8 ed. (August 1, 2003).
24. **Jeusfeld M.** ConceptBase.cc – A System for Metamodeling and Method Engineering. URL
25. **Jeusfeld M.** ConceptBase.cc – A System for Metamodeling and Method Engineering. URL: http://conceptbase.sourceforge.net/
26. **Eng R. K.** How learning Smalltalk can improve your skills as a programmer. 2016. URL: https://medium.com/smalltalk-talk/how-learning-smalltalk-can-improve-your-skills-as-a-programmer-fc5b2f56685
27. **Jordan D., Russell C**. Java Data Objects. 1st ed. O'Reilly Media, 2003, April 22, 384 p.
28. **Jordan D., Russell C**. Java Data Objects. 1st ed. O'Reilly Media, 2003, April 22, 384 p.

29. **Srinivasan K.** Create applications using ObjectDb and JPA in NetBeans. URL: http://www.javabeat.net/2011/02/create-applications-using-objectdb-and-jpa-in-netbeans
30. **Anghel L.** Integrate ObjectDB into Your JPA-based Java Web App. URL: http://www. developer.com/java/web/integrate-objectdb-into-your-jpa-based-java-web-app.html

Материал поступил в редколлегию
Received
21.05.2019

Сведения об авторе / Information about the Author

Яценко Елена Александровна, кандидат технических наук, доцент кафедры автоматизированных систем обработки информации и управления, Политехнический институт, Сургутский государственный университет (ул. Энергетиков, 22, Сургут, Россия)

Elena A. Yatsenko, Candidate of Science (Technology), Associate Professor of Automated information processing and management systems, Polytechnic Institute, Surgut State University (22 Energetikov Str., Surgut, Russian Federation)

elisia@yandex.ru
ORCID iD 0000-0002-3239-4409

Правила оформления текста рукописи

Авторы представляют статьи на русском или английском языке объемом от 0,5 авторского листа (20 тыс. знаков) до 1 авторского листа (40 тыс. знаков), включая иллюстрации (1 иллюстрация форматом 190 × 270 мм = 1/6 авторского листа, или 6,7 тыс. знаков). Публикации, превышающие указанный объем, допускаются к рассмотрению только после индивидуального согласования с редакцией журнала.

Текст рукописи должен быть представлен в редколлегию в виде файла MS Word (.doc, .docx). Гарнитура Times New Roman, размер шрифта 11, межстрочный интервал 1, размеры полей – стандартные значения текстового редактора. Форматирование – выравнивание по ширине страницы, переносы слов включены, каждый новый абзац начинается с красной строки. Не допускается ручное форматирование абзацев (пробелами, лишними переводами строк, разрывами страниц).

Структура статьи

- Индекс УДК (универсальной десятичной классификации). Выравнивание по левому краю
- Название статьи. Выравнивание по центру, полужирный шрифт
- ФИО авторов (инициалы, фамилия). Выравнивание по центру, полужирный шрифт
- Места работы всех авторов. Выравнивание по центру, курсив
- Аннотация статьи
- Ключевые слова, не более 10
- Благодарности, сведения о финансовой поддержке
- Название статьи **на английском языке**. Выравнивание по центру, полужирный шрифт
- ФИО авторов **на английском языке** (инициалы, фамилия). Выравнивание по центру, полужирный шрифт
- Места работы авторов **на английском языке**. Выравнивание по центру, курсив
- Аннотация статьи **на английском языке (Abstract)**, 200–250 слов
- Ключевые слова **на английском языке (Keywords)**, не более 10
- Благодарности, сведения о финансовой поддержке **на английском языке**, если есть соответствующий раздел на русском языке **(Acknowledgements)**
- Основной текст
- Список литературы / **References**
- Сведения об авторах

Требования к оформлению основного текста и иллюстративных материалов

Основной текст должен быть представлен в структурированном виде, рекомендуется использовать подзаголовки – например: Введение, Методика..., Выводы, Результаты, Заключение.

Подзаголовки отделяются и набираются полужирным шрифтом. В целях выделения частей текста и отдельных слов и словосочетаний допускается использование курсива или полужирного шрифта. Подчеркивание, разрядка, изменение основного кегля и выделение цветом не используются.

