# Глава 1 Литературный обзор

Данная глава последовательно даёт обзор источников информации из области современных баз данных, фреймворков (программных каркасов для построения информационных систем) и кластерного анализа данных, в качестве которых рассматриваются литература, электронные документы, компьютерные программы.

Активное развитие сферы информационных технологий значительно отразилось и на посвящённом ей литературном фонде: практически всегда можно найти несколько книг от авторитетных изданий на любую тематику. В то же время видна тенденция перехода формата обучения от печатных источников к видеоурокам и интерактивным обучающим курсам из-за более наглядного представления конечного результата и возможности сразу же приступить к выполнению практических заданий.

Также необходимо отметить быстрое устаревание источников информации на тему информационных технологий: к актуальным можно отнести источники за последние 5-10 лет. Темп развития отрасли также сказывается и на объёме источников информации: значительная часть востребованных знаний представлена в виде коротких статей на веб-ресурсах, ссылаться на которые невозможно по ряду объективных причин.

Источники информации на тему кластерного анализа данных представлены немногочисленными книгами, больше информации можно найти в формате статей. Зачастую вопрос кластеризации данных раскрывается в источниках на смежную тематику.

## 1.1 Текущее состояние области кластерного анализа данных

Термин «кластерный анализ» предложен К. Трионом в 1939 г. (англ. cluster — гроздь, скопление, пучок). Синонимами могут выступать следующие выражения: автоматическая классификация, таксономия, распознавание без обучения, распознавание образов без учителя, самообучение и др. [Мандель Кластерный анализ].

Кластер-анализ — это способ группировки многомерных объектов, основанный на представлении результатов отдельных наблюдений точками подходящего геометрического пространства с последующим выделением групп как «сгустков» этих точек [Мандель Кластерный анализ].

Кластерный анализ используется в различных областях (экономике, биологии, информатике, психологии и т. д.), что обусловило параллельное развитие алгоритмов кластеризации и разное их наименование при одинаковых принципах работы. Тем не менее цель всех алгоритмов кластеризации — выделение устойчивых групп элементов с какими-либо схожими характеристиками [Суслов].

Кластерный анализ, как один из основных методов обучения без учителя, изучает структуру данных из неизвестной области и сам по себе является предметом для исследования. Однако однозначного определения кластеризации до сих пор не существует, его традиционное понимание включает в себя следующие положения [1]:

1) объекты в одном и том же кластеры должны быть сходными, насколько это возможно;

2) объекты, находящиеся в разных кластерах, должны отличаться друг от друга, насколько это возможно;

3) меры сходства или различия должны быть понятными и иметь практическое значение.

Стандартный процесс кластеризации может быть разделён на следующие шаги [2]:

1) предварительный анализ данных: поиск и извлечение основных особенностей предоставленного набора данных;

2) проектирование кластерного алгоритма: разработка кластерного алгоритма в соответствии со свойствами решаемой проблемы;

3) анализ результатов: оценка полученных результатов и пригодности алгоритма;

4) подведение итогов: практическое объяснение полученного разбиения на кластеры.

### 1.1.1 Расстояния и близость

Расстояние (различие) или сходство – основные понятия кластерного анализа. При обработке количественных характеристик данных предпочтительнее использовать расстояния для выделения взаимосвязей между данными [3].

Наиболее популярные расстояния для количественных данных перечислены в таблице 1.

Таблица 1 – Функции расстояний

|  |  |  |
| --- | --- | --- |
| Название | Формула | Пояснение |
| Расстояние Минковского |  | Зависит от значения n:  1) при n=1 становится расстоянием городских кварталов;  2) при n=2 становится евклидовом расстоянием;  3) при становится расстоянием Чебышёва. |
| Стандартизированное евклидово расстояние |  | 1) s – стандартное отклонение;  2) взвешенное евклидово расстояние, основанное на отклонении. |
| Косинусовое расстояние |  | 1) остаётся одинаковым при любом порядке данных;  2) часто применяется при кластеризации документов. |
| Коэффициент корреляции Пирсона |  | 1) Cov – ковариантность, D – дисперсия;  2) измеряет расстояния, основанные на линейной корреляции. |
| Расстояние Махаланобиса |  | 1) S – ковариционная матрица внутри кластера;  2) высокая сложность вычисления. |

Наиболее популярные расстояния для качественных данных перечислены в таблице 2.

Таблица 2 – Функции близости

|  |  |  |
| --- | --- | --- |
| Название | Формула или метод | Пояснение |
| Индекс Жаккара |  | 1) измеряет сходство двух наборов данных;  2) расстояние Жаккара = 1 – индекс Жаккара |
| Мера сходства Хемминга | Минимальное число изменений, необходимых для превращения одного объекта в другой | Чем меньше число, тем больше сходство.  Расстояние Хемминга противоположно мере сходства Хемминга, особенно для текстовых данных. |
| Для данных смешанного типа | Перевести данные в двоичный формат и считать параметр дихотомным |  |

**Оценочные показатели**

Основная цель оценочного показателя – отобразить пригодность кластерного алгоритма для поставленной задачи. Оценочные показатели могут быть разделены на две группы: внутренние и внешние.

Внутренние оценочные показатели принимают внутренние данные для тестирования алгоритма. Тем не менее они не могут однозначно утверждать, какой алгоритм лучше, когда оценки двух алгоритмов основываются в неравной степени на внутренних оценочных показателях. Существует три популярных внутренних оценочных показателя, перечисленных в таблице 3 [4].

Таблица 3 – Внутренние оценочные показатели

|  |  |  |
| --- | --- | --- |
| Название | Формула или метод | Пояснение |
| Индекс Дэвиса-Болдина |  | *K* – число кластеров, *Cx* – центр кластера *x*, *σx* – среднее расстояние между любыми данными в кластере *x* и *Cx*, *d(ci, cj)* – расстояние между *ci* и *cj*. |
| Индекс Данна |  | 1) в основном предназначен для данных, имеющих равномерную плотность и распределение;  2) *d(ci, cj)* – расстояние между *ci* и *cj*, *d'(k)* расстояние в кластере *k*. |
| Индекс силуэта | Оценивает результаты кластеризации на основе среднего расстояния между объектов и другими объектами одного кластера и среднего расстояния между разными кластерами. |  |

Внешние оценочные показатели, пользующиеся большей популярностью по сравнению с внутренними, принимают данные извне для проверки пригодности алгоритма. Тем не менее недавние исследования показали, что внешние оценочные показатели не всегда корректны. Существует шесть популярных внешних показателей, которые перечислены в таблице 4 [5].

Таблица 4 – Внешние оценочные показатели

|  |  |  |
| --- | --- | --- |
| Название | Формула или метод | Пояснение |
| Индекс Рэнда |  | 1) TP – число истинных положительных значений;  2) TN – число истинных отрицательных значений;  3) FP – число ложных положительных значений;  4) FN – число ложных отрицательных значений. |
| F-индекс |  | 1) – точность, – степень совпадения;  2) FP, FN, TP, TN определены выше. |
| Мера Жаккара |  | 1) измеряет сходство двух наборов данных;  2) FP, FN, TP, TN определены выше. |
| Индекс Фаулкса-Мэллоуза |  | FP, FN, TP, TN определены выше. |
| Взаимная информация | Мера из теории информации - как много общей информации содержится в двух кластерах, между которыми может быть обнаружена нелинейная корреляция. |  |
| Матрица несоответствий | Показывает различия между кластером и эталонным кластером. |  |

В последующих разделах, особенно касающихся анализа сложности алгоритмов, *n* обозначает число всех объектов, *k* обозначает число кластеров, *s* обозначает число выборочных объектов, и *t* обозначает число итераций.

