Содержание

1 Понятие парадигмы (стиля) программирования. Основные особенности императивного, функционального, логического и объектно-ориентированного программирования.

Парадигма программирования — это набор логически связанных подходов к решению программистских задач вкупе с понятиями и концепциями, характерными для этих подходов. Парадигмы программирования обычно не являются принадлежностью конкретного языка.

Императивное программирование часто рассматривается как наиболее традиционная модель программистского мышления. В основе императивного программирования лежат понятия переменной и присваивания (смены состояния). Понятие императивного программирования часто смешивают с понятием процедурного программирования, характерной особенностью которого является разбиение программы на части (процедуры), вызываемые ради побочного эффекта.
Чисто императивные языки: ранние версии языков **Basic** или **Fortran**.
Процедурные языки: **Pascal**, **C**.

Функциональное программирование предлагает осмысливать программу как систему взаимозависимых функций, вычисляющих значение на основании заданных аргументов. В чистом функциональном программировании побочные эффекты запрещены, что делает его изобразительную мощность недостаточной для создания интерактивных программ. В качестве примера функционального языка часто называют Lisp, однако ни одна версия языка Lisp не была чисто функциональной, хотя программирование на языке Lisp стимулирует мышление в терминах функций. Термин функциональное программирование покрывает такие концепции, как программирование без побочных эффектов, функции как объекты данных, функции высоких порядков (функционалы), ленивые вычисления, карринг, монады, рекурсию и т.д.
Чисто функциональные языки: **Hope** и **Miranda**.

Основанное на исчислении предикатов **логическое программирование** также часто называют одной из парадигм программирования. В логическом программировании программа рассматривается как набор логических фактов и правил вывода, а выполнение программы состоит в вычислении истинности (попытке доказательства) некоторого утверждения. С логическим программированием связывают понятие декларативной семантики, при использовании которой программист задает условия, которым должно удовлетворять решение, а само решение система находит автоматически. Декларативная семантика порождает парадигму декларативного программирования. Декларативная семантика встречается не только в логическом программировании (SQL).
Языки: **Prolog**.

Объектно-ориентированное программирование — это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. Объектно-ориентированный подход позволяет воспринимать программу как набор объектов «черных ящиков», а выполнение программы — как взаимодействие объектов между собой.
Примеры языков: **Smalltalk** (в большей степени), **C++**, **Java**.

2 Соотношение между парадигмами программирования и возможностями языка программирования. Примеры языков, навязывающих, поддерживающих или запрещающих применение той или иной парадигмы.

Парадигмы программирования обычно не являются принадлежностью конкретного языка. Например, ФП: программа воспринимается как набор взаимозависимых функций, не имеющих побочных эффектов, а выполнение программы представляется как вычисление некоторой функции при заданном значении аргументов. Язык **Hope требует** работы именно в таких рамках, выход за рамки ФП невозможен. **Lisp** или **Haskell стимулируют** применение ФП, однако допускают и отступления от него в виде побочных эффектов функций, применения императивных конструкций и т.п. **C** или **Pascal допускают** применение функционального программирования. Наконец, язык **Fortran** делает применение ФП **невозможным**.

Некоторый язык программирования может находиться с определенной парадигмой в одном из следующих вариантов взаимоотношений:

1) язык **навязывает** применение парадигмы. Программирование на данном языке без применения данной парадигмы категорически невозможно.
   Примеры: **Smalltalk и ООП**; **Fortran и присваивания**
2) язык **вынуждает** к применению парадигмы. Программирование на данном языке без применения данной парадигмы возможно, но очень неудобно.
   Примеры: **Lisp и рекурсия**; **Pascal и присваивания**
3) язык **поощряет** применение парадигмы. Программирование на данном языке без применения данной парадигмы возможно и достаточно удобно, однако при освоении данной парадигмы программист получает вознаграждение в виде резко возрастающего удобства работы.
   Примеры: **C++ и механизм исключений**; **Lisp и функции высокого порядка (функционалы)**
4) язык **поддерживает** применение парадигмы. Язык включает в себя специальные средства для применения данной парадигмы, расчитанные на программистов, привыкших к ее применению, однако допускает и другие, в некоторых случаях более удобные варианты решения аналогичных задач.
   Примеры: **C++ и макропроцессирование**; **Lisp и циклы**; **C и рекурсия**
5) язык **допускает** применение парадигмы. Язык не включает никакой специальной поддержки для данной парадигмы, однако при определенных навыках программист все еще может ее применять.
   Примеры: **Pascal и ФП**; **C и обобщенное программирование**; **Pascal и ООП**
6) язык **препятствует** применению парадигмы. Применение данной парадигмы в данном языке теоретически возможно, однако связано с затратами, делающими ее применение неоправданным.
   Примеры: **C и обработка исключений** (возможно с помощью setjump/longjump, но требует серьезных трудозатрат); **Pascal и виртуальные функции**
7) язык **запрещает** применение парадигмы. В данном языке недостаточно средств для применения данной парадигмы.
   Примеры: **Fortran и ФП**; **Hope и императивное программирование**; **Bourne Shell и ООП**

3 Особенности и основные возможности модели ленивых вычислений.

Ленивые вычисления — стратегия вычисления, согласно которой вычисления следует откладывать до тех пор, пока не понадобится их результат. Противоположный подход — энергичные вычисления, вычисляет значение выражения ровно в тот момент, когда оно встретилось в программе. Когда возникает выражение (например, как аргумент вызова функции), вычислитель запоминает его, но вычисления не происходит, пока это не потребуется. Запоминается не только само выражение, но и контекст (например, значения локальных переменных, лексически участвующих в этом выражении), так как не имея контекста, вычислить выражение невозможно. Отложенные вычисления позволяют сократить общий объем вычислений за счет тех вычислений, результаты которых не будут использованы. Программист может просто описывать зависимости функций друг от друга и не следить за тем, чтобы не осуществлялось «лишних вычислений».

Может показаться, что ленивые вычисления способны очень сильно ускорить программу, так как ненужные значения не будут вычисляться, но на самом деле это не так: за счет необходимости сохранения контекста и передачи в качестве аргумента функции целого выражения (вместо, например, обычного числа), в целом, программы будут работать медленнее. Кроме того, отладка программ, полагающихся на ленивые вычисления, становится намного сложнее. Если функции имеют побочные эффекты, то с ленивыми вычислениями возникает проблема: мы не знаем, когда вычислится функция, когда применится ее побочный эффект. Более того, если функция зависит от глобальных переменных, то она в разные моменты времени может иметь разный результат. Соответственно, для использования модели ленивых вычислений в языке нужно либо полностью отказаться от побочных эффектов, либо ввести отдельные правила для управления и сериализации побочных эффектов. Поэтому большая часть ЯП на ленивость не заточена, а языки, использующие ленивость, в основном навязывают чисто функциональную парадигму программирования.

Примером языка, отказавшегося от побочных эффектов, является **Hope**: абсолютно все функции (даже функции ввода-вывода, которые реализуются в интерпретаторе отдельно и имеют побочные эффекты в виде чтения или записи на экран) вычисляются лениво, однако, так как функции раскрываются слева направо, можно получить предсказуемые результаты ввода-вывода. Примером языка с правилами работы является **Haskell**: монада IO позволяет затребовать выполнение операции, имеющей побочный эффект, в определенный момент. Однако, это требует использования специального типа для функций, желающих работать с побочными эффектами, и понимается на порядок хуже, чем ввод-вывод в классических, энергично вычисляющих значения языках.

Язык, поддерживающий ленивые вычисления, может работать с бесконечными структурами данных, что делает его более выразительным. Однако, нужно быть аккуратным: неправильное использование бесконечных структур данных может зациклить программу. Например, каноническая реализация бесконечного списка, содержащего все числа последовательности Фибоначчи на Haskell выглядит так:

```
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)
```

Что значит следующее: `fibs` — это список, состоящий из 0, 1 и списка, образованного последовательной суммой элементов списка `fibs` и элементов списка, начинающегося со второго элемента списка `fibs`. Язык, не поддерживающий ленивые вычисления, не сможет при генерации списка обращаться к нему же (а не к отдельным его предыдущим элементам).

4 Поколения архитектур компьютеров и парадигмы программирования. Архитектурные особенности современных микропроцессоров. Программно-аппаратная архитектура суперкомпьютеров Ломоносов и Blue Gene/P.

Векторно-конвейерные компьютеры. Середина 70-х годов.
Особенности архитектуры: векторные функциональные устройства , зацепление функциональных устройств, векторные команды в системе команд , векторные регистры.
Программирование: векторизация самых внутренних циклов.
Суперкомпьютер Cray-1

Векторно-параллельные компьютеры. 80-е.
Особенности архитектуры: векторные функциональные устройства, зацепление функциональных устройств, векторные команды в системе команд, векторные регистры. Небольшое число процессоров объединяются над общей памятью.
Программирование: векторизация самых внутренних циклов и распараллеливание на внешнем уровне , единое адресное пространство, локальные и глобальные переменные.
Суперкомпьютер Cray X-MP

Массивно-параллельные компьютеры. Начало 90-х годов.
Особенности архитектуры: тысячи процессоров объединяются с помощью коммуникационной сети по некоторой топологии, распределенная память.
Программирование: обмен сообщениями, отсутствие единого адресного пространства, PVM, Message Passing Interface. Необходимость выделения массового параллелизма, явного распределения данных и согласования параллелизма с распределением.
Суперкомпьютер Cray T3D, Суперкомпьютер Intel Paragon XPS140

Параллельные компьютеры с общей памятью. Середина 90-х годов.
Особенности архитектуры: сотни процессоров объединяются над общей памятью.
Программирование: единое адресное пространство, локальные и глобальные переменные, Linda, OpenMP,
DEC AlphaServer, суперкомпьютер Sun StarFire

Кластеры из узлов с общей памятью. Середина 2000-х.
Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью.
Программирование: неоднородная схема MPI+OpenMP; необходимость выделения массового параллелизма, явное распределение данных, обмен сообщениями на внешнем уровне; распараллеливание в едином адресном пространстве, локальные и глобальные переменные на уровне узла с общей памятью.
Суперкомпьютер МГУ «Чебышев», «К» суперкомпьютер

Кластеры из узлов с общей памятью с ускорителями. 2010-е.
Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью; на каждом узле несколько ускорителей (GPU, Phi).

Программирование: MPI+OpenMP+OpenCL/CUDA.
Суперкомпьютер МГУ «Ломоносов», Суперкомпьютер Tianhe-2

Особенности современных микропроцессоров:

- один процессор, как правило, содержит несколько ядер
- используется многоуровневая система кеширования доступа к памяти (L1 кеш очень быстрый и для одного ядра; L2 кеш больше, но медленнее, но все еще для одного ядра; L3 кеш еще больше и для всего процессора). Кешируются как данные, так и инструкции
- процессор может переупорядочивать или смешивать инструкции и доступы к памяти
- появляются физические акселераторы для некоторых вычислительных задач (например, для криптографии или архивирования/кодирования)
- инструкции конвейеризируются, то есть разбиваются на мелкие части (вроде получения инструкции, декодирования инструкции, чтения операнда из памяти, вычисления выражения и записи результата, на самом деле у современных процессоров таких частей 16-32), что ускоряет работу инструкций
- появляются специализированные процессоры, например, GPU. Они имеют свои особенности: огромное число ядер (которые одновременно исполняют одну и ту инструкцию), специальные правила доступа к памяти, большое количество регистров

Ломоносов состоит из гибридных вычислительных узлов на процессорах **Intel Xeon** (либо 2 процессора по 4 ядра, либо 2 процессора по 6 ядер и 12-48 гб памяти на узле) и GPU-процессоров **Nvidia**. Количество x86 узлов: 5104, графических узлов 1065, x86 процессоров/ядер 12 346 / 52 168, графических ядер 954 240, суммарная оперативная память 92 Тб. Узлы соединяются четырьмя сетями: Infiniband для вычислений, 2 x Gigabit Ethernet для сервисной и управляющей сети, и отдельная сеть для глобальных барьеров и прерываний. Есть общая ФС. Пиковая производительность **1.7 Пфлопс**.

BlueGene состоит из вычислительных узлов с двумя процессорами **PowerPC** 440 700MHz (в процессоре по 4 ядра) и 2GB памяти на узел. В одной стойке находится 1024 вычислительных узла, есть две стойки. Суммарное количество ядер — 16 384, память 4 TB. Есть 5 коммуникативных сетей: тор для пересылок точка-точка, сеть для коллективных операций, сеть для глобальных барьеров и прерываний, функциональная и сервисная Ethernet-сети. Пиковая производительность **28 Тфлопс**.

5 Последовательная и параллельная сложность алгоритмов, информационный граф и ресурс параллелизма алгоритмов.

Последовательная сложность алгоритма — число операций, которые нужно выполнить при его последовательном исполнении.

Параллельная сложность алгоритма — число шагов, за которое можно выполнить данный алгоритм в предположении доступности неограниченного числа необходимых процессоров (функциональных устройств, вычислительных узлов, ядер и т.п.).

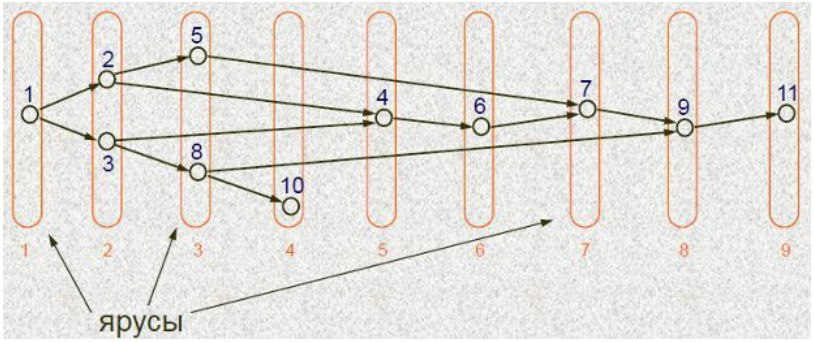
Граф алгоритма — это ориентированный ациклический мультиграф, вершины которого соответствуют операциям алгоритма, а дуги — передаче данных между ними. Вершины графа алгоритма могут соединяться несколькими дугами, в частности когда в качестве разных аргументов одной и той же операции используется одна и та же величина. Граф алгоритма почти всегда является параметризованным графом. Он используется как удобное представление алгоритма при исследовании его структуры, ресурса параллелизма, а также других свойств. Его можно рассматривать как параметризованную информационную историю. Он сохраняет ее информативность, при этом обладая компактностью за счет параметризации. Разработана методика построения графа алгоритма по исходному тексту программ.

Ярусно-параллельная форма (ЯПФ) — это представление графа алгоритма, в котором:

- все вершины разбиты на перенумерованные подмножества ярусов
- начальная вершина каждой дуги расположена на ярусе с номером меньшим, чем номер яруса конечной вершины
- между вершинами, расположенными на одном ярусе, не может быть дуг

Высота ЯПФ — это число ярусов. Ширина яруса — число вершин, расположенных на ярусе. **Ширина ЯПФ** — это максимальная ширина ярусов в ЯПФ.

Канонической ярусно-параллельной формой называется ЯПФ, высота которой на единицу больше длины критического пути, а все входные вершины расположены на первом ярусе. Для заданного графа его каноническая ЯПФ единственна. Высота канонической ЯПФ соответствует параллельной сложности алгоритма. Критический путь графа — путь максимальной длины в ориентированном ациклическом графе. **Степень параллелизма** — число операций, которые можно выполнять в любом порядке.



Ресурс параллелизма — это где и насколько мы можем сделать программу более эффективной за счет распараллеливания.

Закон Амдала

$$S_p \leq \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

где S_p — ускорение, α — доля последовательных операций, p — число процессоров в системе.

6 Логическая модель представления знаний: применение логики предикатов первого порядка. Этапы представления знаний, основные сложности. База знаний и база данных.