Иллюстрации к рукописи статьи должны быть приложены в виде отдельных файлов. При этом в тексте должно содержаться включенное изображение с указанием имени файла. Все иллюстрации, содержащие схемы, графики, алгоритмы и т. п., должны быть представлены

в векторном виде (.ai, .eps, .cdr). Скриншоты и другие растровые изображения должны быть представлены в максимально высоком качестве, без каких-либо потерь и искажений (.jpg, .tif). Все иллюстрации должны иметь подрисуночную подпись – свое название. Надписи к таблицам и подписи к иллюстрациям приводятся **на двух языках (русском и английском)**.

Примеры:

Рис. 1. Диаграмма производительности...
Fig. 1. Performance diagram...

Таблица 1

Сравнение алгоритмов...

Table 1

Comparison of algorithms...

Нумерация последовательная и неразрывная от начала статьи. Не допускается использование других наименований, кроме «Рис.» / «Fig.», «Таблица» / «Table», и усложнение нумерации (например, «Рис. 3.2.»). Ссылка на иллюстрацию в тексте должна быть приведена в круглых скобках, например: (рис. 1), (табл. 1).

Формулы должны быть набраны с использованием редактора MathType либо встроенного редактора формул MS Word. Кегль основных символов – 11, греческие символы набираются прямым шрифтом, латинские – курсивом. Нумеруются только те формулы, на которые автор ссылается в тексте.

Abstract

Аннотация статьи на английском языке (Abstract) не должна быть дословным переводом русскоязычной аннотации. Раздел Abstract, как и основной текст, должен быть структурирован, в нем должно содержаться описание цели работы, методов исследования, научной значимости, выводов / результатов. Требуется качественный перевод на английский язык (при необходимости просим авторов обращаться к профессиональным переводчикам). **Объем Abstract 200–250 слов.**

Список литературы / References

Список литературы и список литературы на английском языке (References) размещаются в общем разделе. Рекомендуемое количество цитируемых в статье источников – не менее 10, в список желательно включать ссылки на актуальные работы по теме исследования, особенно в иностранных периодических изданиях.

В тексте статьи ссылки на литературу указываются цифрами в квадратных скобках, при необходимости указываются номера страниц, например: [2; 3. С. 15].

Список литературы нумеруется в порядке цитирования и оформляется в соответствии с ГОСТ Р 7.0.5-2008 на библиографическое описание (знаки тире в описании опускаются). Ссылки на неопубликованные работы, а также на Интернет-ресурсы (кроме электронных изданий, поддающихся библиографическому описанию) оформляются в виде сноски.

В Список литературы ссылки на источники следует включать на оригинальном языке опубликования. Каждый источник должен быть также оформлен на английском языке (References) по международному стандарту для публикаций в области информатики IEEE Style со следующими отличиями:

- инициалы авторов указываются после фамилии;
- название статьи не берется в кавычки, отделяется точкой;
- отсутствует союз «and» перед фамилией последнего автора;
- в диапазоне страниц указывается одна «p» (вместо «pp. 2–9» – «p. 2–9»);
- год издания указывается после места издания (для книг) и сразу после названия журнала (для периодики).

Перевод источника на английский язык:

- если источник имеет выходные данные на английском языке, то для формирования References **следует использовать именно эти данные**;

- если оригинальная публикация не содержит выходных данных на английском языке, то допускается транслитерация названия материала на латинский алфавит в сочетании с переводом на английский язык в квадратных скобках. В конце описания указывается, на каком языке написана эта работа, например, (in Russ.). При транслитерации можно воспользоваться Интернет-ресурсом http://ru.translit.ru/, рекомендуется выбрать стандарт BSI. Место издания не транслитерируется, указывается полностью на английском языке, например: Moscow. Название издательства / издателя, как правило, транслитерируется. Для журналов, у которых есть официальное название на английском языке, – использовать его (проверить на сайте журнала, или, например, в библиотеке WorldCat), если названия на английском языке нет, использовать транслитерацию по системе BSI. Не следует самостоятельно переводить названия журналов.