### 1.1.2 Традиционная классификация кластерных алгоритмов

Традиционные алгоритмы кластеризации могут быть разделены на 9 видов, которые содержат 26 часто используемых алгоритмов и перечислены в таблице 5.

Таблица 5 – Традиционная классификация алгоритмов

|  |  |
| --- | --- |
| Вид | Типичные представители |
| Итеративные кластерные алгоритмы | K-means, K-medoids, CLARA, CLARANS, PAM |
| Иерархические кластерные алгоритмы | Chameleon, BIRCH, ROCK, CURE |
| Нечёткие кластерные алгоритмы | MM, FCS, FCM |
| Распределительные кластерные алгоритмы | GMM, DBCLASD |
| Плотностные кластерные алгоритмы | OPTICS, DBSCAN, Mean-shif |
| Графовые кластерные алгоритмы | MST, CLICK |
| Сеточные кластерные алгоритмы | CLIQUE, STING |
| Фрактальные кластерные алгоритмы | FC |
| Модельные кластерные алгоритмы | GMM, SOM, COBWEB, ART |

Итеративные кластерные алгоритмы

Основная идея данного вида кластерных алгоритмов – рассматривать центры скоплений объектов как центры соответствующих кластеров. K-means и K-medoids являются двумя наиболее популярными представителями этой группы алгоритмов. Главная идея алгоритма K-means – обновлять центр кластера, который представлен центром скопления объектов, итеративным вычислением, пока оно не будет остановлено условием сходимости. K-medoids – это улучшенный алгоритм K-means для работы с дискретными данными, который принимает объект, наиболее близкий к центру объектов, как типичного представителя соответствующего кластера. В качестве других алгоритмов, принадлежащих этой группе, можно назвать CLARA, CLARANS и PAM [6].

К преимуществам данной группы алгоритмов можно отнести относительно низкую временную сложность и высокую вычислительную эффективность в целом.

Недостатки: не подходят для обработки невыпуклых данных, относительно чувствительны к выбросам, необходимо задавать число кластеров, результат кластеризации чувствителен к числу кластеров.

Временная сложность итеративных алгоритмов кластеризации представлена в таблице 6.

Таблица 6 – Временная сложность итеративных алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K-means | K-medoids | CLARA | CLARANS | PAM |
|  |  |  |  |  |
| Низкая | Высокая | Средняя | Высокая | Высокая |

Иерархические кластерные алгоритмы

Основная идея данного вида алгоритмов – построить иерархические отношения между данными. Предположим, что сначала каждый объект является кластером, после этого два соседствующих кластера объединяются в новый, пока не останется только один кластер – так работают иерархические агломеративные алгоритмы. По обратному принципу работают дивизимные алгоритмы – сначала всё является одним кластером, который затем последовательно разделяется на другие кластеры. Типичными представителями данной группы являются алгоритмы CURE, BIRCH, Chameleon, ROCK. BIRCH выполняет кластерный анализ путём построения дерева характеристик кластеров, в котором каждый узел обозначает подкластер. Такое дерево динамически изменяется при появлении нового объекта. CURE подходит для крупномасштабного кластерного анализа и использует семплинг-метод для разбиения кластеров. ROCK является улучшением алгоритма CURE для работы с данными перечислимого типа, которое учитывает сходство с данными вокруг кластера. Chameleon изначально разбивает данные на кластеры небольшого размера на основе графа ближайшего соседа, после чего кластеры объединяются в более крупные [7].

Преимущества: подходят для данных произвольной формы и параметров, иерархические отношения между объектами легко обнаруживаются, в целом относительно высокие возможности масштабирования.

Недостатки: относительно высокая временная сложность, нужно задавать число кластеров.

Временная сложность иерархических алгоритмов кластеризации представлена в таблице 7.

Таблица 7 – Временная сложность иерархических алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
| CURE | BIRCH | ROCK | Chameleon |
|  |  |  |  |
| Высокая | Низкая | Высокая | Высокая |

Нечёткие кластерные алгоритмы

Основная идея данного вида алгоритмов – преобразовать дискретные значения {0, 1} в непрерывные [0, 1] с целью описать отношения между объектами более достоверно. Типичными представителями данной группы являются алгоритмы MM, FCS и FCM. Главная идея FCM – получить степень схождения каждого объекта в каждый кластер путём оптимизирования функции объекта. FCS, отличающийся от традиционных нечётких кластерных алгоритмов, принимает многоразмерную гиперсферу как прототип каждого кластера с целью разбивать на кластеры на основе функции расстояния гиперсферы. FCS имеет высокую временную сложность из-за применения ядер в алгоритме. MM, основанный на анизотропной мере (англ. mountain function), используются для нахождения центра кластера [8].

Преимущества: более реалистичны в плане отображения принадлежности к кластерам, относительно высокая точность кластеризации.

Недостатки: относительно слабые возможности масштабирования, результат кластеризации чувствителен к изначально заданным параметрам, необходимо задавать число кластеров.

Временная сложность нечётких алгоритмов кластеризации представлена в таблице 8.

Таблица 8 – Временная сложность нечётких алгоритмов

|  |  |  |
| --- | --- | --- |
| FCS | FCM | MM |
| (ядро) |  |  |
| Высокая | Низкая | Высокая |

Распределительные кластерные алгоритмы

Основная идея: данные, полученные из одного распределения, принадлежат одному и тому же кластеру, если существует несколько распределений в изначальных данных. Типичными представителями данной группы являются алгоритмы DBCLASD и GMM. Главная идея DBCLASD, динамического инкрементального алгоритма, - если расстояние между кластером и его ближайшими объектами удовлетворяет распределению ожидаемого расстояния, которое вычисляется из существующих объектов этого кластера, то ближайший объект должен принадлежать данному кластеру. Главная идея алгоритма GMM: GMM состоит из нескольких нормальных распределений, из которых получены изначальные данные, а данные, подчиняющиеся тому же независимому нормальному распределению, должны принадлежать тому же кластеру [9].

Преимущества: более реалистичны в плане отображения принадлежности к кластерам, относительно высокая масштабируемость путём изменения распределения, числа кластеров и т. д.

Недостатки: включают в себя множество параметров, имеющих сильное влияние на результаты кластеризации, относительно высокая временная сложность.

Временная сложность распределительных алгоритмов кластеризации представлена в таблице 9.