В логической модели знания представляются как совокупность правильных формул какой-либо формальной логической системы. Операции над знаниями — на основе логического вывода. **Формальная логическая система** задается четверкой компонент: $T =< E, R, A, P >$, где
E — множество базовых элементов формул
R — синтаксические правила, на основе которых из элементов E образуются синтаксически правильные (правильно построенные) формулы теории
A — множество аксиом: подмножество всего множества синтаксически правильных формул, которым априорно приписывается статус истинности
P — множество правил формального вывода (оперирования символами), применяя которые к формулам (аксиомам), можно получать новые п.п. формулы (теоремы)

Множество E базовых элементов формул включает:
E_1 — предметные постоянные/константы a, b, c, ...
E_2 — множество функциональных символов f, g, h, ...
E_3 — множество предикатных символов P, Q, R, ...
Вспомогательные символы:
E_4 — множество предметных переменных x, y, z, ...
E_5 — логические связки $\wedge, \vee, \neg, \rightarrow$ и кванторы: $\forall$ (всеобщность) и $\exists$ (существование)
E_6 — служебные символы: запятые, скобки

Терм — это либо предметная переменная, либо предметная постоянная/константа, либо выражение вида $f(t_1, ..., t_n)$, где f — функциональный символ, а $t_1, ..., t_n$ — термы.
Элементарная (атомарная) формула — это выражение вида $P(t_1, ..., t_n)$, где P — предикатный символ, а $(t_1, ..., t_n)$ — термы.
Формула (п.п. формула) — это либо элементарная формула, либо выражение $\neg F, F \rightarrow G, \forall x(F), \exists x(F)$, где F и G — п.п.ф.

Логика предикатов первого порядка

- кванторы могут связывать только предметные переменные, запрещены кванторы по предикатам и функциональным символам
- предикаты не могут быть аргументами других предикатов

В чистом исчислении предикатов (ИП) нет предметных констант, функций, собственных (дополнительных) аксиом. Примеры: $\forall x(P(x) \vee \neg P(x))\exists x\forall y Q(x, y) \rightarrow \forall y\exists x Q(x, y)$.
Прикладное ИП — в сигнатуре есть постоянные и функции. Пример: $\forall x(\text{Студент}(x) \wedge \text{Старший}(\text{курс}(x)) \rightarrow \exists y \text{Научный\_руководитель}(y, x))$.

Свойства логики первого порядка:

1) **полнота** чистого ИП: любому истинному высказыванию P соответствует теорема в T
2) **корректность**: все теоремы общезначимы (истинны)
3) тем самым имеется **однозначная связь** логической истинности высказывания и его выводимости
4) **неполнота** некоторых прикладных ИП: в них есть недоказуемые истинные утверждения
5) **непротиворечивость** (формальная и содержательная)
6) **неразрешимость**: не существует общего метода (алгоритма) установления общезначимости (тождественной истинности) произвольной формулы F, т.е. является ли она

теоремой или нет

7) **полуразрешимость**: есть процедура проверки общезначимости, которая для формулы-теоремы дает положительный ответ, а в случае формулы-нетеоремы возможно зацикливание

Для доказательства теорем на практике используется метод резолюций (обратный вывод).

Достоинства логической модели

- изученность и обоснованность аппарата математической логики, ее семантики и синтаксиса
- наличие мощного механизма формального логического вывода/доказательства (метод резолюций и его модификации): полнота формального вывода и универсальность (независимость от ПО)
- экономичность ПЗ: возможность вывода знаний за счет интенсиональной (понятийной) части БЗ
- естественная организация (на основе логического вывода) контроля целостности и непротиворечивости БЗ
- простота пополнения и модификации БЗ

Недостатки логической модели

- сложности перевода знаний на язык предикатов
- громоздкость, плохая читаемость формул: большое число кванторов, термов, скобок ухудшает их понимание
- плоская БЗ: нет средств структурирования и агрегирования знаний в БЗ
- очень сложно представлять в этой модели проблемно-ориентированное процедурное знание
- неэффективность механизма вывода: свойство полноты вывода имеет смысл только при наличии достаточного количества времени и памяти

Этапы представления знаний

1) **Выбор сигнатуры ИП**: E_1, E_2, E_3 и задание интерпретации всех ее составляющих (привнесение конкретного смыслового содержания). Отнесение всех сущностей ПО (объектов, свойств, отношений, процессов, явлений, ситуаций) по компонентам сигнатуры. Термы (константы и выражения на основе функциональных символов) представляют сущности описываемого мира. Предикаты представляют свойства (атрибуты) этих сущностей и их отношения.
2) **Запись формул** в фиксированной сигнатуре, отражающих факты и логические условия ПО.
3) **Логический анализ формул**:
   а) семантический анализ — проверка соответствия между миром (ПО) и его логической моделью
   б) логический анализ набора формул-аксиом:
      i. выполнимость аксиом: существует хотя бы одна интерпретация (модель)?
      ii. избыточность набора формул: есть ли среди них такая, что ее можно исключить, не изменяя множества возможных интерпретаций
      iii. недостаточность формул: допускают ли они нежелательные интерпретации

В реальности это итеративный процесс, на любом этапе возможен возврат (неоднократный) на предшествующие этапы.

Основные сложности

Этап 1 — сложности выбора сигнатуры:

- для практически любой ПО сигнатура может быть выбрана несколькими различными способами
- выбор для представления сущности: постоянная/предикат/функция? пример: отец/Отец(Глеб, Тина)/отец(Тина)
- константа может быть именем понятия/класса/типа, если не важны его разновидности, подтипы, например: $\forall x$(Соблюдать(x, Закон))
- в сигнатуре могут отсутствовать предметные постоянные и/или функциональные символы: функционально-свободная сигнатура

Этап 2 — сложности отображения логических связей, т.к. источником знаний часто выступает текст на ЕЯ:

- нечеткость ЕЯ-текста, подлежащего формализации, требуется выявление предполагаемой информации
- ограничения языка предикатов первого порядка: в ЕЯ могут быть выражения, которым нельзя найти прямого соответствия в языке первого порядка (получается формула более высокого порядка): предикат внутри предиката, квантор по предикату
- несоответствия логических связок и союзов ЕЯ:
  - логическая дизъюнкция неразделительная, т.е. ее аргументы могут быть одновременно истинными, а в ЕЯ исключающее и неисключающее или используются одинаково часто
  - аналогичная проблема с логическим следствием ($p \to q$) — истинно, если p ложно, но в реальной жизни в этом случае («если ... то») не определено

База знаний (БЗ) — набор формул теории, рассматриваемых как множество аксиом:

- экстенсиональная часть: формулы без переменных
  пример: $parent(Tom, Ann)$
- интенсиональная часть: формулы с переменными
  пример: $\forall x \forall y \exists z(woman(x) \wedge parent(z, y) \wedge parent(z, x) \to sister(x, y))$

Сравнение БЗ с БД

В БД могут фигурировать многоместные отношения (предикаты), однако есть ДЛ с многоместными отношениями, которые сводятся к традиционным.

В БЗ есть терминологический компонент (TBox), его слабый аналог — схема БД, она менее выразительна, т.к. допускаются лишь аксиомы вида $R1 \in R2$ (R_i — предикаты с одинаковым числом аргументов).

В БД в качестве запросов по сути допускаются произвольные формулы логики предикатов, а в БЗ ДЛ запросы ограниченных видов, менее выразительны даже конъюнктивные запросы (представляют некоторый фрагмент логики предикатов).

Главное отличие БЗ от БД

В **БД** принято предположение о **замкнутости мира** — если некоторое утверждение не является истинным, то оно принимается ложным.

В **БЗ** принято предположение об **открытости мира** — подобное утверждение считается ни истинным, ни ложным.

Это кардинальным образом влияет на то, какие факты считаются логически выводимыми. БД формально представляет собой одну модель — ту, в которой каждое отношение состоит в точности из тех кортежей, что записаны в БД, и по этой причине проблема произвольных логических запросов разрешима (хотя не разрешима в логике предикатов), т.к. сводится к проверке утверждения на этой фиксированной модели. БЗ представляет целое семейство моделей, и для ответа на запрос требуется их все перебрать. Ответ на запрос в БЗ есть всегда подмножество ответа БД.

7 Логическая модель представления знаний: дескриптивные логики. Концепты и роли, соотношение с логикой предикатов. Терминологии и решаемые для них задачи.

Логика ALC (Attributive Language with Complements) — первая и одна из базовых дескриптивных логик (ДЛ), на основе которой строятся многие другие. Особенности логической модели ПЗ:

- знания представляются как совокупность формул
- операции над знаниями — на базе логического вывода

Ключевые понятия ДЛ

Концепт (понятие) — множество, класс (объектов), одноместный предикат(пример: Студент, Автомобиль).

Роль — свойство, бинарное отношение, предикат (пример: Учиться (кто, где), Родитель (кто, кого)).

Выражения языка — составные концепты строятся из атомарных концептов и атомарных ролей при помощи набора конструкторов (C и B — концепты, R — роль):

- пересечение (конъюнкция) концептов: $C \sqcap B$
- объединение (дизъюнкция) концептов: $C \sqcup B$
- дополнение (отрицание) концепта: $\neg C$
- ограничение на значения роли квантором всеобщности: $\forall R.C$
- экзистенциальное ограничение (ограничение квантором существования): $\exists R.C$

Синтаксические правила для концептов:

- всякий атомарный концепт A является концептом
- выражения $\top$ и $\bot$ являются концептами (истина и ложь)
- если C есть концепт, то его дополнение — концепт
- если C и B — концепты, то их пересечение и объединение — концепты
- если C — концепт, а R — атомарная роль, то выражения $\forall R.C$ и $\exists R.C$ являются концептами

Никакие другие выражения не являются концептами.

Примеры. Атомарные концепты: Animal, Cat, Dog, Human, Male, Female, Student. Роли: has, hasChild.

$Student \sqcap Female$ — множество студенток $\forall hasChild.Male$ — все особи, чьи дети мужского пола $Human \sqcap \forall hasChild.\neg Student$ — люди, все дети которых не являются студентами.

Соотношение с логикой предикатов

Логику ALC (и многие ДЛ) можно рассматривать как фрагменты логики предикатов при естественном переводе концептов в предикатные формулы:

- для атомарных концептов C и атомарных ролей R вводятся соответственно одноместные C(x) и двухместные R(x, y) предикатные символы
- дополнение, пересечение и объединение концептов переходит в булевы связки: $\neg, \wedge, \vee$
- выражение $\forall R.C$ переходит в $\forall y(R(x,y) \rightarrow C(y))$
- выражение $\exists R.C$ переходит в $\exists y(R(x,y) \wedge C(y))$

Знания записываются в виде аксиом, все множество аксиом разбивается на:

TBox (terminological box) — набор утверждений общего вида — терминология , общие знания о понятиях (концептах) и их взаимосвязях (ролях), интенсиональные знания.

ABox (assertional box) — набор утверждений частного вида, знания об объектах-индивидах, их свойствах и связях с другими объектами, экстенсиональные знания.

Терминологические аксиомы:

- вложенность концептов: выражение вида $C \sqsubseteq B$
- эквивалентность концептов: выражение $C \equiv B$

Возможны также аксиомы (в расширениях ALC):

- вложенность ролей: $R \sqsubseteq S$
- эквивалентность ролей: $R \equiv S$

Терминология — конечный набор терминологических аксиом перечисленных видов. Терминологические задачи:

- выполнимость атомарных концептов относительно терминологии (если этого нет, то ошибка ПЗ)
- проверка (аксиом) вложенности концептов
- совместимость аксиом терминологии: проверка того, что терминология вообще имеет хотя бы одну модель
- вывод новых вложений для составных концептов
- классификация терминологии, т.е. построение иерархии (таксономии) концептов — частично упорядоченного множества всех атомарных концептов относительно вложения;
- обнаружение эквивалентных концептов
- проверка ацикличности терминологии

В логиках, содержащих ALC, проблема вложенности концептов сводится к выполнимости концепта. Ацикличность — разумное требование: смысл новых, производных концептов, стоящих в левых частях равенств-определений, однозначным образом определяется через смысл базовых концептов, не встречающихся в левых частях. Пример нарушения ацикличности:$Mother \equiv Parent \sqcap Woman$, $Parent \equiv Mother \sqcup Father$.

8 Сетевая модель представления знаний. Семантические сети и их особенности. Фреймы, их виды и структура. Межфреймовые связи, сети фреймов. Представление значений по умолчанию, понятие немонотонного вывода.

Основная идея семантических сетей — рассматривать ПО как набор сущностей объектов, понятий и связей между ними. Вся совокупность знаний представляется в виде большой ассоциативной сети концептов (понятий).

Сущности — реальные и мнимые объекты ПО индивиды, понятия, свойства, явления, события, процессы, ситуации находятся в определенных отношениях друг с другом.

Семантическая сеть — ориентированный связный граф из поименованных вершин и ребер. Вершины и ребра помечены, а ребра — направлены. Вершины — представляемые сущности.

Ребра — семантические связи вершин-сущностей. Обычно имена, приписываемые вершинам и ребрам, совпадают с названиями соответствую их сущностей и отношений ПО, тем самым семантика ПО.

Особенности

- каждая сущность представляется в сети ровно одной вершиной, поэтому все вершины имеют разные имена (для ребер это не так)
- свойство ассоциативности: любая сущность в сети представлена вершиной, из которой непосредственно доступна вся связанная с ней информация
- это позволяет быстрее находить информацию о сущности (чем в логической модели)
- ассоциативность — основополагающее свойство человеческого мышления
- минимальная значимая информация в семантической сети — ребро и две связываемые им вершины

Виды

- **однородные** и **неоднородные** — один и тот же либо разный вид семантической связи вершин сети
- **простые** и **иерархические**, в которых каждая вершина раскрывается своей сетью
- сети **с пространствами**: можно представить любые логические связки и кванторы (сеть усложняется)
- **сценарии** — однородные сети, единственное отношение в которых: причинно-следственное (каузальные сценарии); временное следование (операциональные); отношение цель/подцель (целевые)

Достоинства сетевой модели

- естественность и наглядность СС: визуальное отображение взаимосвязей между сущностями ПО
- эвристическая модель, учитывающая некоторые закономерности ПЗ у человека, в частности, свойства ассоциативности и наследования
- удобное представление декларативного знания
- возможность представления как общелогических, так и узко-предметных связей понятий

Недостатки сетевой модели

- выразительная мощность (обычной) СС существенно слабее — по сравнению с логической моделью на основе языка предикатов
- нет средств представления процедурных знаний
- «мелкость» единицы знания, что усложняет поиск в СС
- неэффективность поиска в СС (поиск изоморфного подграфа — NP-полная задача)

- вывод в СС не может гарантировать достоверность результата, т.к не имеет строгого (логического) обоснования, возможность возникновения противоречий

Фрейм — набор атрибутов/свойств описываемой сущности, точнее, набор слотов вида: атрибут = значение атрибута. Совокупность всех атрибутов определяет смысл понятия (объекта/явления/ситуации), описываемого фреймом (имманентные его свойства).

Во фреймовой системе различают 2 вида фреймов:

Фрейм-прототип соответствует обобщенной сущности (понятию, объекту, событию, явлению, ситуации). Интенсиональная часть БЗ.

Фрейм-экземпляр (фрейм-пример) соответствует конкретной/индивидной сущности (объекту, факту). Экстенсиональная часть БЗ.

Структура фрейма-экземпляра: $[fname < id_1\ val_1 > < id_2\ val_2 > ... < id_n\ val_n >]$, где $fname$ — уникальное имя фрейма, id_k — имя соответствующего слота, val_k — значение этого слота (заполнитель слота). Не все слоты фрейма-экземпляра могут быть заполнены, что соответствует не полностью определенной ситуации. Слоты могут быть терминальными и нетерминальными:

Терминальные слоты обычно имеют конкретные значения любого стандартного скалярного или составного типа (string, boolean, integer, перечислимый тип и т.п.). Терминальное значение слота соответствует сущности, которая не может быть разложена на более простые сущности.

Нетерминальные слоты имеют в качестве значения имя другого фрейма-экземпляра, представляющего содержание данного слота.

Фрейм-прототип включает присоединенные процедуры, слот в общем случае включает кроме имени атрибута несколько видов связанной с этим атрибутом информации — т.н. фасеты (facets) — аспекты атрибута:

- **ограничения** на значение (Value) соответствующего атрибута во фрейме-экземпляре
- **кардинальность** значения (Single/Multiple): одно или же несколько значений указанного типа могут быть в слоте
- **значение по умолчанию** (Default) стандартное значение, которое может быть потом переустановлено. выводы значения, получаемые на основе значений по умолчанию, называются **выводами по умолчанию**
- **процедуры**, присоединенные к слоту (ассоциированные с ними), записанные на языке реализации фреймовой системы и выражающие процедурное знание

Присоединенные процедуры:

Служебные, присоединяемые к слоту фрейма или ассоциированные с фреймом-прототипом в целом.