Если у цитируемого источника есть **цифровой идентификатор DOI** (https://search.crossref.org), его требуется обязательно указывать в конце библиографической ссылки.

Примеры оформления ссылок. Каждый источник в том же пункте дублируется на английском языке (References).

Источник на русском языке, перевод на английский доступен в метаданных статьи

1. **Журавлев С. С., Рудометов С. В., Окольнишников В. В., Шакиров С. Р.** Применение модельно-ориентированного проектирования к созданию АСУ ТП опасных промышленных объектов // Вестник НГУ. Серия: Информационные технологии. 2018. Т. 16, № 4. С. 56–67. DOI 10.25205/1818-7900-2018-16-4-56-67

Zhuravlev S. S., Rudometov S. V., Okolnishnikov V. V., Shakirov S. R. Model-Based Design Approach for Development Process Control Systems of Hazardous Industrial Facilities. *Vestnik NSU. Series: Information Technologies*, 2018, vol. 16, no. 4, p. 56–67. (in Russ.) DOI 10.25205/1818-7900-2018-16-4-56-67

Источник на английском языке. Оформляем согласно требованиям для References. Приводим только 1 раз.

2. **Telnov V. I.** Optimization of the Beam Crossing Angle at the ILC for E + e- and yy Collisions. *Journal of Instrumentation*, 2018, vol. 13, no. 03, p. P03020–P03020. DOI 10.1088/1748-0221/13/03/p03020

Метаданные источника доступны только на русском языке

3. **Жижимов О. Л., Федотов А. М., Шокин Ю. И.** Технологическая платформа массовой интеграции гетерогенных данных // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, вып. 1. С. 24–41ю

Zhizhimov O. L., Fedotov A. M., Shokin Yu. I. Tekhnologicheskaya platforma massovoi integratsii geterogennykh dannykh [Technology Platform For the Mass Integration of Heterogeneous Data]. *Vestnik NSU. Series: Information Technologies*, 2013, vol. 11, no. 1, p. 24–41. (in Russ.)

Сведения об авторах

Последний раздел статьи – информация об авторе / авторах **на русском и английском языках**:

- ФИО полностью, ученая степень, ученое звание, должность, место работы, адрес места работы
- e-mail
- идентификаторы автора, такие как ORCID или ResearcherID (всем авторам рекомендуется использовать данные сервисы для ведения актуального списка своих публикаций)
- контактный телефон (не публикуется)

Если статья представляется на английском языке, необходимо приложить перевод на русский язык названия, аннотации, ключевых слов, сведений об авторе.

Доставка материалов

Материалы предоставляются в редакцию по электронной почте inftech@vestnik.nsu.ru.

Порядок рецензирования

Все статьи сначала проходят проверку на заимствование и только после этого отправляются на рецензирование. Редакционный совет не допускает к публикации материал, если имеется достаточно оснований полагать, что он является плагиатом.

Тип рецензирования статей – двухуровневое, одностороннее анонимное («слепое»).

Для каждой статьи редколлегией выбираются рецензенты, научная деятельность которых связана с темой представленного материала. Ответственный секретарь журнала обращается к ним с просьбой дать экспертную оценку статье либо помочь организовать рецензирование.

Рецензии для журнала «Вестник НГУ. Серия: Информационные технологии» составляются по единой схеме и подразумевают оценку по следующим критериям: соответствие тематике журнала, оригинальность и значимость результатов, качество изложения материала.

Заполненный бланк рецензии высылается на электронный адрес редакции. В зависимости от экспертных заключений статья может быть принята редакционным советом к опубликованию, рекомендована автору к доработке (с последующим повторным рецензированием либо без него) или отклонена (с предоставлением автору мотивированного отказа). Автору на электронный адрес высылается текст рецензии без указания ФИО рецензента и его контактных данных.

Все рецензии хранятся в редакции журнала не менее 5 лет. Редколлегия журнала обязуется при поступлении соответствующего запроса направлять копии рецензий в Министерство науки и высшего образования Российской Федерации.