Таблица 9 – Временная сложность распределительных алгоритмов

|  |  |
| --- | --- |
| DBCLASD | GMM |
|  |  |
| Средняя | Высокая |

Плотностные кластерные алгоритмы

Основная идея данного вида алгоритмов: данные, которые находятся в области с высокой плотностью данных, принадлежат одному и тому же кластеру. Типичными представителями являются DBSCAN, Mean-shift и OPTICS. DBSCAN – наиболее популярный плотностный алгоритм кластеризации, принцип работы которого соответствует основной идее данной группы алгоритмов. OPTICS является доработкой DBSCAN и преодолевает один из его недостатков – DBSCAN чувствителен к параметрам радиуса соседства и минимального числа объектов в соседстве. Алгоритм Mean-shift работает следующим образом: сначала вычисляется среднее значения отклонения текущего объекта, после следующий объект считается основанным на текущем объекте и отклонении, и такие итерации будут продолжаться до тех пор, пока не будет выполнено какое-то условие. Временная сложность Mean-shift высока из использования ядра в алгоритме [10].

Преимущества: высокая эффективность кластеризации, подходят для данных произвольной формы.

Недостатки: низкие качества кластеризации, когда плотность пространства данных неравномерна, необходимы мощные вычислительные ресурсы, результат кластеризации очень чувствителен в отношении параметров.

Временная сложность плотностных алгоритмов кластеризации представлена в таблице 10.

Таблица 10 – Временная сложность плотностных алгоритмов

|  |  |  |
| --- | --- | --- |
| DBSCAN | OPTICS | Mean-shift |
|  |  | (ядро) |
| Средняя | Средняя | Высокая |

Графовые кластерные алгоритмы

Согласно данному типу кластерных алгоритмов, кластеризация осуществляется на графе, где узел – это объект, а ребро отношения между объектами. Типичными представителями являются алгоритмы CLICK и MST-кластеризация. Основная идея алгоритмы CLICK – выполнить минимальное по весу разделение графа итерациями с целью разбиения на кластеры. Создание минимального остовного дерева на основе графа данных является ключевым шагом для осуществления кластерного анализа для MST-метода [11].

Преимущества: кластеризация высокой эффективности и точности.

Недостатки: существенно повышающаяся временная сложность при усложнении графа.

Временная сложность графовых алгоритмов кластеризации представлена в таблице 11, где *v* означает количество вершин, *e* обозначает количество рёбер, *f(y, e)* – означает временную сложность вычисления минимального разбиения.

Таблица 11 – Временная сложность графовых алгоритмов

|  |  |
| --- | --- |
| CLICK | MST |
|  |  |
| Низкая | Средняя |

Сеточные кластерные алгоритмы

Основная идея данного вида кластерных алгоритмов: изначальное пространство данных преобразуется в сеточную структуру определённого размера. Типичными представителями данной группы являются алгоритмы STING и CLIQUE. Главная идея алгоритма STING, который может быть использован для параллельных вычислений, - пространство данных разделено на множество прямоугольных областей путём создания иерархической структуры, и данные внутри каждого структурного уровня соответствующе кластеризуются. CLIQUE сочетает в себе преимущества сеточных и плотностных алгоритмов кластеризации [12].

Преимущества: низкая временная сложность, высокая масштабируемость, хорошо подходят для параллельных вычислений и инкрементальных обновлений.

Недостатки: результат кластеризации чувствителен к степени детализации (размеру ячеек), высокая вычислительная эффективность ценой снижения качества кластеризации.

Временная сложность сеточных алгоритмов кластеризации представлена в таблице 12.

Таблица 12 – Временная сложность сеточных алгоритмов

|  |  |
| --- | --- |
| STING | CLIQUE |
|  |  |
| Низкая | Низкая |

Фрактальные кластерные алгоритмы

Фрактал обозначает имеющий свойство самоподобия объект, который может быть разделён на несколько частей, чем-то похожих на целое. Типичным представителем данной группы является алгоритм FC, основная идея которого заключается в том, что изменение любых внутренних данных кластера не влияет на внутреннее качество фрактальной размерности. Временная сложность алгоритма FC равна [13].

Преимущества: кластеризация высокой эффективности и масштабируемости, эффективная обработка выбросов, подходят для обработки данных произвольной формы и высокой размерности.

Недостатки: результат кластеризации чувствителен к параметрам.

Модельные кластерные алгоритмы

Основная идея – выбрать определённую модель для каждого кластера и найти наилучшее соответствие для неё. Можно назвать две основных группы модельных алгоритмов: один из них основан на метод статистического обучения, другой – на методе обучения нейронных сетей.

Типичными представителями группы, опирающейся на метод статистического обучения, являются алгоритмы COBWEB и GMM [14]. Основная идея COBWEB – построить дерево классификации, основанное на каких-то эвристических критериях, с целью создать иерархическую кластеризацию на предположении, что распределение вероятностей каждого атрибута является независимым. Временная сложность COBWEB в основном низкая, она зависит от распределения, используемого в алгоритме.

Типичными представителями алгоритмов, основанных на обучении нейронных сетей, являются SOM и ART [14]. Ключевая идея SOM – построить отображение уменьшения размерности из входного пространства высокой размерности в пространство вывода малой размерности на предположении, что во входных данных существует топология. Временная сложность SOM чаще всего высока, она зависит от структуры слоёв нейронной сети. Основная идея ART, инкрементального алгоритма, - динамически создавать новый нейрон, соответствующий новому шаблону для создания нового кластера, когда существующих нейронов недостаточно. Временная сложность ART обычно средняя, она зависит от типа ART и структуры слоёв нейронной сети.

Преимущества: разнообразная и хорошо разработанная модель, обеспечивающая средства для адекватного описания данных, каждая модель имеет собственные особенности, которые могут являться значительными преимуществами в специфичных областях.

Недостатки: в целом относительно высокая временная сложность, результат кластеризации чувствителен к параметрам.

Временная сложность иерархических алгоритмов кластеризации представлена в таблице 13.

Таблица 13 – Временная сложность иерархических алгоритмов

|  |  |  |
| --- | --- | --- |
| COBWEB | SOM | ART |
| (распределение) | (слои) | (тип+слои) |
| Низкая | Высокая | Средняя |

### 1.1.3 Современная классификация кластерных алгоритмов

Современные алгоритмы кластеризации могут быть разделены на 10 групп, которые содержат 45 часто используемых алгоритмов, что представлено в таблице 14.

Таблица 14 – Современная классификация кластерных алгоритмов

|  |  |
| --- | --- |
| Группа алгоритмов | Представители |
| Ядровые кластерные алгоритмы | kernel K-means, kernel SOM, kernel, FCM, SVC, MMC, MKC |
| Интегрированные кластерные алгоритмы | CSPA, HGPA, MCLA, VM, HCE, LAC, WPCK, sCSPA, sMCLA, sHBGPA |
| Роевые кластерные алгоритмы | ACO\_based(LF), PSO\_based, SFLA\_based, ABC\_based |
| Квантовые кластерные алгоритмы | QC, DQC |
| Спектральные кластерные алгоритмы | SM, NJW |
| Метод распространения близости | AP |
| Кластерные алгоритмы на основе плотности и расстояний | DD |
| Кластерные алгоритмы для пространственных данных | DBSCAN, STING, Wavecluster, CLARANS |
| Кластерные алгоритмы для потоковых данных | STREAM, CluStream, HPStream, DenStream |
| Кластерные алгоритмы для больших данных | K-means, BIRCH, CLARA, CURE, DBSCAN, DENCLUE, Wavecluster, FC |