Демоны, присоединяемые к слоту и запускаемые автоматически при обращении к нему и выполнении определенного условия.

Для **связей фреймов** используются встроенные слоты:

- is-a/a-kind-of/AKO — слот для указания имени (имен) родового фрейма-прототипа
- Descendents/Instances — слот для задания «потомков» видовых фреймов-прототипов для данного фрейма
- Examples — слот для задания всех фреймов-экземпляров (примеров) рассматриваемого фрейма-прототипа

Виды наследования во фреймовых системах:

Строгое (жесткое) наследование — наследование без исключений: нижние узлы иерархи-

ческого дерева всегда автоматически наследуют свойства верхних узлов, и унаследованные свойства нельзя отменить, тем самым: дедуктивный, **монотонный вывод**.
Нестрогое, гибкое наследование, которое может быть отменено при необходимости: **немонотонный вывод**.

Обычно **сеть фреймов-прототипов** строится на основе таксономической связи (Род–Вид): у фреймов есть общие слоты, возникает иерархическая структура взаимосвязанных слотов и фреймов. Фреймы могут быть организованы в сеть на основе других семантических отношений: часть-целое, причинных, временных, пространственных, атрибутивных, ассоциативных, отношений сходства.

Достоинства фреймовой модели

- естественность, единообразие, интуитивность, компактность единицы ПЗ: основные свойства сущности собраны в одном месте (фрейме)
- возможность представления знаний разного уровня: поверхностный (запись лингвистической информации) и глубокий (естественный и эффективный путь классификации и построения сетей фреймов ПО)
- фрейм-прототип: представление взаимосвязанных декларативных и процедурных знаний
- присоединенные процедуры: возможность вычислить значение слота, средство организации различных способов вывода

Недостатки фреймовой модели

- слабый теоретический базис, отсутствие стандартов на структуру/состав фреймов
- трудоемкость внесения изменений в родовидовую иерархию (таксономию)
- сложность модели во всей ее полноте
- неэффективность процедур вывода на фреймах: отслеживание интерпретатором условий вызова процедур-демонов, многопроходность по АКО-связям
- трудности управления наследованием в условиях исключений (непредсказуемость поведения)
- сложность отладки фреймовых представлений

9 Понятие онтологии в инженерии знаний: состав онтологии, типы отношений концептов. Классификация онтологий, примеры. Особенности лингвистических онтологий.

Онтология — описание ПО, охватывающее:
- понятия (концепты) и их определения
- связи понятий (возможна иерархическая организация)
- правила их использования, ограничивающие их значения в рамках данной ПО

В общем случае онтология состоит из иерархии понятий, связей между ними и законов, которые действуют в рамках этой модели O = {C, R, A}, где
O — онтология
C — совокупность понятий/концептов ПО и их внутренняя структура
R — совокупность отношений/связей концептов (IS-A, PART-OF, Предок-Потомок, Выше-Ниже, Ассоциация)
A — набор аксиом (законов и правил), описывающих свойства концептов, ограничения на их связи.

Классификация онтологий
- глубина проработки
  - Heavy-weighted: тяжелые онтологии, содержащие аксиомы
  - Light-weighted: легковесные онтологии без аксиом
- степень охвата областей
  - верхнего уровня (Upper Model) или General (общие знания для нескольких ПО)
  - предметных областей (Domain-oriented)
  - прикладных задач (представление модели корпорации)
- степень формализации, язык описания
  - неформальные
  - формализованные
  - формальные: на спец. языках часто: языки логики
- владелец/пользователь онтологии
  - индивидуальная
  - групповая (сообщество, компания)
  - всеобщие
- общая методология построения
  - от слов к понятиям — лингвистическая онтология
  - от понятий — понятийная онтология
- тип отношений между понятиями
  - неоднородные
  - однородные
    - таксономии: иерархия по связи IS-A
    - партономии: иерархия по связи PART-OF
    - генеалогии: отношение «Предок-Потомок»
    - атрибутивные онтологии
    - причинно-следственные

Лингвистический подход к созданию онтологий
- методология построения: от слов к понятиям
- предполагает изучение ЕЯ и привлечение корпусов текстов

- на выходе — менее формальный ресурс онтологического характера (лингвистическая онтология)
- поскольку понятия связаны со значениями языковых единиц, для переноса на другой язык требуется серьезная адаптации
- для более строгого описания знаний о мире необходимо сопоставлять лингвистические онтологии с формальными

Виды

- **-ономия** — понятия узкой области в одном определенном отношении
  - таксономия — отношение класс-подкласс
  - партономия — отношение часть-целое
- **рубрикатор** (классификатор, каталог) — иерархическая/фасетная классификация понятий
  - предназначены для систематизации информации, знаний, документов
  - содержат взаимно непересекающиеся рубрики, покрывающие всю ПО
  - назначение рубрики — четко определить отдельную концептуальную категорию
- **тезаурус** — понятия предметной области (ПО) в отношениях синонимии, иерархии, ассоциации
  - статьи упорядочены по смыслу, а не по алфавиту
  - смысл понятия выражается посредством соотнесения его с другими понятиями и их группами, а не через определение

Различие: создаются для разных целей.

Сходство: понятиям сопоставлены текстовые/языковые (лексические) единицы.

10 Продукционная модель представления знаний: структура и цикл работы продукционной системы. Нечеткие знания и их обработка в продукционных экспертных системах.

Продукция — это правило выра ение вида: ситуация (условие) → действие или (пред)посылка → заключение (если ... то).

Единица знаний — правило продукции. Последовательное применение правила продукций к текущему состоянию задачи — порождение так называемой **цепочки вывода** $S_1 \rightarrow_{r_1} S_2 \rightarrow_{r_2} ... \rightarrow_{r_i} S_i$, где S_i — набор фактов, описывающих текущее состояние решаемой задачи.

Процесс решения задачи (вывод) — поиск решающей цепочки правил продукций для решения определенной задачи исходя из заданных фактов.

Продукционная система (ПС) — специальный вычислительный формализм.

Основные компоненты ПС

Рабочая память: **База фактов**

Предназначена для описания решаемой задачи, в начальный момент — исходная информация.

База знаний: **База правил**

Набор правил продукций, выражающих в совокупности знания в некоторой ПО.

Управляющая стратегия (Система управления)

Стратегия выбора и применения продукций при поиске решения задачи (т.е. решающей цепочки продукций).

Виды управляющих стратегий

Классификация по различным признакам:

- возможность возврата в точках выбора применяемой продукции
- направление построения решающей цепочки продукций: прямой или обратный вывод
  - **прямой вывод** в ПС: поиск решения — в прямом направлении от исходного (начального) состояния к целевому состоянию (управляемый данными)
  - **обратный вывод** в ПС: формирование пробных гипотез и проверка их на соответствие текущим фактам (управляемый целями, целевое состояние — начальная гипотеза)

Прямой и обратный вывод: сравнение

- наиболее эффективное направление поиска решения определяется структурой задачи
- если цели известны и их немного, то лучше обратный вывод
- если цели неизвестны или их много, а исходных данных мало, то лучше прямой вывод
- в диагностических экспертных системах (ЭС) чаще применяется прямой вывод, а в ЭС планирования более эффективным оказывается обратный вывод
- иногда эффективен (если возможен) двусторонний поиск: движение в прямом и обратном направлении одновременно
- при реализации ПС с прямым выводом обратный вывод может быть использован для ее настройки/отладки

Цикл работы ПС с прямым выводом

1) Сопоставление левых частей правил с фактами РП: поиск применимых правил продукций.
2) Разрешение конфликта: выбор одного правила согласно управляющей стратегии.
3) Выполнение выбранного правила, при этом возможны:
   - изменение РП (множества фактов)

- изменение БЗ
- изменение управляющей стратегии

Повторение этого цикла, пока есть применимые правила.

Конфликтное множество — это множество найденных правил, применимых в данный момент.

Разрешение конфликта: упорядочение продукций

- по степени используемости (частоте): выбирается продукция с максимальной частотой использования
- по времени последнего использования: выбирается продукция, которая была использована последней
- по длине («жесткости») условной части продукции: выбирается продукция с наиболее «длинным» условием (т.е. выражающая специальные, частные случаи)
- посредством введения приоритета правил:
  - статический приоритет задается априори, на основании сведений о важности продукции
  - динамический приоритет формируется в процессе функционирования ПС (может учитывать время нахождения продукции в конфликтном множестве)

Достоинства подукционной модели ПЗ

- простота создания и понимания правил продукций, выражающих самостоятельные фрагменты знаний
- удобство и простота пополнения и модификации БЗ, т.е. легкость построения продукционной БЗ
- возможность изменения УС без изменения БЗ и БД
- естественная возможность асинхронного и параллельного выполнения продукций (применимых в текущий момент)
- возможность программирования немонотонных выводов

Недостатки подукционной модели ПЗ

- эвристический метод: слабый теоретический базис, не хватает строгой теории
- неэффективность вывода в ПС: с увеличением числа продукций существенно замедляется скорость вывода
- сложность управления выводом в ПС: метаправила повышают гибкость, но еще больше снижают эффективность
- сложность проверки непротиворечивости БЗ с ростом числа продукций (из-за их независимости). При уверенности в правильности любой продукции все же априори нельзя быть уверенным в полноте и непротиворечивости модели ПО в виде ПС

Нечеткие знания

Истоки — не-факторы: неопределенность/нечеткость/неполнота знаний, имеют разнообразный характер:

- недоопределенность: не все объекты и связи ПО можно априори полностью определить
- нечеткость качественных оценок (красиво, хорошо, часто, молодой/старый)
- неточность получаемых количественных оценок опытных данных и эмпирических фактов (при замерах оказывает влияние погрешность прибора)
- многозначность интерпретаций ЕЯ фраз
- неоднозначность, ненадежность выводов (недетерминированность выводов в ПС)
- некорректность (противоречивость) БЗ

Модель оперирования с неточными знаниями:

- способ представления меры точности/достоверности высказываний — скалярный либо интервальный
- механизмы вывода на неточных знаниях:
  - механизмы, носящие «присоединенный характер»: пересчет мер достоверности присоединен к правилам вывода для точных высказываний
  - механизмы, для которых характерно наличие особых схем вывода, специально ориентированных на используемый язык представления неточности

Механизм неточного вывода в ПС

Простейший механизм неточного вывода присоединенного характера для ПС с правилами продукции вида $r : p \to q$ получается путем введения функций для пересчета:
Меры достоверности $t(p)$ левой части правила по мерам достоверности составляющих его частей: $t(p) = f_p(t(e_1), t(e_2), ..., t(e_n))$. Для вычисления меры достоверности $t(p)$ левой части правила могут быть использованы операции нечеткой логики (их суперпозиции):

- $t(e_1 \& e2) = min\{t(e_1), t(e_2)\}$
- $t(e_1 \vee e_2) = max\{t(e_1), t(e_2)\}$
- $t(\neg e) = 1 - t(e)$

Меры достоверности правой части (заключения правила) по мерам достоверности левой части и самого правила: $t(q) = f_r(t(r), t(p))$, где $t(e_k)$ — мера достоверности утверждения/факта e_k в РП, $t(r)$ — мера достоверности правила r (достоверности заключения при истинности посылок). Например, $t(q) = min\{t(r), t(p)\}$.

11 Методы эвристического поиска в пространстве состояний и их оптимизация: алгоритм восхождения к вершине (Hill Climbing), лучевой поиск (Beam Search), A*-алгоритм, метод ветвей и границ (Branch and Bound). Сокращение пространства состояний.

Алгоритм **поиска с восхождением к вершине** представляет собой итерационный алгоритм, который начинает со случайного решения проблемы (максимизация/минимизация целевой функции $f(\vec{x})$, где $\vec{x}$ — вектор непрерывных и/или дискретных значений), а затем пытается найти улучшенное решение путем малых изменений отдельных элементов этого решения. Если изменение приносит к улучшению результата, оно принимается в качестве нового решения и процесс повторяется до тех пор, пока изменения приносят улучшение. Алгоритм хорошо применим для поиска **локального оптимума**, но он не дает гарантии нахождения глобального оптимума.

В **алгоритме локального лучевого поиска** предусмотрено отслеживание k состояний, а не только одного состояния. Работа этого алгоритма начинается с формирования случайным образом k состояний. На каждом этапе формируются все преемники всех k состояний. Если какой-либо из этих преемников соответствует целевому состоянию, алгоритм останавливается. В противном случае алгоритм выбирает из общего списка k наилучших преемников и повторяет цикл.

A* — алгоритм поиска, который находит во взвешенном графе маршрут наименьшей стоимости от начальной вершины до выбранной конечной. В начале работы просматриваются вершины, смежные с начальной; выбирается та из них, которая имеет минимальное значение функции $f(x) = g(x) + h(x)$, где $g(x)$ — наименьшая стоимость пути в x из стартовой вершины, а $h(x)$ — эвристическое приближение стоимости пути от x до конечной вершины. После чего эта вершина раскрывается — в очередь добавляются пути через смежные вершины. Приоритет пути определяется по значению $f(x)$, где x — последняя вершина в пути. Алгоритм продолжает свою работу до тех пор, пока текущая вершина не окажется конечной.

A* является полным в том смысле, что он всегда находит решение, если таковое существует. Чтобы A* был оптимален, выбранная функция $h(x)$ должна быть допустимой эвристической функцией, то есть для любой вершины v значение $h(v)$ меньше или равно весу кратчайшего пути от v до цели. Более сильное условие — функция $h(x)$ должна быть монотонной, то есть для любой вершины v_1 и ее потомка v_2 разность $h(v_1)$ и $h(v_2)$ не превышает фактического веса ребра $c(v_1, v_2)$ от v_1 до v_2.

Метод ветвей и границ является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений. Общая идея метода может быть описана на примере поиска минимума функции $f(x)$ на множестве допустимых значений переменной x. Для метода ветвей и границ необходимы две процедуры: ветвление и нахождение оценок (границ).

Процедура ветвления состоит в разбиении множества допустимых значений переменной x на подобласти (подмножества) меньших размеров. Процедуру можно рекурсивно применять к подобластям. Полученные подобласти образуют дерево, называемое деревом поиска или деревом ветвей и границ. Узлами этого дерева являются построенные подобласти (подмножества множества значений переменной x).

Процедура нахождения оценок заключается в поиске верхних и нижних границ для решения задачи на подобласти допустимых значений переменной x.

В основе метода ветвей и границ лежит следующая идея: если нижняя граница значений функции на подобласти A дерева поиска больше, чем верхняя граница на

какой-либо ранее просмотренной подобласти B, то A может быть исключена из дальнейшего рассмотрения (правило отсева). Если нижняя граница для узла дерева совпадает с верхней границей, то это значение является минимумом функции и достигается на соответствующей подобласти.

Сокращение пространства состояний
Переменная V — то, чему можно присвоить значение
Значение X — то, что можно присвоить переменной
Домен D — область принимаемых значений
Ограничение C — условие, наложенное на значения переменных
Например, для задачи раскраски карты: V — область, X — цвет, D — множество цветов, C — граничащие области не могут быть раскрашены в один цвет.

При рассмотрении очередной вершины на область принимаемых значений могут накладываться ограничения. Если в итоге нет ни одного удовлетворяющего значения, то мы выходим из текущей ветки рекурсии и сокращаем область значений.
Виды ограничений

- ограничение из-за соседства. Много тупиковых ситуаций. Долгий поиск решения
- ограничение при распространении. Малое количество тупиковых ситуаций. Значительно уменьшает время поиска решения.
  - выбор уменьшенных доменов
  - выбор тех доменов, размер которых был уменьшен до 1

12 Классическая линейная модель регрессии: описание модели, основные ограничения, метод наименьших квадратов.

Пусть в нашей модели объект может быть описан множеством значений признаков $X = \{x_1, ..., x_m\}$ (объясняющие факторы), каждому объекту ставится в соответствие некая целевая переменная $y \in \mathbb{R}$ (объясняемая переменная), которая может быть вычислена из объясняющих факторов неизвестной функцией $y = f(\vec{x})$. Задача вычисления неизвестной целевой переменной $y \in \mathbb{R}$ по известному набору значений признаков X называется задачей регрессии. Поскольку рассматривается линейная модель регрессии, то функция f имеет вид

$$y = f(\vec{x}) = w_0 + \sum_{i=1}^{m} w_i x_i$$

Если добавить фиктивную размерность $x_0 = 1$ для каждого объекта, тогда выражение можно будет переписать, поместив w_0 под знак суммы:

$$y = \sum_{i=0}^{m} w_i x_i = \vec{w}^T \vec{x}$$

Если рассматривать матрицу объекты-признаки (или наблюдения-признаки, так как объекты иногда называют наблюдениями), у которой в строках находятся значения признаков каждого объекта из набора данных, то нам необходимо добавить единичную колонку слева.