Ядровые кластерные алгоритмы

Ключевая идея данного типа кластерных алгоритмов в том, что данные входного пространства преобразуются в пространство признаков высокой размерности с помощью нелинейного отображения. Типичными представителями данной группы являются алгоритмы kernel K-means, kernelFCM, kernelSOM, MMC, SVC и MKC. Основная идея алгоритмов kernel K-means, kernelSOM и kernelFCM – совместить преимущества ядрового метода с оригинальным кластерным алгоритмом, преобразуя изначальные данные в многоразмерное пространство признаков с помощью нелинейное ядровой функции. Ключевая идея алгоритма SVC – найти сферу с минимальным радиусом, которая может покрыть все объекты в высокоразмерном пространстве признаков, после развернуть сферу обратно в изначальное пространство данных для образования изолинии, а именно – границы кластеров, и данные, находящиеся в замкнутой изолинией области, должны принадлежать одному кластеру. MMC пытается найти гиперплоскость с максимальным допуском к кластеру. MKC, доработка MMC, пытается найти лучшую гиперплоскость, основываясь на нескольких ядрах у кластера. Общим у MMC и MKC является ограничение вычислений по степени [15].

Все представители данной группы алгоритмов имеют высокую временную сложность.

Преимущества: более простая кластеризация в высокоразмерном пространстве признаков, подходят для данных произвольной формы, способны распознавать шум и разграничивать пересекающиеся кластеры, нет необходимости в предварительном изучении структуры кластеризуемых данных.

Недостатки: результат кластеризации чувствителен к типу ядра и его параметрам, высокая временная сложность, не подходят для данных большого размера.

Интегрированные кластерные алгоритмы

Интегрированные кластерные алгоритмы названы так потому, что их идей является вычислить первоначальное разбиение на кластеры каким-то одним определённым методом, а после обработать полученное разбиение другим способом.

Можно назвать четыре основных группы методов для получения первоначального разбиения:

1) для того же набора данных, осуществить кластеризацию тем же алгоритмом, но с другими параметрами или начальными условиями;

2) для того же набора данных, применить разные алгоритмы;

3) для подмножеств, осуществить кластеризацию по отдельности;

4) для того же набора данных, выполнить кластеризацию в другом пространстве признаков на основе других ядер [16].

Первоначальное разбиение по кластерам затем интегрируется с помощью функций согласованности, среди которых можно выделить 6 групп, перечисленных в таблице 15.

Таблица 15 – Функции согласованности

|  |  |
| --- | --- |
| Название | Типичный алгоритм или приложение |
| Основанная на разделении графа | CSPA, HGPA and MCLA |
| Основанная на переименовании и голосовании | VM |
| Основанная на генетическом алгоритме | HCE |
| Основанная на локальной адаптации | LAC |
| Основанная на ядровом методе | WPCK |
| Основанная на нечётких множествах | sCSPA, sMCLA и sHBGPA |

Временная сложность данного вида алгоритмов основана специфичных методах и алгоритмах, включённых в алгоритм.

Преимущества: устойчивы к нарушениям исходных предпосылок, масштабируемы, могут быть распараллелены, используют преимущества интегрированных алгоритмов.

Недостатки: неочевидные представления о качестве первоначальных разбиений, существующие недостатки в работе функций согласованности.

Роевые кластерные алгоритмы

Основная идея данного вида алгоритмов состоит в симуляции процесса изменения биологической популяции. Можно выделить четыре типичных группы алгоритмов: ACO\_based, PSO\_based, ABC\_based, SFLA\_based. Основная идея LF, типичного алгоритма ACO\_based, состоит в том, что данные распределяются случайно по двумерной сетке, после данные либо выбираются, либо нет для дальнейшей обработки, основанной на решении муравья, и этот процесс зациклен до тех пор, пока не будет достигнут приемлемый результат кластеризации. PSO\_based алгоритмы считают объект как частицу. Первоначальное разбиение кластеров частиц получается другим кластерным алгоритмом, после чего кластеры частиц непрерывно обновляются на основе центров кластеров, положения и скорости каждой частицы до тех пор, пока не будет достигнут приемлемый результат кластеризации. Основная идея SFLA\_based алгоритмов – симулировать обмен информацией между лягушек и воспользоваться преимуществом локального поиска и глобального взаимодействия информации. Основная идея ABC\_based алгоритмов – симулировать поведение трёх типов пчёл при поиске пищи, задача которых – найти источник пищи, в пчелином улье и применить локальный и глобальный обмен информацией для кластеризации [17].

Временная сложность алгоритмов этой группы является высокой, в основном из-за большого количества итераций.

Преимущества: простой для понимания алгоритм.

Недостатки: низкая масштабируемость, низкая операционная эффективность, не подходят для высокоразмерных данных или данных большого размера.

Квантовые кластерные алгоритмы

Идея квантовых кластерных алгоритмов заключается в изучении закона распределения выборочных данных в масштабированном пространстве путём изучения закона распределения частиц в энергетическом поле. Типичными алгоритмами данной группы можно считать QC и DQC [18]. Ключевая идея алгоритма QC (англ. quantum clustering – квантовая кластеризация), способного обрабатывать высокоразмерные данные, - вычислить потенциальную энергию каждого объекта с помощью уравнения Шрёдингера, используя алгоритм итеративно нисходящего градиента, считая объект с низкой потенциальной энергией как центр кластера, и поместить объекты в разные кластеры на основе определённой функции расстояния. Временная сложность QC высока, так как процесс решения уравнения Шрёдингера может занять большое количество итераций. DQC, улучшение QC, применяет зависящее от времени уравнение Шрёдингера с целью динамически изучить изменения изначального набора данных и структуры функции потенциальной квантовой энергии. Временная сложность DQC, более практичного по сравнению с DC, является средней.

Преимущества: небольшое число параметров, определение центра кластера основано на возможной информации выборочных данных.

Недостатки: результат кластеризации чувствителен к параметрам алгоритма, модель алгоритмов не способна полностью описать закон изменения данных.

Спектральные кластерные алгоритмы

Основная идея данного вида алгоритмов – считать объект вершиной и сходство между объектами как взвешенные рёбра с целью преобразовать проблему кластеризации в проблему разбиения графа. Задача состоит в поиске метода разбиения графа, делающего вес связей между различными группами как можно меньшим и общий вес связей среди рёбер внутри небольшой группы как можно более высоким. Алгоритмы данного вида могут быть разделены на две группы: рекурсивные спектральные и многоходовые спектральные, и типичными алгоритмами этих групп будут являться SM и NJW соответственно. Ключевая идея алгоритма SM, который обычно используется для сегментации изображений, - минимизировать нормализованный разрез с помощью эвристического метода, основанного на собственном векторе. Временная сложность SM высока из-за вычисления собственных векторов и задействованного эвристического метода. NJW выполняет кластерный анализ в пространстве признаков, построенного собственными векторами, соответствующих k наибольшим собственным векторам матрицы Лапласа. Временная сложность NJW высока из-за вычисления собственных векторов [19].

Преимущества: пригодны для наборов данных произвольной формы и высокой размерности, стремятся к глобальному оптимальному, для ввода требуется только матрица расстояний, не чувствительны к выбросам.

Недостатки: результат кластеризации чувствителен к параметру масштабирования, относительно высокая вычислительная сложность, неопределённость в построении матрицы расстояний, выбор собственных векторов не оптимизирован, нужно задавать число кластеров.

Метод распространения близости

Метод распространения близости (англ. Affinity Propagation, AP) - многообещающий алгоритм, который был разработан в 2007 году. Ключевая идея AP – считать все объекты как потенциальные центры кластеров, а отрицательные значения евклидова расстояния между двумя объектами как близость. Тогда сумма близостей одного объекта к другим объектам больше, вероятность объекта быть центром кластера – выше. AP-алгоритм применяет жадную стратегию, которая максимизирует значения глобальной функции кластерной сети во время каждой итерации [20]. Вычислительная сложность данного метода равна .

Преимущества: простая и наглядная идея алгоритма, нечувствителен к выбросам, не нужно задавать число кластеров.

Недостатки: высокая временная сложность, не пригоден для очень больших наборов данных, результат кластеризации зависит от параметров, включённых в AP-алгоритм.

Кластерные алгоритмы на основе плотности и расстояний

DD (англ. Density and Distance – плотность и расстояние) – ещё один примечательный алгоритм кластеризации, предложенный в 2014 году [21]. Основная особенность DD в описании центра кластера:

1) с высокой локальной плотностью: число объектов около центра кластера внутри определённой области должна быть достаточно большой;

2) вдали от других точек данных с высокой локальной плотностью: центр кластера должен быть удалён от других точек данных, которые могут быть центром кластера.

Ключевая идея DD состоит в том, чтобы на основе функции расстояния вычислить локальную плотность каждого объекта и кратчайшее расстояние между каждым объектом и другими объектами с высокой локальной плотностью с целью сначала построить дерево принятия решений, после выбрать центры кластеров на основе дерева, и поместить оставшиеся объекты в ближайшие кластеры с высокой локальной плотностью.

Временная сложность алгоритма DD равна .

Преимущества: простая и наглядная идея алгоритма, пригодная для наборов данных произвольной формы, нечувствителен к выбросам.

Недостатки: относительно высокая временная сложность, относительно высокая субъективность при выборе центров кластеров на основе дерева принятия решений, результат кластеризации чувствителен к параметрам, использующимся в алгоритме.

Кластерные алгоритмы для пространственных данных

Пространственные данные обозначают данные с двумя измерениями, временем и пространством, в то же время характеризующиеся большим масштабом, высокой скоростью и сложной структурой. Типичными алгоритмами данной группы являются DBSCAN, Wavecluste, STING и CLARANS [22]. Ключевой идеей алгоритма Wavecluster, который может быть использован в параллельных вычислениях, является выполнение кластеризации в новом пространстве признаков с помощью применения вейвлет-преобразований по отношению к исходным данным. Ключевой идеей алгоритма CLARANS является выборка на основе алгоритма CLARA и выполнение кластеризации с помощью алгоритма PAM.

Временная сложность кластерных алгоритмов для пространственных данных представлена в таблице 16.

Таблица 16 – Временная сложность пространственных алгоритмов

|  |  |
| --- | --- |
| Wavecluster | CLARANS |
|  |  |
| Низкая | Высокая |

Кластерные алгоритмы для потоковых данных

Поток данных характеризуется последовательным поступлением, большим размером и ограниченной частотой чтения. Типичными алгоритмами данной группы можно назвать CluStream, STREAM, DenStream, HPStream. STREAM, основанный на идее «разделяй и властвуй», постепенно обрабатывает данные согласованно с последовательностью данных, поступающей с целью построения иерархической структуры. CluStream, который в основном исправляет недостаток STREAM, который описывает исходные только статично, работает с данными как с динамично изменяющимся процессом. Таким образом, CluStream может не только вовремя выдать ответ на запрос, но также предоставляет результат кластеризации с нужной степенью детализированности по времени путём вычисления микро-кластеров онлайн и офлайн. HPStream, доработка CluStream, учитывает ослабление влияния данных с течением времени и более пригоден для кластеризации данных высокой размерности. DenStream, берущий за основу идею алгоритма кластеризации на основе плотности, подходит для невыпуклого набора данных и может эффективно справляться с выбросами по сравнению с алгоритмами, упомянутыми выше [23].

Временная сложность кластерных алгоритмов для потоков данных представлена в таблице 17.

Таблица 17 – Временная сложность алгоритмов для потоков данных

|  |  |  |  |
| --- | --- | --- | --- |
| STREAM | HPStream | CluStream | DenStream |
|  | (онлайн и офлайн обработка) | | |
| Низкая | Низкая | | |

Кластерные алгоритмы для больших данных

Большие данные характеризуются так называемыми 4V свойствами: велики по объёму (volume), богаты на разнообразие (variety), высоки по скорости прироста (velocity) и сомнительны в своей достоверности (veracity) [24]. Основные идеи по кластеризации больших данных могут быть объединены в 4 группы:

1) выборочная-кластеризация;

2) кластеризация объединённых данных;

3) кластеризация со снижением размерности;

4) параллельная кластеризация.

Типичными алгоритмами данной группы можно назвать K-means, BIRCH, CLARA, CURE, DBSCAN, DENCLUE, Wavecluster and FC.

## 1.2 Обзор существующих программных комплексов для осуществления кластерного анализа

Основная цель данной работы – предоставить анализ существующих программных комплексов, в функциональность которых входит какой-либо вид кластерного анализа, выделить их особенности, преимущества и недостатки. Программные комплексы отбираются на основе их популярности и доступности для рядового пользователя.

**RStudio** – интегрированная среда разработки с открытым исходным кодом для языка программирования R, предназначенная для статистической обработки данных и работы с графикой. Язык программирования R по умолчанию включает в себя минималистичный графический интерфейс, RStudio добавляет следующие возможности: как консоль для выполнения команд, продвинутый редактор кода, средства отладки, браузер объектов, интеграция с документацией, средства для управления графиками [Verzani].

Язык программирования R использует менеджер пакетов CRAN, поэтому пользователю предоставляется на выбор около сотни пакетов, предназначенных для работы с кластерным анализом. Среди них представлены методы иерархического (AGNES, CLARA - пакеты cluster, genie, hybridHclust, isopam, protoclust и др.), вероятностного (KMeans, KCentroid, пакеты kmeans, cluster, apcluster, clustMixType и др.), модельно-ориентированного (EM-алгоритм, алгоритмы на основе Байесовской оценки решений) и другие.

Преимущества RStudio: open-source, многообразие пакетов, кроссплатформенность, результат напрямую зависит от навыков пользователя.

Недостатки RStudio: сложность освоения – нужно знать язык программирования R и изучать документацию пакетов.

**RapidMiner** – программный комплекс, который представляет собой платформу для работы с данными не посредством программирования, а управления блоками, обозначающими ту или иную операцию. Блоки называются «операторами», имеют вход и выход, называемые «портами». На вход принимаются данные, на выход поступают обработанные данные [Крахалёв].

RapidMiner поддерживает следующие виды операторов, осуществляющих кластерный анализ: K-means, X-means, K-medoids, DBSCAN, EM-алгоритм, аггломеративная иерархическая кластеризация, случайная кластеризация.