Таким образом получаем **линейную модель регрессии**:

$$\vec{y} = X\vec{w} + \varepsilon$$

или для одного объекта:

$$y_i = \sum_{j=1}^{m+1} X_{ij} w_j + \varepsilon_i$$

где $\vec{y} \in \mathbb{R}^n$ — объясняемая (или целевая) переменная
$\vec{w}$ — вектор параметров модели (также называются весами)
X — матрица наблюдений и признаков размерности n строк на $m + 1$ столбцов (включая фиктивный столбец из единиц слева)
ε — случайная переменная, соответствующая случайной, непрогнозируемой ошибке модели

Ограничения

Помимо описанного выше, чтобы модель окончательно стала моделью линейной регрессии, накладываются следующие ограничения:

- математическое ожидание случайных ошибок равно нулю: $\forall i : M[\varepsilon_i] = 0$
- дисперсия случайных ошибок одинакова и конечна, такое свойство называется гомоскедастичностью: $\forall i : D[\varepsilon_i] = \sigma^2 < \infty$
- случайные ошибки не скоррелированы: $\forall i \neq j : Cov(\varepsilon_i, \varepsilon_j) = 0$
- случайные ошибки распределены нормально: $\varepsilon_i \sim N(0, \sigma^2)$

МНК (Метод Наименьших Квадратов)

Одним из способов вычисления значения параметров модели является метод наименьших

квадратов (МНК), который минимизирует среднеквадратичную ошибку между реальным значением целевой переменной и прогнозом, выданным моделью:

$$L(X, \vec{y}, \vec{w}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \vec{w}^T \vec{x}_i)^2 = \frac{1}{2n} \|\vec{y} - X\vec{w}\|_2^2 = \frac{1}{2n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w})$$

Необходимо найти такие значение весов w, чтобы значение функции потерь L стало минимальным. Для этого надо найти производную $\partial L / \partial w$, приравнять ее к нулю и решить уравнение относительно w. В результате получится следующее решение:

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

Полученная МНК оценка весов является лучшей оценкой параметров модели линейной регрессии среди всех линейных и несмещенных оценок (согласно теореме Маркова-Гаусса).

13 Дискриминантный анализ: постановка задачи и ее решение в случае известных параметров.

Дискриминантный анализ

Предположим, что полученные данные неоднородны. Например, они выбраны из 2-х совокупностей с разными средними. Основная задача: найти процедуру, позволяющую разделить все наблюдения по признаку принадлежности к одной из совокупностей. То есть дискриминантный анализ дает инструменты для решения задачи классификации. В рамках дискриминантного анализа решается две основные задачи:

Интерпретация — можно ли по измеренным характеристикам различить изучаемые совокупности.

Классификация — найти одну или несколько функций от измеренных характеристик, которые позволят разделить изучаемые группы.

Задача классификации и ее решение в случае известных параметров

Предполагаем, что у нас есть две совокупности W_1, W_2, описываемые случайным вектором признаков $X = (x_1, ..., x_p)$ и имеющие многомерное нормальное распределение с вектором средних $m = (m_1, ..., m_p)$ и матрицей ковариаций $\Sigma = (\sigma_{ij})$. Будем считать, что матрица Σ одинаковая у обеих совокупностей и они отличаются только средними $m^{(1)} = (m_1^{(1)}, ..., m_p^{(1)})$ и $m^{(2)} = (m_1^{(2)}, ..., m_p^{(2)})$. Предполагается, что Σ и m известны (по условию решаем задачу с известными параметрами).

Для различения совокупностей W_1, W_2 будем применять линейную дискриминантную функцию $z = \alpha_1 x_1 + ... + \alpha_p x_p$. Правило классификации будет иметь вид: W_1 если $z \geq C$, иначе W_2. Случайная величина z имеет одномерное нормальное распределение со средним $\zeta = \sum\limits_{i=1}^{p} \alpha_i m_i$ и дисперсией $\sigma_z^2 = \sum\limits_{i,j=1}^{p} \sigma_{ij}\alpha_i\alpha_j$. Расстояние Махаланобиса:

$$\Delta^2 = \frac{(\zeta_1 - \zeta_2)^2}{\sigma_z^2}$$

где ζ_1, ζ_2 — средние для z в W_1 и W_2.

Основная задача:

1) Определить параметры модели $\alpha_1, ..., \alpha_p$.
2) Найти константу C.

Хотим максимизировть $\zeta_1 - \zeta_2$, и C чтоб было по середине. Интуитивное решение:

1) $\alpha_1, ..., \alpha_p$ находятся из условия, что $\Delta^2 \to max_\alpha$
2) C есть середина отрезка с концами ζ_1, ζ_2

Далее для решения этой задачи используется Байесовская постановка: на основе прошлой информации, известно, что очередной объект принадлежит W_1 с вероятностью q_1 и к W_2 с вероятностью q_2, $q_1 + q_2 = 1$. Случайный вектор X имеет распределение $P(X|W_1)$ в случае первой совокупности и $P(X|W_2)$ — в случае второй. Если для вектора X полученные конкретные измерения x,то по формуле Байеса получаем апостериорные вероятности:

$$P(W_k|x) = \frac{q_k P(x|Wk)}{\sum\limits_{j=1}^{2} q_j P(x|W_j)}$$

Классификация в Баейсовской постановке проводится по правилу, если $P(W_1|x) \geq P(W_2|x)$, то выбирается первая совокупность, иначе — вторая.

Обозначим $P(2|1)$ — вероятность выбрать вторую совокупность, когда верна первая; $P(1|2)$ — наоборот. Тогда величина $q_1P(2|1) + q_2P(1|2)$ описывает вероятность ошибочной классификации. Если применить все это к нашей линейной модели, то получим: принимаем W_1, если $\sum\limits_{i=1}^{p} \alpha_i x_i \geq \frac{\zeta_1+\zeta_2}{2} + \ln \frac{q_2}{q_1}$, иначе — W_2.

Существует обобщение такой постановки задачи: пусть $C(1|2)$ — потери, связанные с принятием W_1, когда верна W_2; $C(2|1)$ — аналогично. Тогда схожие рассуждения приводят к обобщенной Байесовской классификации: W_1, если $\sum\limits_{i=1}^{p} \alpha_i x_i \geq \frac{\zeta_1+\zeta_2}{2} + \ln \frac{q_2 C(1|2)}{q_1 C(2|1)}$, иначе W_2.

14 Модели прогнозирования на основе деревьев принятия решений. Алгоритмы построения дерева ID3 и C4.5: критерии поиска разбиений, основные особенности.

Деревья решений являются одним из самых распространенных классификаторов. Дерево решений как алгоритм машинного обучения: объединение логических правил вида «(Значение признака a меньше x) И (Значение признака b меньше y) И ... $\Rightarrow$ Класс 1» в структуру данных дерево. Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Пусть есть выборка N и набор признаков $X = (X_1, ..., X_p)$. Общая схема построения дерева следующая: на каждом шаге выбирается тот признак, при разделении выборки по которому прирост информации (в соответствии с критерием разбиения) оказывается наибольшим. Дальше процедура повторяется рекурсивно, пока критерий не станет равен нулю. Но если доводить до такого, то дерево точно переобучится (когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении). В разных алгоритмах применяются разные эвристики для «ранней остановки»или «отсечения», чтобы избежать построения переобученного дерева. В результате получается иерархическая структура с правилами в узлах и объектами в листьях.

Наиболее распространенными являются следующие критерии разбиения:
Индекс Джини. Он измеряет «равномерность», если p_i — частоты представителей разных классов в листе дерева, то коэффициент Джини для него равен $1 - \sum_{i=1}^{n} p_i^2$.
Энтропия Шеннона. Энтропия Шеннона определяется для системы с N возможными состояниями следующим образом: $S = -\sum_{i=1}^{N} p_i \log_2 p_i$, где p_i — вероятности нахождения системы в i-ом состоянии. Интуитивно, энтропия соответствует степени хаоса в системе. Чем выше энтропия, тем менее упорядочена система и наоборот. Для предотвращения переобучения используют различные критерии останова и методы «стрижки» (прунинга) деревьев. В качестве критерия останова могут выступать следующие ограничения:

- ограничение максимальной глубины дерева
- ограничение минимального числа объектов в листе
- ограничение максимального количества листьев в дереве
- останов в случае, если все объекты в листе относятся к одному классу
- требование, что функционал качества при дроблении улучшался как минимум на s процентов

Стрижка дерева является альтернативой критериям останова, описанным выше. При использовании стрижки сначала строится переобученное дерево (например, до тех пор, пока в каждом листе не окажется по одному объекту), а затем производится оптимизация его структуры с целью улучшения обобщающей способности. Существует ряд исследований, показывающих, что стрижка позволяет достичь лучшего качества по сравнению с ранним остановом построения дерева на основе различных критериев. Тем не менее, на данный момент методы стрижки редко используются и не реализованы в большинстве библиотек для анализа данных. Причина заключается в том, что деревья сами по себе являются слабыми алгоритмами и не представляют большого интереса, а при использовании в композициях они либо должны быть переобучены (в случайных лесах), либо должны иметь очень небольшую глубину (в бустинге), из-за чего необходимость в стрижке отпадает.

Одним из методов стрижки является cost-complexity pruning. Обозначим дерево, полученное в результате работы жадного алгоритма, через T_0 . Поскольку в каждом из листьев находятся объекты только одного класса, значение функционала $R(T)$ будет минимально

на самом дереве T_0 (среди всех поддеревьев). Однако данный функционал характеризует лишь качество дерева на обучающей выборке, и чрезмерная подгонка под нее может привести к переобучению. Чтобы преодолеть эту проблему, введем новый функционал $R_\alpha(T)$, представляющий собой сумму исходного функционала $R(T)$ и штрафа за размер дерева: $R_\alpha(T) = R(T) + \alpha|T|$, где $|T|$ — число листьев в поддереве T, а $\alpha \geq 0$ — параметр. Это один из примеров регуляризованных критериев качества, которые ищут баланс между качеством классификации обучающей выборки и сложностью построенной модели.

ID3: использует энтропийный критерий. Строит дерево до тех пор, пока в каждом листе не окажутся объекты одного класса, либо пока разбиение вершины дает уменьшение энтропийного критерия.

C4.5: использует критерий нормированный энтропийный критерий. Критерий останова — ограничение на число объектов в листе. Стрижка производится с помощью метода error-based pruning, который использует оценки обобщающей способности (способность выдавать правильные результаты не только для примеров, участвовавших в процессе обучения, но и для любых новых) для принятия решения об удалении вершины. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами.

15 Нейронные сети прямого распространения. Архитектура MLP: структура сети, виды функций активации, обучение методом обратного распространения ошибки, проблема переобучения и локальных минимумов.

MLP (Multilayer perceptron)

Многослойные сети прямого распространения, или многослойный перцептрон, обычно имеют следующую архитектуру: первый слой называется входным, последний выходным, а все, что между ними — скрытые слои. Скрытых слоев может быть любое количество от 0 и больше. На сети прямого распространения накладываются ограничения: нет циклов и сигнал от входа двигается только в направлении от входа к выходу(нейроны из одного слоя не имеют между собой соединений).

На вход подается вектор-столбец признаков $X = (x_1, ..., x_n)$, который последовательно проходит через все слои к выходному слою. При этом на каждом слое производится трансформация входного вектора следующим образом (пример для первого слоя): $\vec{z_1} = a_1(W_1\vec{X} + b_1)$, где

W_1 — матрица весов для первого слоя (количество строк равно количеству нейронов в следующем слое, в данном случае в первом, а количество столбцов равно количеству нейронов на предыдущем слое, в данном случае во входном)

b_1 — вектор сдвига

$\vec{X}$ — входной в слой вектор

a_1 — функция активации

Обучение сетей

Обучение сетей производится градиентными методами по следующей процедуре:

1) Задается функция потерь $L(W, \vec{y}, \vec{X})$, где $\vec{y}$ — результат.
2) Инициализируются веса W_0.
3) Итеративно осуществлять прогон объектов через сеть, вычислять градиент $\partial L/\partial W$, обновить веса в направлении антиградиента по правилу $W_{k+1} = W_k - \eta\frac{\partial L}{\partial W}$, где η — шаг обучения (learning rate).

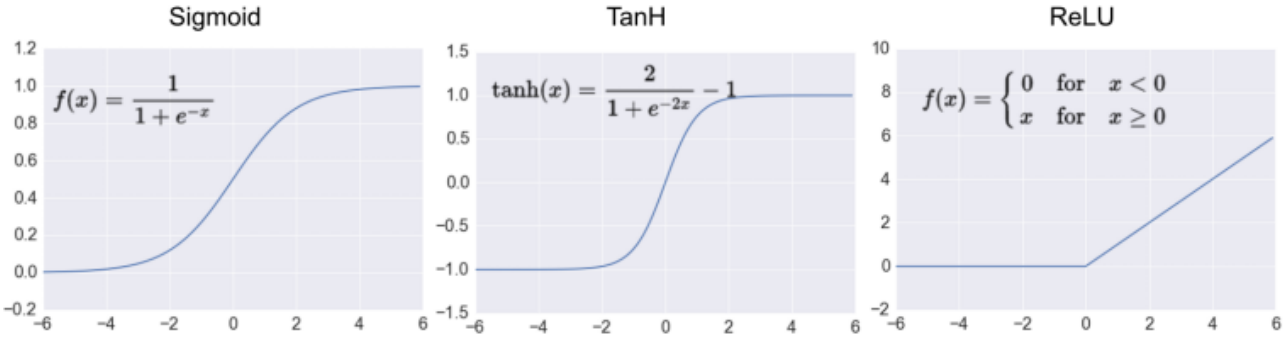
По такой же схеме обучают и другие модели регрессии, логистическую регрессию, SVM и т.д. Но в обучении сетей есть два существенных отличия:

Самое главное — вычисление градиента $\partial L/\partial W$. В сетях получается очень сложная функция, для которой крайне сложно или почти невозможно аналитически выписать производную. Поэтому для сетей был придуман специальный алгоритм вычисления градиентов — алгоритм обратного распространения ошибки.

Этот пункт также связан с вычислением градиента функции потерь, а точнее с количеством данных, на которых его считать. В классическом алгоритме градиентного спуска градиент считается по всем доступным данным, но в случае обучения нейронных сетей обычно используются большие объемы данных, и каждый раз пересчитывать функцию потерь по всему объему будет крайне медленно. Другой вариант алгоритма стохастический градиентный спуск, где градиент считается не по всему объему данных, а по одному произвольно выбранному объекту. Такой вариант алгоритма хорош тем, что в самом начале резко продвигается к минимуму функции потерь, но через некоторое время градиент начинает сильно «колбасить», из-за чего резко снижается скорость сходимости. Поэтому был придуман гибридный вариант — подсчет градиента не на всем объеме, а на небольших пакетах (batch-ах) данных.

Функции активации играют одну из ключевых ролей в построении сетей. Благодаря им в сеть добавляется нелинейность.

- **Sign**. Самая первая используемая в перцептроне Розенблата функция активации — функция sign. Использовалась для решения бинарной классификации.
- **Линейная**. Возвращает значение аргумента $a(x) = x$, хороша тем, что не насыщается и всегда есть ненулевой градиент.
- **Сигмоидная (логистическая) функция**. $a(x) = \frac{1}{1+e^{-x}}$. Возвращает значение в диапазоне от 0 до 1. Часто это значение рассматривают как вероятность или log-вероятность принадлежности объекта к классу 1. Ранее часто использовалась в обучении сетей, но имеет, с точки зрения обучения, два основных недостатка:
  1) не центрирована относительно 0, что снижает скорость сходимости
  2) довольно быстро наступает стадия насыщения при больших и малых значения аргумента, то есть когда график функции становится горизонтальным или почти горизонтальным. Это плохо, потому что градиент функции становится близким к нулю, что приводит к проблеме «размытия градиента», когда он становится слишком маленьким, из-за чего снижается или совсем останавливается обучение соответствующих нейронов
- **Гиперболический тангенс**. $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Лучше сигмоидной функции в плане центрирования относительно 0, но имеет те же проблемы с насыщением.
- **ReLU (Rectified Linear Unit)**. $ReLU(x) = max\{0, x\}$. Обладает следующими свойствами:
  + производная равна либо 0, либо 1. Таким образом либо пробрасывает на предыдущие слои градиент в неизменном виде (умножение на 1), либо полностью «убивает». Это приводит к прореживанию данных ⇒ быстрее сходится алгоритм оптимизации
  + по сравнению с сигмоидой/тангенсом не требует сложных вычислительных операции на примере возведения экспоненты в степень
  – не центрирован относительно 0
  – не дифференцируема в нуле. Хотя это не такая уж проблема, так как редко, когда доходит до нуля, в ином случае просто берется правая или левая производные
  – основной недостаток — при неправильной инициализации весов или в процессе обучения может произойти ситуация под названием «мертвого ReLU», то есть когда нейрон всегда выдает 0 и не обучается сам, и не дает предыдущим слоям



Алгоритм обратного распространения ошибки

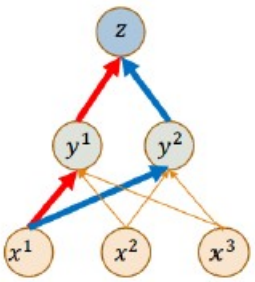
Используется в сетях для подсчета градиента $\partial L / \partial W$. Основан на идее дифференцирования сложных функций: есть функции $z = f(y)$ и $y = g(x)$, тогда $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$. Если $x \in \mathbb{R}^m, y \in \mathbb{R}^n, z \in \mathbb{R}$, $\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$ или в векторном виде $\frac{\partial z}{\partial x} = \left(\frac{\partial y}{\partial x}\right)^T \frac{\partial z}{\partial y}$, где $\frac{\partial y}{\partial x}$ — Якобиан.

В общем случае для произвольного слоя k выражением будет иметь следующий вид:

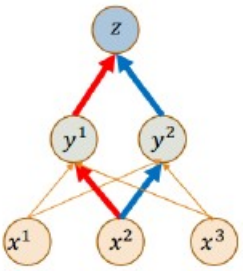
$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial a_L}\frac{\partial a_L}{\partial a_{L-1}}\frac{\partial a_{L-1}}{\partial a_{L-2}}\cdots\frac{\partial a_k}{\partial W_K}$$

или кратко

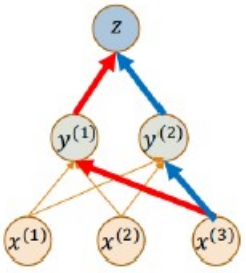
$$\frac{\partial L}{\partial W_k} = \left(\frac{\partial a_k}{\partial W_k}\right)^T \frac{\partial L}{\partial a_k}; \frac{\partial L}{\partial a_k} = \left(\frac{\partial a_{k+1}}{\partial a_k}\right)^T \frac{\partial L}{\partial a_{k+1}}$$



$$\frac{dz}{dx^1} = \frac{dz}{dy^1}\frac{dy^1}{dx^1} + \frac{dz}{dy^2}\frac{dy^2}{dx^1}$$



$$\frac{dz}{dx^2} = \frac{dz}{dy^1}\frac{dy^1}{dx^2} + \frac{dz}{dy^2}\frac{dy^2}{dx^2}$$



$$\frac{dz}{dx^3} = \frac{dz}{dy^1}\frac{dy^1}{dx^3} + \frac{dz}{dy^2}\frac{dy^2}{dx^3}$$

Как уже было сказано, итоговая функция имеет сложную структуру, с большим количеством экстремумов. Задача поиска минимума такой функции является задачей невыпуклой оптимизации (локальный минимум $\neq$ глобальный минимум). Поэтому при минимизации функции потерь градиентными методами часто может возникать ситуация, когда приходим в локальный минимум и застреваем в нем. Чтобы избежать этого, используют разные подходы, некоторые из них:

- Разработка других версий алгоритмов спуска. Например, алгоритм, в котором продвигаемся в направлении антиградиента, подсчитанного не в данной точке, а в следующей (как бы смотрим на шаг вперед).
- Варьируем значения шага обучения. Наиболее часто используемый «трюк». Сначала ставим довольно большие значения шага (значения зависят от используемого алгоритма оптимизации), чтобы перепрыгивать локальный минимумы и быстрее сходится, а через какое-то время уменьшать его значение.

Проблема переобучения

Как и другие алгоритмы обучения, нейронные сети могут переобучаться. Чем меньше данных есть и глубже сеть, тем быстрее это происходит. Обычно ловится момент, когда модель начинает переобучаться, путем построения графика ошибки на обучающей и валидационной выборках. Для решения проблемы используют разные подходы: регуляризация, ранняя остановка и другие методы.

16 Машинное обучение: обучение с учителем. Метод ближайших соседей и метод опорных векторов: общая характеристика, области применения, способы оценки решений.

Обучение с учителем (Supervised learning) — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов и множество возможных ответов. Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов — пар «объект-ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ.

Области применения в обработке текстов

- упорядочивание и навигация по набору документов (составление интернет-каталогов)
- ограничение области поиска (в поисковых системах)
- фильтрация потока документов (фильтрация спама, выявление «искусственных» текстов (боты), определение дубликатов документов)
- персонализированный/тематический подбор информации (контекстная реклама, новости об определенном событии)

Метод ближайших соседей — простейший метрический классификатор, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

Пусть задана обучающая выборка пар «объект-ответ» $X^m = \{(x_1, y_1), ..., (x_m, y_m)\}$. Пусть на множестве объектов задана функция расстояния $\rho(x, x')$. Эта функция должна быть достаточно адекватной моделью сходства объектов. Чем больше значение этой функции, тем менее схожими являются два объекта x, x'. Для произвольного объекта u расположим объекты обучающей выборки x_i в порядке возрастания расстояний до u:

$$\rho(u, x_{1,u}) \leq \rho(u, x_{2,u}) \leq \cdots \leq \rho(u, x_{m;u})$$

где через $x_{i,u}$ обозначается тот объект обучающей выборки, который является i-м соседом объекта u. Аналогичное обозначение введем и для ответа на i-м соседе: $y_{i,u}$. Таким образом, произвольный объект u порождает свою перенумерацию выборки. В наиболее общем виде алгоритм ближайших соседей есть

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{m} \big[y_{i,u} = y\big] w(i, u)$$

где $w(i, u)$ — заданная весовая функция, которая оценивает степень важности i-го соседа для классификации объекта u. Естественно полагать, что эта функция неотрицательна и не возрастает по i.

По-разному задавая весовую функцию, можно получать различные варианты метода ближайших соседей.

- $w(i, u) = [i = 1]$ — простейший метод ближайшего соседа
- $w(i, u) = [i \leq k]$ — метод k ближайших соседей
- $w(i, u) = [i \leq k]q^i$ — метод k экспоненциально взвешенных ближайших соседей, где предполагается $q < 1$

Основные проблемы и способы их решения

- **Выбор числа соседей k**. При $k = 1$ алгоритм неустойчив к шумовым выбросам: он дает ошибочные классификации не только на самих объектах-выбросах, но и на ближайших к ним объектах других классов. При $k = m$, наоборот, алгоритм чрезмерно устойчив и вырождается в константу. Таким образом, крайние значения k нежелательны.
- **Отсев шума (выбросов)**. Исключение из выборки шумовых (находящихся «в гуще» чужого класса) и неинформативных (плотно окружены другими объектами того же класса) объектов дает несколько преимуществ одновременно: повышается качество классификации, сокращается объем хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов.
- **Сверхбольшие выборки**. Метод ближайших соседей основан на явном хранении всех обучающих объектов. Сверхбольшие выборки ($m \gg 10^3$) создают несколько чисто технических проблем: необходимо не только хранить большой объем данных, но и уметь быстро находить среди них k ближайших соседей произвольного объекта u. Проблема решается двумя способами:
  1) выборка прореживается путем выбрасывания неинформативных объектов
  2) применяются специальные индексы и эффективные структуры данных для быстрого поиска ближайших соседей (например, kd-деревья)
- **Проблема выбора метрики**. Если объекты описываются числовыми векторами, часто берут евклидову метрику. Этот выбор, как правило, ничем не обоснован — просто это первое, что приходит в голову. Если признаков слишком много, то возникает проблема проклятия размерности. Проблема решается путем отбора относительно небольшого числа информативных признаков (features selection).

Метод опорных векторов (SVM)

Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

Мы полагаем, что точки имеют вид: $\{(\vec{x_1}, c_1), ..., (\vec{x_n}, c_n)\}$, где c_i принимает значение 1 или -1, в зависимости от того, какому классу принадлежит точка $\vec{x_i}$. Каждое $\vec{x_i}$ — это p-мерный вещественный вектор, обычно нормализованный значениями $[0, 1]$ или $[-1, 1]$. Если точки не будут нормализованы, то точка с большими отклонениями от средних значений координат точек слишком сильно повлияет на классификатор. Мы можем рассматривать это как учебную коллекцию, в которой для каждого элемента уже задан класс, к которому он принадлежит. Мы хотим, чтобы алгоритм метода опорных векторов классифицировал их таким же образом. Для этого мы строим разделяющую гиперплоскость, которая имеет вид: $\vec{w} \cdot \vec{x} - b = 0$. Вектор $\vec{w}$ — перпендикуляр к разделяющей гиперплоскости. Гиперплоскости могут быть описаны следующими уравнениям (с точностью до нормировки): $\vec{w} \cdot \vec{x} - b = 1, \vec{w} \cdot \vec{x} - b = -1$.

Случай линейной разделимости

Проблема построения оптимальной разделяющей гиперплоскости сводится к минимизации

$\|\vec{w}\|$. Это задача квадратичной оптимизации, которая имеет вид:

$$\begin{cases} \|\vec{w}\|^2 \to \min \\ c_i(\vec{w} \cdot \vec{x_i} - b) \geq 1,\ 1 \leq i \leq n,\ c_i = \pm 1 \end{cases}$$

Сводится к эквивалентной задаче квадратичного программирования, содержащую только двойственные переменные $\lambda = (\lambda_1, ..., \lambda_n)$:

$$\begin{cases} L = \sum\limits_{i=1}^{n} \lambda_i - \frac{1}{2} \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} \lambda_i \lambda_j c_i c_j (\vec{x_i} \cdot \vec{x_j}) \to \min_\lambda \\ \lambda_i \geq 0,\ 1 \leq i \leq n \\ \sum\limits_{i=1}^{n} \lambda_i c_i = 0 \end{cases}$$

Тогда $\vec{w}$ и b можно найти по формулам: $w = \sum\limits_{i=1}^{n} \lambda_i c_i \vec{x_i},\ b = \vec{w} \cdot \vec{x_i} - c_i$. В итоге алгоритм классификации может быть записан в виде:

$$a(x) = sign\left(\sum_{i=1}^{n} \lambda_i c_i \vec{x_i} \cdot \vec{x} - b\right)$$

Случай линейной неразделимости. Ядра

В этом случае нужно вложить пространство признаков в пространство большей размерности с помощью отображения φ. Результирующий алгоритм крайне похож на алгоритм линейной классификации, с той лишь разницей, что каждое скалярное произведение в приведенных выше формулах заменяется нелинейной функцией ядра (скалярным произведением в пространстве с большей размерностью). Наиболее распространенные ядра:

- полиномиальное (однородное): $k(\vec{x}, \vec{x'}) = (\vec{x} \cdot \vec{x'})^d$
- полиномиальное (неоднородное): $k(\vec{x}, \vec{x'}) = (\vec{x} \cdot \vec{x'} + 1)^d$
- радиальное: $k(\vec{x}, \vec{x'}) = e^{-\gamma \|\vec{x} - \vec{x'}\|^2}$, для $\gamma > 0$

Преимущества и недостатки SVM

- это наиболее быстрый метод нахождения решающих функций
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение
- метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию
- метод чувствителен к шумам и стандартизации данных
- не существует общего подхода к автоматическому выбору ядра (и построению спрямляющего подпространства в целом) в случае линейной неразделимости классов

Оценка качества классификации

- ошибка классификатора (доля неправильных решений) на тестовом подмножестве
- точность показывает, какая доля объектов, выделенных классификатором как положительные, действительно является положительными $Pr = \frac{TP}{TP+FP}$
- полнота показывает, какая часть положительных объектов была выделена классификатором $Re = \frac{TP}{TP+FN}$
- F-мера — взвешенное гармоническое среднее точности и полноты. $F_\beta = \frac{(\beta^2+1)Pr}{\beta^2 Pr + Re}$. Обычно $\beta^2 = 1 \Rightarrow F_1 = \frac{2PrRe}{Pr+Re}$

17 Машинное обучение: обучение без учителя. Метод кластеризации K-средних. Иерархическая кластеризация. Оценка качества кластеризации. Кластеризация текстовой коллекции: признаковая модель текста и ее применение.

Обучение без учителя (Unsupervised learning) — раздел машинного обучения, изучающий широкий класс задач обработки данных, в которых известны только описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами.

Области применения в обработке текстов

- упорядочивание и навигация по набору документов (составление интернет-каталогов)
- информационный поиск
  - улучшение результатов поиска
  - «интеллектуальная» группировка результатов

Плоские алгоритмы создают неструктурированное множество кластеров.
Иерархические алгоритмы создают структурированное множество кластеров.

Алгоритм K-средних — алгоритм машинного обучения, решающий задачу кластеризации. Этот алгоритм является неиерархическим, итерационным методом кластеризации. Основная идея алгоритма заключается в том, что данные произвольно разбиваются на кластеры, после чего итеративно перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Цель алгоритма заключается в разделении n наблюдений на k кластеров таким образом, чтобы каждое наблюдение принадлежало ровно одному кластеру, расположенному на наименьшем расстоянии от наблюдения.

Есть набор из n наблюдений $X = x_1, x_2, ..., x_n,\ x_i \in \mathbb{R}^d,\ i = \overline{1,n}$. k — требуемое число кластеров, $k \in \mathbb{N}, k \leq n$. Требуется разделить множество наблюдений X на k кластеров $S_1, S_2, ..., S_k$ так, чтобы: $S_i \cap S_j = \varnothing, i \neq j$ и $\bigcup\limits_{i=1}^{k} S_i = X$.

Шаги алгоритма

1) **Инициализация кластеров.** Выбирается произвольное множество точек μ_i, рассматриваемых как начальные центры кластеров: $\mu_i^{(0)} = \mu_i,\ i = \overline{1,k}$
2) **Распределение векторов по кластерам.**
   Шаг t: $\forall x_i \in X,\ i = \overline{1,n} : x_i \in S_j \Leftrightarrow j = \arg\min\limits_k \rho(x_i, \mu_k^{(t-1)})^2,\ \rho$ — функция расстояния.
3) **Пересчет центров кластеров**. Шаг t: $\forall i \in \overline{1,k} : \mu_i^{(t)} = \frac{1}{|S_i|} \sum\limits_{x \in S_i} x$
4) **Проверка условия останова**. Если $\exists i \in \overline{1,k} : \mu_i^{(t)} \neq \mu_i^{(t-1)}$, то новая итерация (шаг 2).

Достоинства

- сравнительно высокая эффективность при простоте реализации
- высокое качество кластеризации
- возможность распараллеливания
- существование множества модификаций

Недостатки

- количество кластеров является параметром алгоритма

- инициализация центров кластеров в значительной степени влияет на результат кластеризации
- выбросы, далекие от центров настоящих кластеров, все равно учитываются при вычислении их центров
- возможность сходимости к локальному оптимуму
- алгоритм неприменим к данным, для которых не определено понятие «среднего», например, категориальным данным

Иерархическая кластеризация — комплекс алгоритмов, использующих разделение крупных кластеров на более мелкие или объединение мелких в более крупные. Соответственно, выделяют разделительную (нисходящую) и объединительную (восходящую) кластеризации.

В разделительной кластеризации все исходное множество данных сначала рассматривается как один кластер, который расщепляется на два, те в свою очередь еще на два и т. д. до тех пор, пока каждый из них не будет состоять из единственного объекта.

В объединительной кластеризации сначала каждый объект исходного множества рассматривается как отдельный кластер, затем ищутся два объекта, расстояние между которыми минимально, и объединяются в один и т. д. Данная процедура продолжается до тех пор, пока все объекты не будут собраны в единственный кластер.