Преимущества RapidMiner: open-source, кроссплатформенность, удобный и наглядный конструктор операторов.

Недостатки RapidMiner: не поддерживается русский язык.

**ClusterEnG** – веб-ресурс, предоставляющий возможность осуществить кластерный анализ с графическим отображением результатов. ClusterEnG поддерживает следующие методы кластеризации: K-means, K-medoids, Affinity Propagation, Mixture model, спектральная и иерархическая кластеризация [Manjunath].

Для того чтобы осуществить кластерный анализ с помощью ClusterEnG, необходимо зайти на сайт проекта, загрузить файл с данными (максимальный объём – 1Гб), выбрать нужный алгоритм и указать параметры кластеризации (число кластеров или эпсилон), если того требует алгоритм.

Преимущества ClusterEnG: находится в свободном доступе, можно произвести кластерный анализ онлайн.

Недостатки ClusterEnG: данные передаются по сети – при невысокой скорости соединения и большом объёме данных будут сложности, не поддерживается русский язык, небольшие возможности настройки работы алгоритмов.

**Weka** – программный комплекс, созданный для решения задач машинного обучения (machine learning). Weka использует пакетный менеджер, позволяющий подключать алгоритмы для обработки данных различных категорий (классификация, предобработка, регрессия, кластеризация и другие) на различных языках программирования (R, Python, Clojure) при поддержке популярных технологий (JDBC, Cassandra, Hadoop, Spark).

Осуществление кластерного анализа в Weka происходит путём выбора файла с данными, выбора алгоритма кластеризации и его настройки.

Преимущества Weka: open-source, кроссплатформенность, многообразие пакетов и поддерживаемых технологий.

Недостатки Weka: не поддерживается русский язык.

**NCSS** – программный комплекс, включающий в свою функциональность около 500 инструментов для работы с графиками и диаграммами и интеллектуального анализа данных. Работа с NCSS очень похожа на работу с редактором электронных таблиц (например, Excel): по умолчанию данные представляются в виде электронной таблицы, затем пользователь выбирает в меню метод обработки данных, настраивает его параметры в появившемся окне, и программа демонстрирует результат в виде отчёта.

NCSS поддерживает следующие виды и методы кластеризации: нечёткая, регрессионная, иерархическая аггломеративная кластеризация, K-means, K-medoids.

Преимущества NCSS: огромное количество инструментов для обработки данных, простота освоения.

Недостатки NCSS: платная лицензия (free trial – 30 дней), не поддерживается русский язык.

**SPSS Statistics** – программный комплекс от компании IBM, схожий по своим возможностям с комплексом NCSS. Данные также представляются в виде электронной таблицы, на выбор предоставляется около сотни инструментов для работы с данными. Кластерный анализ осуществляется аналогично. SPSS поддерживает иерархическую и двухэтапную кластеризацию и метод K-means.

Преимущества SPSS: простота освоения, поддерживается русский язык.

Недостатки SPSS: запутанная документация, платная лицензия (free trial – 14 дней).

**Deductor** – разработанный в России программный комплекс, предлагающий продвинутую аналитику без программирования, схожий по концепции с RapidMiner (однако продукты различаются в способе импорта данных). Рабочий процесс в Deductor заключается в импорте файла с данными и последующем выборе и настройки метода обработки данных, результат сразу можно представить в различных способах представления.

Deductor поддерживает методы K-means и G-means, CLOPE, EM-алгоритм, самоорганизующиеся карты Кохонена.

Преимущества Deductor: простота освоения, поддерживается русский язык, поддержка редких методов кластеризации.

Недостатки Deductor: бесплатная версия ограничена в возможностях.

**Statistica** - программный комплекс от компании Dell, ориентированный на решение задач бизнес-аналитики: анализ, управление, добыча данных. Statistica может быть интегрирована с языками программирования R и Python.

Аналогично SPSS, Statistica поддерживает иерархическую и двухэтапную кластеризацию и метод K-means.

Преимущества STATISTICA: множество поддерживаемых форматов импорта, большое разнообразие инструментов для анализа и визуализации.

Недостатки STATISTICA: не поддерживается русский язык, платная лицензия (free trial – 30 дней).

**Orange** - программный комплекс, предоставляющий возможность обработки данных с помощью функциональных блоков без применения программирования, что делает его схожим с комплексом RapidMiner, кластерный анализ осуществляется аналогично. Orange поддерживает подключение скриптов на языке Python.

Преимущества Orange: простота освоения, open-source, кроссплатформенность, удобный и простой конструктор операторов.

Недостатки Orange: не поддерживается русский язык.

Подводя итоги, можно отметить такие программные комплексы, как RapidMiner, Weka и Orange, по большинству параметров не уступающие дорогостоящим бизнес-решениям в лице NCSS и SPSS. Более продвинутым пользователям, обладающим минимальным опытом программирования, можно посоветовать изучить язык программирования R и использовать RStudio для более полного контроля процесса кластерного анализа.

## **1.3 Обзор современных СУБД**

На данный момент выделяют следующие основные группы СУБД:

* реляционные. Наиболее популярные представители: Oracle, MySQL, MS SQL Server, PostgreSQL, DB2, SQLite;
* столбцовые: C-Store;
* семейство столбцов: Cassandra, Hbase, Hypertable;
* «ключ-значение». Redis, Memcached, Riak, Amazon DynamoDB;
* документоориентированные. MongoDB, Couchbase, CouchDB;
* графовые: Neo4j;
* мультимодельные: OrientDB, ArangoDB [Elmasri].

Вкратце рассмотрим основные особенности каждой из групп.

Один из основных идеологов реляционного подхода к базам данных Эдгар Кодд предложил использовать для обработки данных аппарат теории множеств. Он продемонстрировал, что представление данных является совокупностью двумерных таблиц особого вида, называемых в математике «отношением» (англ. relation). Основными понятиями реляционных БД являются сущность, атрибут, первичный и внешний ключ. На практике сущностями являются таблицы, атрибутами — колонки таблиц, а ключи используются для установления отношений между таблицами [Кириллов].

Реляционные БД поддерживают три типа связи между сущностями: один-к-одному, один-ко-многим, многие-ко-многим. Связи между записями разных сущностей не поддерживаются.

Помимо теоретической работы над реляционным подходом к БД, Эдгар Кодд также создал и практический инструмент для работы с отношениями — реляционную алгебру. Каждая операция данной алгебры использует одну или несколько таблиц в качестве операндов и в итоге создаёт новую таблицу [Кириллов].

Столбцовые СУБД возникли вследствие недостатков производительности реляционных СУБД. В отличие от реляционной СУБД, где вся база данных хранится в одном файле, столбцовые СУБД хранят значения столбцов отдельно друг от друга, в разных файлах. Такой подход ограничивает загружаемый объём данных за каждый запрос (в реляционных СУБД записи загружаются со всеми столбцами), что снижает время совершения запросов и занимаемое дисковое пространство. Столбцовые СУБД используют язык запросов SQL [Фаулер].

Следующую группу направлений развития баз данных специалисты относят к так называемым NoSQL-базам данных. Мы не рассматриваем подробно происхождение и значение этого термина (во многом из-за его неоднозначности), отметим его основное отличие — базы данных этой группы не используют SQL в качестве языка запросов по умолчанию (однако могут поддерживать ради совместимости) и привычную табличную структуру представления данных.