Процесс завершается при достижении одного из следующих условий:
- кластеры, образованные на новом шаге, теряют компактность
- образуется требуемое число кластеров
- кластеризация удовлетворяет требованиям эксперта исследователя

Достоинства — понятность

Недостатки
- не масштабируется: квадратичная временная сложность от числа кластеризуемых объектов
- жадный алгоритм — локальная оптимальность с точки зрения минимизации внутриклассовых расстояний и максимизации межклассовых
- относительно слабая интерпретируемость

Внешняя оценка качества

A — число пар элементов, находящихся в одном кластере как в полученном, так и в эталонном разбиении

B — число пар элементов, находящихся в разных кластерах как в полученном, так и в эталонном разбиении

C — число пар элементов, находящихся в одном кластере полученного, но в разных эталонного разбиения

D — число пар элементов, находящихся в разных кластерах полученного, но в одном эталонного разбиения

- Rand Measure: $RandIndex = \frac{A+B}{A+B+C+D}$
- F-measure: $Pr = \frac{A}{A+C},\ Re = \frac{A}{A+D},\ F_1 = \frac{2PrRe}{Pr+Re}$
- Индекс Жакара: $J = \frac{A}{A+C+D}$

Внутренняя оценка качества

- среднее внутрикластерное расстояние должно быть как можно меньше

$$F_0 = \frac{\sum\limits_{i<j}[y_i = y_j]\rho(x_i, x_j)}{\sum\limits_{i<j}[y_i = y_j]} \to \min$$

- среднее межкластерное расстояние должно быть как можно больше

$$F_1 = \frac{\sum\limits_{i<j}[y_i \neq y_j]\rho(x_i, x_j)}{\sum\limits_{i<j}[y_i \neq y_j]} \to \max$$

Признаковая модель текста

- признаки определены для каждого документа
- информационные признаки
  - лингвистические
    - слова: обычно значимые, не служебные; реже — контексты слов и словосочетания
    - N-граммы (шинглы)
  - статистические
    - количество слов с большой буквы
    - доля различных частей речи в тексте
    - доля сложных предложений
    - средняя длина слова/предложения и т.д.
  - экстралингвистические
    - тип документа, автор, заголовок
    - дата публикации, источник информации
    - гиперссылки и пр.
  - структурные
- виды модели: bag of words (мешок слов), векторная

Классификация (рубрицирование) — отнесение документа к заранее известным классам (рубрикам). Часто документы классифицируют по их содержанию. Классы с заданными характеристиками, иерархическая система классов.

Кластеризация — разбиение заданного множества документов на подмножества (кластеры). Документы в кластерах похожи по смыслу/теме/структуре. Характеристики, количество, структура кластеров заранее не заданы.

18 Уровни языка и основные этапы и модули анализа текста лингвистическим процессором. Подходы к решению прикладных задач компьютерной лингвистики: основанный на правилах и основанный на машинном обучении.

Уровни языка и текста

Структурная модель ЕЯ учитывает особенности уровней текста:

1) графематический (символы)
2) морфологический (слова)
3) синтаксический (предложения)
4) семантический (слова, предложения, текст)
5) дискурсивный — уровень связного текста: текст в целом

Этапы обработки ∼ Уровни языка

Этапы обработки текста

1) **графематический** анализ и сегментация: выделение и классификация основных единиц текста: слов, предложений, абзацев
   - инженерный подход. Опора на:
     - формальный аппарата (грамматики, регулярные выражения)
     - эвристические правила определения токенов и предложений
     - словари сокращений, имен, знаков и т.д.
   - машинное обучение по размеченным текстам
   - гибридный подход
2) **морфологический** анализ: нормализация словоформ или их стемминг, определение морфологических характеристик. Для построения морфопроцессоров в основном используется инженерный подход. Опора на:
   - словари основ, словоформ, флексий, исключений
   - правила преобразования словоформ
   - эвристические правила анализа новых слов и т.д.
3) **синтаксический** анализ: построение синтаксической структуры предложения
   - инженерный подход: строится грамматика составляющих или зависимостей
     - экспертами-лингвистами
     - автоматизированно на основе корпусов текстов с синтаксической разметкой
   - статистический подход: сбор статистики на большом корпусе с синтаксической разметкой (для получения наиболее вероятного дерева)
   - гибридный подход
4) **семантический** и **дискурсивный** анализы

   Формальное представление семантики (свойства объектов, их отношения, состояния, действия) на основе моделей представления знаний (фреймы, формулы исчисления предикатов, семантические сети)
   - **локальный семантический анализ**
     - определение семантики слов/словосочетаний: разрешение многозначности, определение их семантических характеристик (МЕСТО, ВРЕМЯ, ...)
     - установление семантических отношений между словами и словосочетаниями

     Основные подходы:
     - словари и правила
     - машинное обучение по размеченным данным
   - **глобальный семантический анализ** (один из видов — дискурсивный анализ)

     Дискурсивный анализ – анализ связного текста, с учетом его назначения,

функции, цели. Подходы:

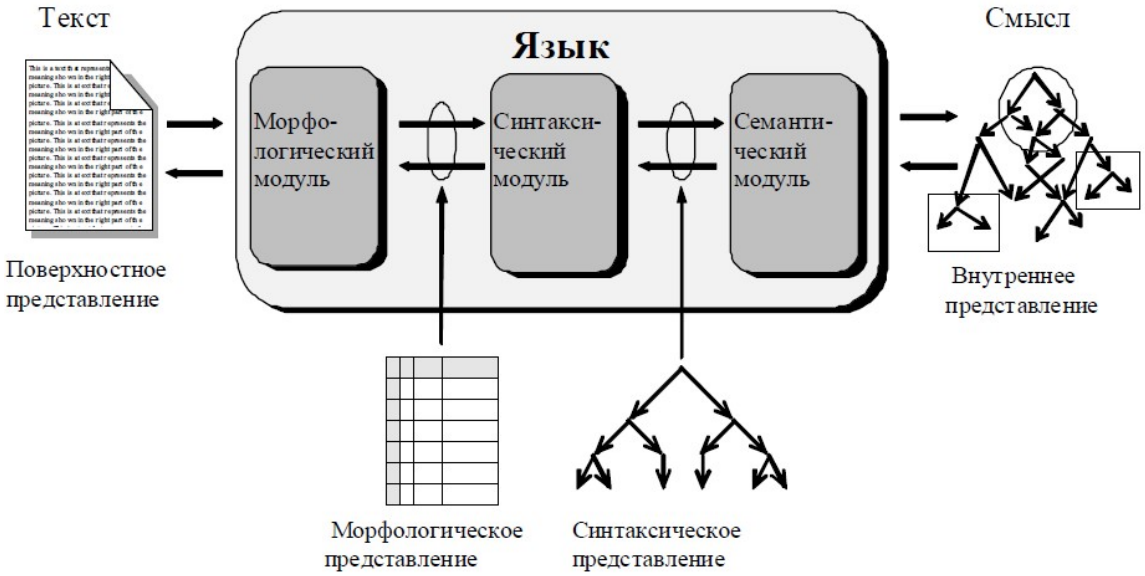
- лингвистическая информация и правила: словари местоимений, имен и названий, общие критерии установления кореференции (тождество двух языковых выражений), эмпирические закономерности текстов конкретной ПО
- машинное обучение: есть тексты на английском, с размеченными дискурсивными отношениями

Лингвистический процессор(ЛП) — программа обработки текстов на ЕЯ в рамках конкретной прикладной задачи. ЛП могут обеспечивать, в зависимости от задачи:

- анализ: разбор и «понимание» поступившей на вход фразы на ЕЯ
- синтез: построение фразы ЕЯ по формальному описанию ее смысла

Основой ЛП является модель языка и/или текста (модели зависят от ЕЯ и прикладной задачи). ЛП опирается на лингвистические ресурсы.

Основные модули обработки текста в структурной модели



Выбор подхода

Основанный на правилах, инженерный (модель — **набор лингвистических правил**)

- есть эксперты и словарные ресурсы
- мало размеченных данных
- есть временные ресурсы
- необходимо хорошее (и выше) качество работы

Основанный на машинном обучении (модель — **машинный классификатор**)

- нет экспертов ПО и словарных ресурсов
- много размеченных данных и получение их дешево
- необходимо быстрое построение приложения
- не требуется лингвистическая интерпретация результатов

19 Статистическая языковая модель: униграммные и N-граммные модели, области их применения в задачах обработки текстов. Сглаживание модели. Перплексия и способ ее подсчета.

Статистическая языковая модель: приписывание любой фразе/предложению вероятности ее появления в тексте. При создании модели:

- вычисляются вероятности (по корпусу)
- устраняются недостатки модели: сглаживание
- оценивается качество построенной модели

N-граммная модель

Рассматриваются N-граммы — последовательности слов $W = w_1w_2...w_N$. Обычно $N = 1, 2, 3, 4$ (униграммы, биграммы, триграммы, тетраграммы). Делается предположение: вероятность появления очередного слова в предложении зависит от предыдущих $N-1$ слов.

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_N|w_1w_2...w_{N-1})$$

Вероятность всего предложения вычисляется как произведение вероятностей входящих в него N-грамм. Теория: вероятности вычисляются по корпусу с опорой на частоты фраз

$$P(w_N|w_1w_2...w_{N-1}) = \frac{F(w_1w_2...w_N)}{F(w_1w_2...w_{N-1})}$$

где F — частота.

Реальность — ограниченность корпуса: чем больше N, тем большего количества правильных N-грамм в корпусе нет $\Rightarrow P(w_N|w_1w_2...w_{N-1})$ не всегда можно верно вычислить. Марковская модель: учитывается только $k < N-1$ предыдущих слов. При $k = 2$ получим

$$P(w_1w_2...w_N) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_N|w_{N-1})$$

Плюсы и минусы N-граммной модели

\+ возможность построения модели по корпусу
\+ относительная простота использования
– предположение независимости вероятности от более дальних слов
– большие объемы хранимой информации
– недостаточность данных для построения модели (ограниченность корпуса и его специфика): N-грамма правильная, но не встретилась в корпусе ($P = 0$)

Области применения моделей

- Выявление ошибок
  - опечатки: P(на странице) $> P$(на страннице)
  - неверный порядок слов: $P(the\ bed\ is) > P(the\ is\ bad)$
  - некорректное употребление слов: $P(my\ house\ is\ small) > P(my\ home\ is\ small)$
- Машинный перевод: выбор слова при переводе
  herd of horses — табун или стадо лошадей
- Распознавание речи: выбор между одинаково звучащими словами
  P(*у меня грипп*) $> P$(*у меня гриб*)
- Генерация текста: составление рефератов и аннотаций

Сглаживание

Недостаточность данных для построения модели $\Rightarrow$ некорректные вероятности. Для устранения некорректности применяется сглаживание: понижают P одних и повышают

P других N-грамм. Один из способов — добавление $\alpha \leq 1$. $P(w_2|w_1) = \frac{F(w_1 w_2)+\alpha}{F(w_1)+\alpha V}$, где V — количество различных слов в корпусе.

Оценка модели: перплексия Перплексия — величина обратная средней вероятности появления слов в тестируемом тексте. Позволяет оценить вероятность появления некоторого текста в рамках построенной модели. Перплексию также можно трактовать как среднее количество слов, которые могут следовать за некоторым фиксированным словом. Таким образом, чем больше перплексия, тем выше неоднозначность, и следовательно, хуже языковая модель.

$$PP = P_a(w_1 w_2 ... w_V)^{-\frac{1}{V}} = \sqrt[V]{\prod_{i=1}^{V} \frac{1}{P(w_i|w_1 ... w_{i-1})}}$$

где V — объем тестируемого текста.

20 Генерация текста: методы и приложения. Машинный перевод: основные подходы и стратегии. Автоматическое реферирование и аннотирование документов: базовые технологии.

Генерация текста

- осмысленный текст: шаблоны, формальные грамматики, правила, нейронные сети
- псевдоосмысленный текст: марковские цепи, добавление вводных слов, разбавление текста, перестановка слов, использование синонимов

Шаблоны

- выдача готовой строки (canned method), неизменяющийся шаблон
  Welcome. Please insert your card and enter your password
- бесконтекстная вставка (templates), изменяющийся шаблон, заполнение пропусков, слотов
  Уважаемый(ая) [#1]! Ваше сообщение получено [# data]
- контекстная вставка (phrase-based) [Шаблон с указанием признаков][Имя]
  *Уважаем**ый** Иван!* или *Уважаем**ая** Елена!*
- каскадные единицы (cascaded items) [Выбор шаблона в зависимости от признаков][Имя]
  Дерево шаблонов строится путем последовательного уточнения первоначального абстрактного шаблона
- свойства (features). Свойство — изменяющийся параметр текста: варианты порядка слов, морфологические характеристики, тип предложения. Значение свойства — конкретное значение, определяющее изменения.
  <Число N> <Уточнение><If N<2 file else files><глагол-время>

Приложения

- имитация речевой деятельности человека в ранних системах; «научные», «философские» статьи, «стихи»
- размножение текстов для продвижения сайтов, товаров, поисковой оптимизации, пропаганды (реклама, отзывы)
- вопросно-ответные системы
- виртуальные собеседники (чатботы)

Машинный перевод

Rule-based Machine Translation (RBMT), Машинный перевод, основанный на правилах

- Пословный перевод.
- Трансферные системы: морфологический, синтаксический и семантический анализ на языке входа, преобразование в структуру выходного языка, синтез на выходном языке.
- Интерлингвистические системы: анализ и синтез через язык-посредник.

Достоинства и недостатки:

\+ синтаксическая и морфологическая точность
\+ стабильность и предсказуемость результата
\+ возможность настройки на предметную область
– трудоемкость и длительность разработки
– необходимость поддерживать и актуализировать лингвистические базы данных
– машинный акцент — неестественность перевода

Example-Based MT (EBMT), Машинный перевод, основанный на примерах

Требуется БЗ — языковой корпус из пар предложений. Перевод по аналогии.

Достоинства и недостатки:

+ высокое качество перевода, хорошо справляется с контекстными задачами (перевод идиом), для разработки не нужны квалифицированные лингвисты

– нужны большие параллельные корпуса текста, размеченные определенным образом, от них сильно зависит перевод, продолжительное время обучения, требовательность к ресурсам

Statistical Machine Translation (SMT), Статистический машинный перевод

База: Параллельные текстовые корпусы (парламентские отчеты, документы Евросоюза, переводы роликов и фильмов)

Теория: Языковая модель — набор N-грамм многоязычного корпуса с их вероятностными характеристиками, на базе которого ведется поиск наиболее вероятного перевода. Необходимо найти в конечном языке такое предложение Y, которое с наибольшей вероятностью является переводом предложения X начального языка.

Достоинства и недостатки:

+ быстрая настройка основных модулей по корпусу (машинное обучение)

– дефицит параллельных корпусов текстов

– нестабильность перевода, т.к. зависимость от статистики (Лев Толстой — Lion Thick), грамматических ошибок и опечаток в текстах

Hybrid Machine Translation (HMT), Гибридный машинный перевод

Основные направления:

- Интеграция статистического модуля в перевод, основанный на правилах (перевод по правилам, затем выбор нужного варианта по статистике).
- Интеграция правил в статистическую модель (предобработка исходного текста с помощью правил).

Neural Machine Translation (NMT), Нейронный машинный перевод

Целью НМП является создание полностью обучаемой модели, каждый компонент которой настраивается на основе обучающих корпусов, чтобы максимизировать качество перевода. Полностью обучаемая НМП-модель генерирует как можно более естественное представление целевого предложения. Используются рекурентные нейронные сети. Возможность учитывать не только типичное значение фразы, но и ее контекст.

Автоматическое реферирование и аннотирование документов

Экстракты - извлечение фрагментов исходных текстов (предложений). Может основываться на:

Определении значимости (веса) предложения

- вес предложения зависит от весов слов. Из потенциального списка значимых слов исключаются стоп-слова (предлоги, союзы и т.д.), слова, широко-употребляемые в области, слова с низкой частотностью. Для назначения весов используется частоты встречаемости слов, tf-idf...
- вес предложения увеличивается, если в нем есть слова заголовка
- присутствие особых конструкций: *подчеркнем, результатом будет*
- позиция предложений в тексте: предложения в начале более важны
- длина предложения, именованные сущности, связность предложения с предыдущим

Построении тематических моделей текста

- Лексические цепочки — объекты могут описываться набором связанных слов и предложений. Для формирования цепочек используются тезаурусы. При каждой итерации отбираются предложения содержащие узловой элемент цепочки и предложения,

содержащие 2 или более значимые цепочки.

- Латентный семантический анализ — неявное представление семантических особенностей текста на основании статистики совместной встречаемости слов.
- Байесовские тематические модели — исследуется вероятность появления пары (документ-слово).

Определении наибольшей значимости предложения на основе графа предложений

Вершины графа — предложения. Дуги — сходство (similarity) между предложениями (определяется косинусной мерой). Важны значительные степени сходства, выше заданного порога. Связи ниже этого порога не рассматриваются. Центральность предложения — при заданном пороге оно связано с максимальным числом других предложений. Это предложение выбирается в аннотацию.

Машинном обучении

Использование различных признаков предложений и их комбинаций. Задача бинарной классификации: разбиение всех предложений входной коллекции на принадлежащие и не принадлежащие итоговой аннотации. При обучении подбирается комбинация из характеристик извлекаемого предложения:

- длина предложения (больше/меньше порога)
- позиция в тексте: номер абзаца, позиция внутри абзаца (в начале/середине/конце)
- присутствие ключевых фраз, наличие слов из заголовка документа, количество тематических слов (слов ПО), присутствие имен собственных

Стратегии отбора предложений

- **локальная оптимизация**. Оптимизация выбора очередного предложения. Максимизируется сходство с исходным документом. Минимизируется сходство с уже отобранными в аннотацию предложениями
- **глобальная оптимизация**. Оптимизация некоторой функции (присутствие наиболее частотных слов или выражений в аннотации)

После извлечения предложений производится улучшение связности и читаемости аннотации: замена аббревиатур, приведение номеров и дат к стандартному виду, замена временных ссылок (*в конце следующего года* → *в конце 2018 г.*), замена двусмысленностей и дискурсивных форм (*Но, это значит...* → *Это значит...*), конечная сортировка предложений.

Проблемы аннотирования на основе извлечения:

- дублирование информации
- отсутствие связности
- нарушение исходного смысла

Абстракты — генерация текста. В настоящее время возможно для текстов из узких предметных областей. Общая схема: извлечение определенной информации из текстов (обычно с использованием лингвистических шаблонов) → заполнение извлеченными данными заранее определенных шаблонов типичных фраз аннотаций.

Приложения

- Автоматизированный отбор документов, соответствующих тематике.
- Автоматизированная кластеризация научных статей.
- Автоматическое аннотирование голосовых сообщений, форумных веток, совещаний и встреч.

21 Извлечение информации из текстов: особенности направления, виды извлекаемых данных, применяемые подходы. Понятие лингвистического шаблона. Задача извлечения мнений и анализа тональности текстов.

Извлечение информации — автоматическое извлечение данных из текста на ЕЯ:

- обрабатывается отдельный текст или коллекция
- тексты неструктурированные (без метаданных)
- извлекаются данные, релевантные определенной: проблеме, вопросу, теме
- извлеченные данные:
  - структурируются в виде таблиц, шаблонов
  - накапливаются в базах знаний
  - обрабатываются: сортируются, размечаются, отбираются, сохраняются в базах данных

Приложения

- мониторинг новостных лент
- составление дайджестов, рефератов, досье
- сбор данных для анализа экономической, производственной и прочей деятельности

Виды извлекаемой информации

- значимые объекты для темы/области, именованные сущности (ИС): *персоны, организации*. Нахождение именований сущности в тексте → Определение категории сущности → Разрешение кореференции → Связывание сущности с референтом (если сущность является именем собственным)

  Сложности извлечения ИС
  - нельзя зафиксировать в словаре, например, все названия компаний
  - особенности именования не являются строгими и однозначными
  - в качестве извлекаемых объектов могут выступать обычные слова и словосочетания текста: *научный сотрудник*
  - в зависимости от контекста сущность может относиться к разным видам (категориям): *В России прошли... — географический объект, Россия отказалась от ... — страна*
  - из контекста может быть не понятен референт: *Нам задали читать Толстого*
- отношения между объектами: *быть частью, быть владельцем*
- атрибуты объектов: для персоны — *место работы, должность, телефон, подразделение*
- факты/события: *прошла встреча, выдан кредит*. Объекты, их атрибуты и отношения извлекаются из текстов. Из них формируются факты и события. События описываются набором параметров и их значений (именованные сущности). Такой набор образует так называемый **семантический фрейм события**

Особенности

- большое количество разных сущностей/объектов, постоянно появляются новые
- объекты разнородны, правила их именования и извлечения различны
- современные прикладные системы извлечения информации, как правило, ориентированы на обработку текстов в узких предметных областях

Подходы

1) Машинное обучение (опора на статистику, необходим размеченный корпус текстов). Используемые методы: наивный байесовский классификатор, деревья решений, метод опорных векторов, логистическая, HMM и CRF, нейронные сети.
2) Инженерный подход (лингвистические шаблоны и правила, нужны эксперты для их

написания, а также специальные языки и ПО их поддержки).

3) Комбинации подходов

Лингвистические шаблоны — формальное описание языковых конструкций, их лексического состава и грамматических свойств. Виды шаблонов: лексические, лексико-синтаксические. Элементы шаблонов: словоформы, лексемы (с указанием характеристик), грамматические образцы: именные и другие группы.

- объекты и атрибуты
  (^([0-9]+[.]+[0-9]+)|(0)$) — 0, 1.2, 12.345
  A N — прелестный кварк, солнечное сплетение
- отношения между объектами
  (Ng — именная именная группа) Ng_1 «является частью» Ng_2
  Процессор является частью компьютера
- факты и события
  Fact=(Loc, Ship)
  В Белом море пошел ко дну корабль ВМС Индии: Loc «пошел ко дну» Ship
  Корабль ВМС Индии затонул в Белом море: Ship «затонул» Loc

Извлечение мнения и определение тональности

Мнение — суждение автора по поводу некоторого объекта/услуги/продукта/товара/организации.

Тональность — авторская эмоциональная оценка.

Приложения

- извлечение и оценка мнений о персонах, партиях, товарах, услугах
- анализ новостей, обзоров, сообщений пользователей

Сложности

- оценочные слова в разных контекстах могут иметь разные смыслы: *cool story — cool heater*
- оценочные слова могут не нести оценки: *посоветуйте мне ресторан, в котором вкусно кормят*
- предложения без оценочных слов могут нести оценку: *попался волос в хлебе*

Извлечение мнений

Мнением может быть: часть предложения, все предложение или текст в целом

- Нахождение текстовых фрагментов, выражающих мнение (разновидность извлечения информации)
  - распознавание определенных языковых конструкций (прямая и косвенная речь)
  - поиск эмоционально-окрашенных слов и фраз
- Выявление свойств мнений: объект мнения, эмоциональная оценка (тональность), метаинформация (автор мнения, время).

Анализ тональности

Можно анализировать тональность документа, предложения, суждения об объекте и его характеристиках — аспектный анализ.

Виды оценок

- двоичная: позитивная/негативная
- троичная: позитивная/негативная/нейтральная
- по некоторой шкале

Подходы

- **Инженерный подход**

  - используются шаблоны и правила словоформ оценочной лексики
  - оценочная лексика зависит от ПО оцениваемого текста (в том числе окрас слов)
  - много неформальной лексики, затрудняющих оценку

  Составление словаря:
  - вручную, экспертами — надежно но трудоемко
  - автоматизировано:
    * на основе правил, шаблонов и контекстов слов оценочной лексики
    * с учетом статистических показателей слов
    * с использованием словарей
    * основная проблема с назначением конкретной тональности
  - комбинация методов
- **Машинное обучение**
  Признаки — слова, части речи, биграммы, позиции в тексте и т.д. Разные методы классификации оценочных слов: Наивный Байес, SVM, логистическая регрессия, HMM. Методы кластеризации используются реже: учитываем слова, которые часто встречаются в данном тексте, но редко в большинстве других.

Аспектный анализ мнений

Сущность — объект мнения (продукт, сервис, тема, человек, организация). **Аспекты** — составные части, свойства, характерные признаки и черты сущности. Мнение касается определенной сущности. В нем может высказываться суждение о сущности в целом и/или о ее отдельных аспектах.

Виды аспектов

- **явные**: указывают на аспекты, но не содержат в себе оценку (*номер, завтрак, wi-fi*)
- **неявные**: сочетают аспект и оценку в одном слове (*вкусно = хорошая еда*)
- **тональные факты**: описывают состояние дел и несут в себе оценку (*чайник в номере*)

Задачи аспектного анализа

- Выявление аспектных терминов (опора на частоту существительных и их групп, учет конструкций с оценочной лексикой, машинное обучение, гибридный подход).
- Объединение аспектных терминов, ссылающихся на один аспект (учет семантических отношений: синонимия, род-вид; использование метрик схожести фраз: расстояние Хэмминга, Левенштейна; машинное обучение).
- Определение тональности для:
  - мнения: классические подходы
  - аспекта: определяется как объединение тональности всех предложений, описывающих аспект; учитывают линейное расстояние, синтаксическое дерево
  - сущности в целом:
    * по аспектам: обобщаем информацию о тональности каждого аспекта
    * по сущности (аспект GENERAL): тональность определяется как объединение тональности всех предложений, описывающих сущность
- Обобщение полученной информации.

22 Лингвистические ресурсы и их назначение. Словари, тезаурусы, коллекции и корпуса текстов. Характеристики корпусов, виды разметки текстов. Семантические отношения в тезаурусах.

Лингвистические ресурсы предназначены для представления лингвистической информации для лингвистических процессоров.

В **словарь** входят:

- слова/словосочетания (термины, стоп-слова, дискурсные слова и выражения и т.д.)
- статистическая/лингвистическая информация (частота употребления, определения, связи между терминами и т.д.)

Способы составления словарей:

- ручной (работа с картотеками и информантами)
  Составление словарей вручную не лучший способ из-за быстрой изменчивости ЕЯ и субъективности автора
- автоматический по коллекциям текстов
  1) Предобработка коллекции текстов.
  2) Отбор кандидатов (слова/словосочетания, соответствующие шаблонам, N-граммы, коллокации).
  3) Присвоение кандидатам весов.
  4) Ранжирование по весам и отбор наиболее подходящих слов и словосочетаний.

  Проблема: корпус может неадекватно отражать употребление исследуемых единиц (низкая частота, неверное употребление и др.)

Тезаурус (или идеографический словарь):

- статьи упорядочены по смыслу, а не по алфавиту
- смысл понятия выражается посредством соотнесения его с другими понятиями и их группами, а не через определение
- описывает понятия определенной ПО, облегчает коммуникацию людей
- понятиям сопоставлены языковые единицы
- единицы с близкими значениями сгруппированы в концепты (понятия/дескрипторы), семантические отношения между концептами явно указаны

Виды

- типа Тезауруса Роже: цель — классификация слов ЕЯ для подбора синонимов и близких по смыслу слов, например, при написании текстов. Отношения, как правило, не специфицированы
- информационно-поисковые тезаурусы: цель — унифицированное описание содержания документов и/или поисковых запросов к автоматизированным ИП системам
- типа WordNet: цель — подробное описание лексической системы конкретного ЕЯ

Отношения

Ниже-выше, часть-целое, причина-следствие, сырье-продукт, административная иерархия, процесс-объект, процесс-субъект, функциональное сходство, свойство-носитель свойства, антонимия.

Корпусная лингвистика — теория и практика создания и использования корпусов текстов. Лингвистический, или языковой **корпус** — представительный массив языковых данных. Корпуса используются для решения лингвистических задач, построения приложений компьютерной лингвистики (КЛ).

Коллекция текстов — собрание текстов, объединенных каким-то общим признаком

(*Wikipedia, коллекция нормативно-правовых документов*).

Основные отличия коллекции от корпуса:

- коллекция решает нелингвистические задачи
- отбор текстов — на усмотрение составителей
- тексты рассматриваются не как образцы языковых явлений, а сами по себе
- тексты не имеют лингвистической разметки

Характеристики корпусов

- соответствие решаемой задаче
- представительный объем — типичность данных и полнота охвата изучаемых явлений
- естественность контекста данных — возможность их всестороннего и объективного изучения
- разметка по определенным правилам
- электронный вид и наличие корпусного менеджера — многократное использование данных при решении различных задач

Типы корпусов

- различие по языку корпуса
- параллельность (многоязычность)
- даты выхода текстов (синхронистический, диахронистисческий)
- стиль и жанр
- разметка текстов (полный текст, фрагменты)
- вид данных (письменные, речевые, мультимедийные)
- разметка и ее тип (морфологическая, синтаксическая и т.п.)
- доступность
- назначение
- динамичность (пополняемость)

Состав корпуса: массив данных с разметкой (собственно корпус) и корпусный менеджер, который обеспечивает поиск данных (слов и словосочетаний, контекстов употреблений) по шаблонам или с учетом различных типов разметки, получение статистической информации, отображение результатов в удобной форме, сохранение информации в различных форматах, быструю работу с большими объемами данных.

Виды разметки текстов

- экстралингвистическая (сведения об авторе и тексте: автор, название, дата создания)
- структурная (глава, предложение, токен)
- лингвистическая (морфологическая, синтаксическая, семантическая)

Разметка либо автоматическая с помощью соответствующих анализаторов, либо ручная (более качественная, снята омонимия). Сейчас в основном полуавтоматическая: сначала автоматическая разметка, потом ручная правка.

Проблемы корпусной лингвистики

- создание корпуса — трудоемкий процесс
- соблюдение сбалансированности — соответствие типов текстов их доле в языке
- соблюдение представительности — всестороннего отражения особенностей языка (исходя из задач)
- соблюдение масштабируемости — возможность увеличения размера корпуса
- решение проблем авторских прав
- приведение текстов к единому формату
- разработка ПО
- выбор разметки и стиля разметки

23 Процессная модель управления проектом: основные процессы и их взаимодействие. Особенности управления программным проектом.

Проект — временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.
Управление проектом — это приложение знаний, навыков, инструментов и методов к работам проекта для удовлетворения требований, предъявляемых к проекту. Это приложение знаний требует результативного управления процессами управления проектом.
Процесс — это набор взаимосвязанных действий и операций, осуществляемых для создания заранее определенного продукта, услуги или результата. Каждый процесс характеризуется своими входами, инструментами и методами, которые могут быть применены, а также результирующими выходами.
Процессы проекта можно разделить на две основные категории:

- процессы управления проектом обеспечивают результативное исполнение проекта
- процессы, ориентированные на продукт, определяют и создают продукт проекта

Процессы управления проектом разделяются на пять категорий, известных как группы процессов управления.
Группа процессов инициации состоит из процессов, выполняемых для определения нового проекта или новой фазы существующего проекта путем получения авторизации на начало проекта или фазы. Принимается решение, должен ли проект быть продолжен, приостановлен или отменен. В рамках процесса:

- определяется изначальное содержание и выделяются изначальные финансовые ресурсы
- определяются внутренние и внешние заинтересованные стороны
- выбирается руководитель проекта, если он еще не назначен

Данная информация закрепляется в уставе проекта и в реестре заинтересованных сторон. После утверждения устава проекта считается, что проект официально авторизован. Ключевая цель — привести в соответствие между собой ожидания заинтересованных сторон и цель проекта, дать заинтересованным сторонам наглядное представление о содержании и целях, показать, каким образом их участие в проекте и связанных с ним фазах может обеспечить удовлетворение их ожиданий.

Группа процессов планирования состоит из процессов, выполняемых для установления содержания работ, постановки и уточнения целей и определения направления действий, требуемых для достижения целей. Первоначальное планирование ограничивают по времени. План управления проектом и документы проекта, разрабатываемые как выходы группы процессов планирования, описывают все аспекты содержания, сроков, стоимости, качества, коммуникаций, человеческих ресурсов, рисков, закупок и вовлечения заинтересованных сторон.

Группа процессов исполнения состоит из процессов, выполняемых для исполнения работ, указанных в плане управления проектом, с целью соответствия спецификациям проекта. Во время исполнения проекта результаты могут потребовать внесения обновлений в план и принятия новых базовых планов. На осуществление процессов группы процессов исполнения затрачивается большая часть бюджета проекта.

Группа процессов мониторинга и контроля состоит из процессов, требуемых для отслеживания, анализа, а также координации прогресса и исполнения проекта; выявления областей, требующих внесения изменений в план; и инициирования соответствующих изменений. В проектах, состоящих из нескольких фаз, она координирует фазы проекта. Группа процессов мониторинга и контроля также включает в себя: контроль изменений

и разработку рекомендаций по применению корректирующих воздействий или предупреждающих действий для предотвращения возможных проблем; мониторинг соответствия текущих операций проекта плану управления проектом и базовому плану исполнения проекта; оказание влияния на факторы, которые могут действовать в обход интегрированного контроля изменений или управления конфигурацией, с тем чтобы в исполнение приводились только одобренные изменения.