Отличительной особенностью баз данных «семейств столбцов» (англ. column-family, тж. wide-column) является распределение данных как на основе строк, так и на основе столбцов. При этом происходит объединение столбцов в группы или в семейства с целью показать, какие столбцы лучше хранить вместе. Базы данных данного типа позволяют каждой строке иметь различную структуру столбцов без каких-либо ограничений со стороны схемы, размер строк также может быть разным. В качестве параллели с базами данных «семейства столбцов» можно привести двумерный массив данных. Преимуществом данного вида БД является возможность легко добавить новые столбцы к существующим строкам; таблица может быть разреженной (множество null-значений) без каких-либо накладных расходов. Структурно БД «семейства столбцов» занимают положение между реляционными СУБД и БД «ключ-значение» [Редмонд].

Хранилища данных «ключ-значение» состоят из простых пар «ключа» и ассоциируемого с ним «значений», которое обычно является массивом данных. Подобные базы данных обеспечивают структуру, которая позволяет хранить и читать значения на основе «ключа». «Ключ» обычно является строкой и во многих отношениях схож с первичным ключом в реляционной БД. Отдельные записи в «значении» не отслеживаются и не различаются, поэтому при необходимости их изменения необходимо обновление всего «значения». Для базы данных «значение» представляет собой произвольный набор байтов, любая обработка которого отводится на использующую БД систему. Единственные операции, которые позволяют осуществлять БД «ключ-значение», - это put (для записи «значения»), get (для чтения «значения») и delete (для удаления пары «ключ-значение»). Операция обновления данных не поддерживается. [Hoffner]

Документоориентированная СУБД хранит данные в виде структурированных документов, обычно в формате XML или JSON. При этом определение «документоориентированная СУБД» не подразумевает какую-либо специфику насчёт модели хранения: документоориентированные СУБД могут выполнять ACID-транзакции или другие функции традиционных реляционных СУБД, хотя популярные документоориентированные обеспечивают относительно скромную транзакционную поддержку.

Документоориентированные базы данных, позволяя описывать данные без использования схемы, возможно, являются золотой серединой между жёсткой схемой реляционных баз данных и свободных от схемы хранилищ «ключ-значение». Сочетание с практикой веб-разработки вылилось в появление JSON-баз данных (MongoDB в частности), которые стали выбором по умолчанию для многих веб-разработчиков [Harrison G. Next Gen.].

Графовая база данных состоит из набора вершин (узлов, сущностей) и граней (связей, отношений, рёбер). Узлы воспринимаются как объекты со свойствами, между которыми моделируются отношения с помощью граней, которые также могут иметь свойства. Отношения имеют направления, на их основе происходит организация узлов, что позволяет единожды записать данные и затем по-разному их интерпретировать [Jordan] [Фаулер NoSQL].

Графовая структура позволяет представить данные в более естественном виде без искажений, как это может произойти в реляционных базах данных, а также применить различные типы графовых алгоритмов к этим данным. Одна из ключевых особенностей графовых БД — возможность обхода графа по его узлам и граням, перемещения от одного узла к другому, следуя направленным отношениям. Эта возможность называется «index free adjacency» (примерно переводится как «смежность без индекса"), смысл которой заключается в поиске прилежащих узлов без использования поиска по индексу, что значительным образом сказывается на производительности [Bruggen].

## 1.4 Обзор современных баз знаний

База знаний в информатике и исследованиях искусственного интеллекта — это особого рода база данных, разработанная для оперирования знаниями. В отличие от базы данных, база знаний подразумевает наличие каких-либо методов, предназначенных для обработки содержащихся в ней данных.

Данные в базе знаний хранятся в соответствии с моделью данных базы знаний и называются структурированными данными. По степени структурированности данные можно разделить на неструктурированные (обычный текст), слабо структурированные (содержащие теги и другие маркеры для отделения семантических элементов, однако не соответствующие модели данных базы знаний) и структурированные.

Чтобы база знаний смогла оперировать определёнными данными, их прежде необходимо структурировать. Задача правильной структуризации данных — одна из самых важных в области инженерии знаний. Однако каждая база знаний имеет собственное решение данной проблемы, что позволяет исследовать несколько подходов к структуризации данных.

Рассматривая современные базы знаний, нужно отметить большое число веб-ресурсов, подходящих под таковое определение, тематика которых посвящена какой-либо узкой области (Приложение А). К самым развитым и популярным из можно отнести следующие: Knowledge Graph, DBpedia, Викиданные, Cyc.

Google Knowledge Graph – это ранее существовавшая база знаний Freebase, интегрированная в поисковой сервис Google. Knowledge Graph является частью поиска и не имеет пользовательского интерфейса, поэтому работа с ним ничем не отличается от стандартного поиска по ключевым словам. В дальнейшем эта база знаний не рассматривается из-за отсутствия специализированного языка запросов, позволяющего извлекать данные.

Cyc (читается «сайк») – проект по созданию объёмной онтологической базы знаний, позволяющей программам решать сложные задачи из области искусственного интеллекта на основе логического вывода и привлечения здравого смысла. Хранит знания в формате утверждений, фактов, правил.

Проект Cyc по своей классификации относится несколько ближе к экспертной системе, чем к базе знаний, так как имеет аппарат логического вывода, однако нами проект Cyc рассматривается из-за того, что имеет те же цели, что и другие базы знаний.

Модель данных Cyc отличается высокой сложностью и строится на основе утверждений наподобие «Всякое дерево является растением» и «Растения смертны». Кроме того, модель данных Cyc содержит дополнительные типы данных: отдельные элементы (#$BillClinton, #$France), совокупности (#$Tree-ThePlant — все деревья), функции истинности, функции.

Язык запросов Cyc называется CycL и основывается на исчислении предикатов наподобие языка программирования Lisp. Пример высказывания на языке:

(#$genls #$Tree-ThePlant #$Plant) - «Все деревья являются растениями»

(#$capitalCity #$France #$Paris) - «Париж — столица Франции»

Пытаясь реализовать как можно больше возможностей, разработчики проекта Cyc создали чрезмерно сложную, перегруженную модель данных, трудную для восприятия и изучения. Проект Cyc также критикуют за сложности масштабируемости, отсутствие метрики производительности машины вывода, нехватку программной документации, закрытый исходный код.

Базы знаний DBpedia и Wikidata рассмотрим совместно, так как они используют схожую модель данных.

**DBpedia** — проект, направленный на извлечение структурированной информации из данных, созданных в рамках проекта Википедия. DBpedia позволяет пользователям запрашивать информацию, основанную на отношениях и свойствах ресурсов Википедии.

Основой модели данных является RDF-формат. RDF-формат подразумевает структурирование данных в так называемые триплеты, состоящие из субъектов, предикатов и объектов (рис. 1).



Рисунок 1 — Данные в виде RDF

Утверждение «небо голубого цвета» в RDF-терминологии можно представить следующим образом: субъект — «небо», предикат — «имеет цвет», объект — «голубой». Множество RDF-утверждений образует ориентированный граф, в котором вершинами являются субъекты и объекты, а рёбра отображают отношения.

RDF предоставляет средства для построения информационных моделей, но не касается семантики описываемого. Взятый в отдельности граф RDF можно понимать только как граф.

Для выражения семантики требуются словари, таксономии и онтологии.

Словарь представляет собой собрание определённых используемых терминов, имеющих одинаковый смысл во всех контекстах.

Таксономия — это словарь иерархически организованных терминов.

Онтология использует предопределённый зарезервированный словарь терминов для определения концепций и отношений между ними для конкретной предметной области. Онтологии можно использовать для выражения семантики терминов словаря, их взаимоотношений и контекстов использования.

Язык запросов DBpedia называется SPARQL и является SQL-подобным языков. Пример запроса:

PREFIX dbprop: <http://dbpedia.org/property/>

PREFIX db: <http://dbpedia.org/resource/>

SELECT ?who ?work ?genre WHERE {

db:Tokyo\_Mew\_Mew dbprop:illustrator ?who .

?work dbprop:author ?who .

OPTIONAL { ?work dbprop:genre ?genre } .

}

Как видно из примера, язык запросов SPARQL нельзя назвать наглядным. Для использования языка запросов пользователю прежде нужно будет изучить основы SQL.

**Wikidata** — совместно редактируемая база знаний, созданная Фондом Викимедиа. Она используется для обеспечения централизованного хранения данных, которые могут содержаться в статьях Википедии — например, интервики-ссылок или статистической информации.

Модель данных Wikidata также имеет в основе формат RDF, однако содержит дополнительные поля для представления данных (рис. 2).

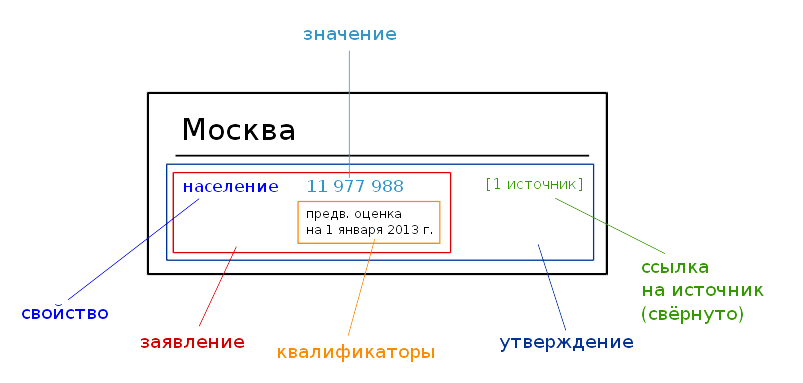


Рисунок 2 — Представление сущности в базе знаний Wikidata

Согласно англоязычной документации, модель данных Wikidata можно представить в виде следующей иерархии:

- **Элемент**, состоящий из:

1. **Идентификатор элемента** (число с префиксом *Q*)
2. **Метка,** состоящая из:
   1. Мультиязычный **заголовок**\*
   2. Мультиязычное **описание**\*
   3. Мультиязычные **псевдонимы**
3. **Утверждения**, каждое состоящее из:
   1. **Заявление**, состоящее из:
      1. Свойство
      2. Значение
      3. Квалификаторы (дополнительные пары свойство-значение)
   2. **Ссылки** (каждая состоит из одной и более пар свойство-значение)
   3. **Ранг**
4. **Ссылки на сайты**

- **Свойство**, состоящее из:

1. **Идентификатор свойства** (число с префиксом *P*)
2. **Метка,** состоящая из:
   1. Мультиязычный **заголовок** \*
   2. Мультиязычное **описание** \*
   3. Мультиязычные **псевдонимы**
3. **Утверждения**, каждое состоящее из:
   1. **Заявление**, состоящее из:
      1. Свойство
      2. Значение
      3. Квалификаторы (дополнительные пары свойство-значение)
   2. **Ссылки** (каждая состоит из одной и более пар свойство-значение)
   3. **Ранг**
4. **Тип данных**

- **Запрос к базе знаний**

Язык запросов базы знаний Wikidata находится в разработке, какие-либо наработки или концепции касательного него широкой публике не представлены, поэтому возможности проанализировать его нет.

Кратко опишем элементы приведённой выше иерархии:

1) сущность может представлять собой элемент, свойство или запрос;

2) элемент — это страница, посвящённая какой-либо теме, концепции или предмету реального мира;

3) свойство — это дескриптор (описатель) для значения, либо некоторого другого отношения или составного (или, возможно, отсутствующего) значения, но не значение или значения как таковые;

4) запрос (эта функция находится в разработке) — это преднастроенный поиск среди элементов Викиданных, представляет собой лишь описание (дескриптор) поиска, но не его результаты;

5) заявление — это отдельный фрагмент данных об элементе, на странице которого содержится это заявление, состоит из свойства (например, «Местоположение») и значения (скажем, «Германия»), может содержать квалификаторы;

6) утверждение — единичный фрагмент данных об элементе, записанный на странице этого элемента, состоит из заявления (т. е. пары свойство-значение, например, «Местоположение: Германия», с возможными квалификаторами), дополненного возможными ссылками (говорящими об источнике заявления) и возможным рангом (используемым для различения между несколькими заявлениями, содержащими одно и то же свойство);

7) значения — это фрагменты информации, содержащиеся в каждом заявлении, могут быть одиночными (как число) или состоять из нескольких частей (как географические координаты с широтой и долготой);

8) квалификатор — это часть заявления, которая сообщает нечто об этом конкретном заявлении, часто в описательной форме. Квалификатор может быть термином в рамках определённого словаря, но также может быть и произвольной описательной фразой.

DBpedia и Wikidata фактически являются очень схожими по модели данных базами знаний. Разработчики и той, и другой базы знаний ставят перед собой задачу структурирования огромного объёма слабоструктурированных данных Википедии, и, что самое важное, стремятся отобразить эти данные в энциклопедическом стиле. В плане представления данных предпочтение стоит отдать базе знаний Wikidata, модель данных которой специализирована под эту цель. Однако и DBpedia, и Wikidata имеют коренной недостаток, происходящий из основы их модели данных.

Рассматривая формат RDF, нужно отметить, что в первую очередь он был предназначен для описания веб-ресурсов сети Интернет, а не сущностей баз знаний. Следует отметить два недостатка формата RDF:

1) оторванность от семантической составляющей — как уже было рассмотрено выше, формат RDF представляет собой лишь структуру, не наполненную смысловой составляющей. Формат RDF нуждается в словарях, таксономии и онтологии, которые, в свою очередь, имеют собственную модель данных, которая не всегда может хорошо сочетаться с представлением в виде RDF-триплетов;

2) подмена семантической составляющей предиката. Рассматривая предыдущий пример триплета («небо голубого цвета»), преобразуем его для наглядности в «небо имеет голубой цвет», с точки зрения семантики RDF эти выражения равны: субъект — «небо», предикат — «имеет цвет», объект — «голубой». В этом выражении предикат является составным — состоит из двух семантических единиц «иметь» и «цвет», которые имеют собственную семантику. Проблема предиката в формате RDF в том, что, объединение составных частей предиката в новую семантическую единицу не позволит учитывать семантику составных компонентов предиката.