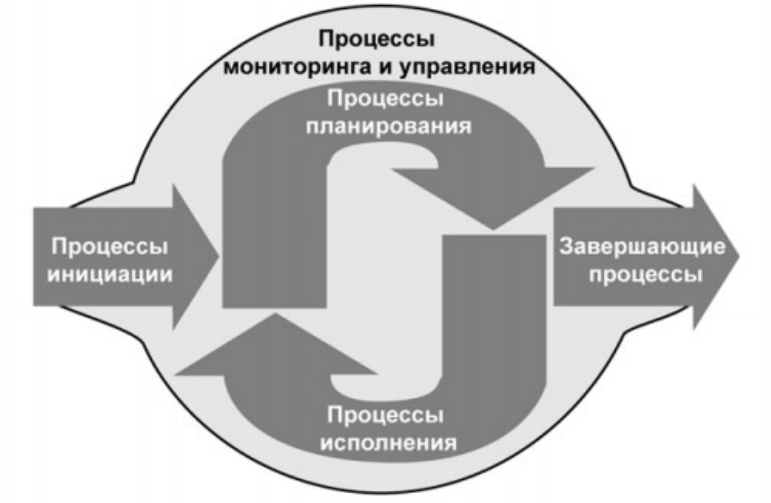
Группа процессов закрытия состоит из процессов, выполняемых для завершения всех операций в рамках всех групп процессов управления проектом в целях формального завершения проекта, фазы или договорных обязательств. Данная группа процессов также формально устанавливает преждевременное закрытие проекта.

При закрытии проекта или фазы может происходить следующее:

- получение подтверждения заказчика или спонсора для формального закрытия проекта или фазы
- проведение анализа после окончания проекта или фазы
- документирование последствий адаптации любого процесса, извлеченных уроков
- внесение необходимых обновлений в активы процессов организации
- архивация всех значимых документов проекта в информационной системе управления проектами
- завершение всех операций по закупке для закрытия всех соответствующих соглашений
- выполнение оценки всех членов команды и высвобождение ресурсов проекта

Общие взаимодействия процессов управления проектом

Процессы управления проектом представлены в качестве дискретных процессов с определенными границами. Однако на практике они накладываются друг на друга. Выход одного процесса, как правило, становится входом для другого процесса или является поставляемым результатом проекта, подпроекта или фазы проекта. Группы процессов не являются фазами жизненного цикла проекта. Если проект разделен на фазы, группы процессов взаимодействуют в рамках каждой фазы. Исходной идеей для взаимодействия между процессами управления проектом является цикл **планирование-исполнение-проверка-воздействие**. **Интегративный характер** управления проектом требует, чтобы группа процессов мониторинга и контроля взаимодействовала с другими группами процессов. **Итеративный характер** управления проектом означает, что процессы из любой группы могут быть повторно использованы на протяжении всего жизненного цикла проекта.



Особенности управления программным проектом

Программный продукт — совокупность программ и сопроводительной документации по их установке, настройке, использованию и доработке.

Процесс разработки ПО — совокупность процессов, обеспечивающих создание и развитие программного обеспечения. Чтобы программный проект стал успешным, необходимо:

1) Четко ставить цели.
2) Определять способ достижения целей.
3) Контролировать и управлять реализацией.
4) Анализировать угрозы и противодействовать им.
5) Создавать команду.

IT-проекты нацелены на развитие технологий. К особенностям относятся:

- набор работ
- статьи затрат
- быстрое моральное устаревание
- состав специалистов
- большой объем планирования

Жизненный цикл IT-проекта

Планирование проекта → Анализ требований к системе → Проектирование Системы → Разработка и настройка системы и переход к опытной эксплуатации → Переход на работу в новой системе (переход к опытно-промышленной эксплуатации) → Эксплуатация (переход к промышленной эксплуатации)

При инициализации следует учесть:

- влияние ИТ на бизнес-процессы
- влияние ИТ на пользователей
- как ИТ взаимодействуют с имеющимися Hard/Soft
- как другие компании используют эту технологию (решают эти проблемы)
- что из себя представляют поставщики ИТ и как поддерживают продукт
- возможность реализации проекта как в наилучших, так и в наихудших условиях (worst-best сценарии): план управления проектом при возникновении негативных событий

24 Сетевое планирование проекта. Примеры создания сетевых графиков и назначения ресурсов.

Создание иерархической структуры работ — процесс разделения результатов проекта и работ по проекту на меньшие элементы, которыми легче управлять. Нижний уровень — пакеты работ. **Работа, операция, задача** — элементарная, неделимая часть комплекса действий, выполняемых при реализации проекта. Работы связаны друг с другом зависимостям и определяющими порядок их выполнения относительно друг друга. Работы, задержка выполнения которых может отразиться на сроках выполнения проекта, называются критическими работами. Критические работы образуют самый длинный и продолжительный путь — **критический путь**.

Типы зависимостей работ:

- финиш-старт (FS): работа последователь может начаться только после окончания работы-предшественника
- финиш-финиш (FF): работа последователь может завершиться только после того как завершится работа-предшественник
- старт-старт (SS): работа последователь может начаться только после того как начнется работа-предшественник
- старт-финиш (SF): работа последователь может завершиться только после того как начнется работа-предшественник

Для задания временных интервалов между работами возможно использование **временного лага** — задержки между последователем и предшественником. Временной лаг между работами может иметь как положительное значение, так и отрицательное. Отрицательный лаг между работами моделирует начало работы последователя за некоторый промежуток времени до соблюдения заданных условий связи.

Сетевой график проекта — это инструмент, используемый для планирования, составления расписания и мониторинга хода выполнения проекта.

Сетевой план разрабатывается на основе информации, собранной для структуризации работ, и представляет графическую схему последовательности плана работ по проекту.

Сетевой график отражает операции проекта, которые необходимо выполнить, логическую последовательность и взаимозависимость этих операций и, в большинстве случаев, время начала и окончания самой продолжительной цепочки операций — критический путь. Сетевой график удобно отображать в виде ориентированного графа, представляющего собой совокупность вершин, соединенных между собой дугами. Образование петель и условные переходы от одной операции к другой не допускаются.

Диаграммы сети расписания проекта. Эти диаграммы обычно представлены в формате диаграммы «операции в узлах/дугах» или диаграммы Ганта.

Диаграмма Ганта — линейчатая диаграмма, в которой операции перечислены на вертикальной оси, даты приведены на горизонтальной оси, а длительности операций показаны в виде горизонтальных полос, расположенных в соответствии с датами старта и финиша.

Пример

Метод «операции в узлах» (метод предшествования)

Метод составления сетевых диаграмм, в которых плановые операции представляются прямоугольниками (или узлами). Плановые операции графически связаны одной или несколькими логическими взаимосвязями, которые показывают последовательность выполнения операций. **Временные параметры работ**: ранний/поздний срок начала/окончания, суммарный/свободный резерв времени.

Прямой анализ — определение ранних сроков начала операций. ES — **максимальное** значение EF из входящих работ, $EF = ES + Dur$.

Обратный анализ — определение поздних сроков завершения операций. LF — **минимальное** значение LS из исходящих работ, $LS = LF - Dur$.

Свободный резерв представляет разницу между LS и ES ($SL = LS - ES$) или между LF и EF ($SL = LF - EF$). **Критический путь** можно определить, как те операции, у которых $SL = 0$.

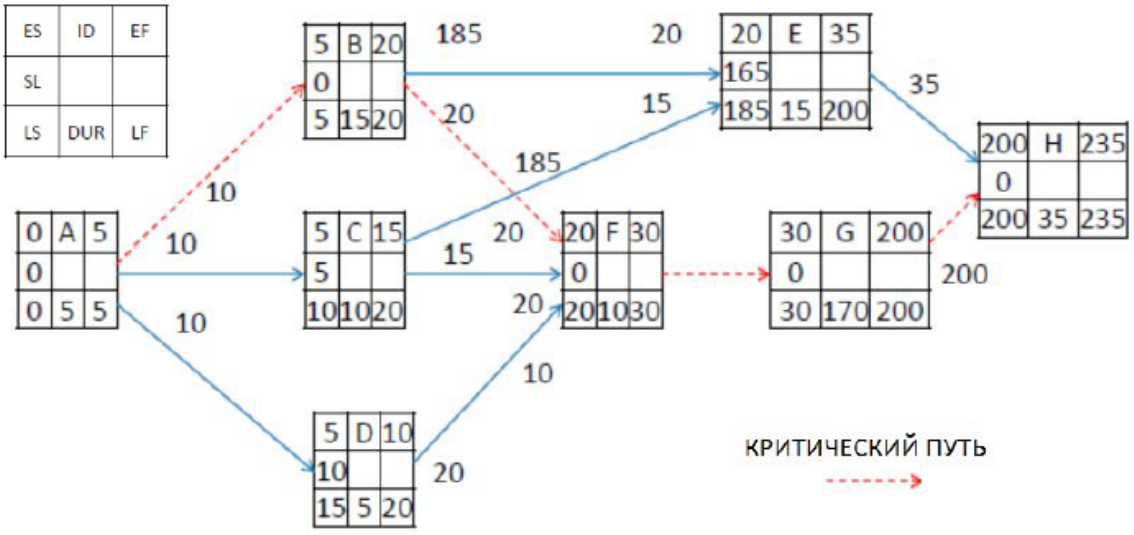
Преимущества

- не используются фиктивные операции
- не используются события
- легче начертить, если нет интенсивной зависимости между операциями
- значение операций понятно менеджерам первого уровня
- прослеживание пути упрощается за счет схемы нумерации операция/событие

Недостатки

- трудно определить путь по номеру операции. Если график не прикладывается, то в компьютерных распечатках должны быть указаны предшествующие и последующие шаги дли каждой операции.
- изображение и понимание сети затрудняется по мере увеличения масштабов графика

Пример. Даны Dur для всех операций, ES для первой (обычно 0) и LF для последней (обычно ее EF).



Метод «операции на дугах» (метод стрелочных диаграмм)

Метод построения сетевой диаграммы расписания, когда плановые операции отображаются на стрелках. Графики состоят из двух типов элементов: работ (представляются дугами) и событий (представляются вершинами).

Событие — это момент завершения какого-либо процесса, отражающий отдельный этап выполнения проекта. У любой работы есть начальное и конечное события. **Временные параметры событий**: ранний/поздний срок наступления, резерв времени. **Виды работ**:

действительные/фиктивные работы, ожидания, фиктивные работы. На графике должно быть только одно исходное и одно завершающее событие и не должно быть замкнутых петель. Любые два события должны быть непосредственно связаны не более чем одной работой.

Путем называется любая последовательность работ, в которой конечное событие каждой работы совпадает с начальным событием следующей за ней работы. **Полными путями** называются пути, начало которых совпадает с исходным событием сетевого графика, а конец — с завершающим. **Критическим путем** называется полный путь наибольшей продолжительности.

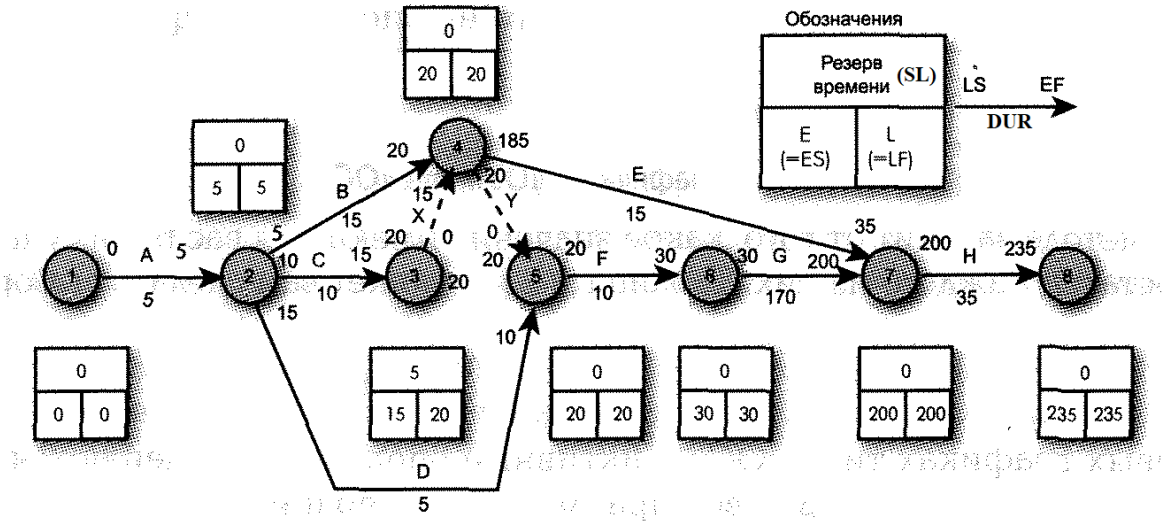
Преимущества

- путь упрощен благодари схеме нумерации операция/событие
- легче начертить график, если зависимости интенсивны
- ключевые события можно легко определить

Недостатки

- использование фиктивных операции увеличивает потребность в данных
- сосредоточение внимания на событиях может отвлечь от операций. Задержка операций может вызвать задержку событий и проектов

Пример. Даны Dur для всех операций, ES для первой (обычно 0) и LF для последней (обычно ее EF). X, Y — фиктивные операции. Критическии путь — A, B, Y, F, G, H.



Назначение ресурсов

Проблема планирования ресурсов

- Будет ли назначенная рабочая сила и/или оборудование соответствовать моему проекту и можно ли будет ими воспользоваться?
- Нужны ли внешние подрядчики?
- Существует ли непредвиденная зависимость от ресурсов?
- Есть ли новый критический путь?
- Достаточно ли гибкости в использовании ресурсов?
- Действительно ли реальным является срок сдачи проекта?

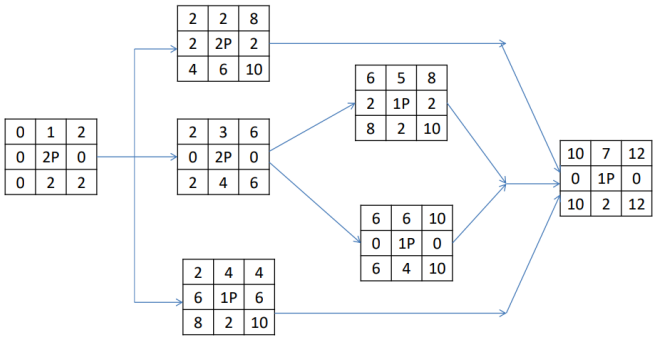
Ресурсы: люди, материалы, оборудование.

Параллельный метод – повторяющийся процесс, который начинается с начала жизни проекта и, когда требуемые ресурсы превосходят имеющиеся ресурсы, сохраняет операции по следующей схеме приоритетов:

1) минимальный резерв времени
2) наименьшая продолжительность

3) наименьший номер операции

Пример



ES	ID	EF
SL	RES	SL
LS	DUR	LF

Диаграмма планирования ресурсов по самому раннему возможному началу операции

ID	RES	DUR	ES	LF	TS	0	1	2	3	4	5	6	7	8	9	10	11
1	2P	2	0	2	0	2	2										
2	2P	6	2	10	2			2	2	2	2	2	2				
3	2P	4	2	6	0			2	2	2	2						
4	1P	2	2	10	6			1	1								
5	1P	2	6	10	2							1	1				
6	1P	4	6	10	0							1	1	1	1		
7	1P	2	10	12	0											1	1
Суммарное количество ресурсов						2	2	5	5	4	4	4	4	1	1	1	1

ID	RES	DUR	ES	LF	TS	0	1	2	3	4	5	6	7	8	9	10	11
1	2P	2	0	2	0	2	2										
2	2P	6	2 3	10	2 1			X									
3	2P	4	2	6	0			2	2	2	2						
4	1P	2	2	10	6			1	1								
5	1P	2	6	10	2												
6	1P	4	6	10	0												
7	1P	2	10	12	0												
Суммарное количество ресурсов						2	2	3	3	2	2	4	4	1	1	1	1
Имеющийся в распоряжении ресурс						3	3	3	3	3	3	3	3	3	3	3	3