Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.В. ГАВРИЛОВ

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Методические указания для студентов
заочной формы обучения АВТФ

Новосибирск
2004

В настоящем пособии рассматриваются основные понятия теории и прикладных систем искусственного интеллекта, методов представления и приобретения знаний, нейронных сетей, общения с компьютером на естественном языке. Даются рекомендации по выбору темы для контрольной работы.

Пособие предназначено для студентов заочного отделения АВТФ, обучающихся по специальностям 22.01 и 22.04 при изучении ими дисциплин "Системы искусственного интеллекта" и "Системы с искусственным интеллектом", соответственно, а также, может быть использовано студентами других специальностей для изучения родственных дисциплин.

Составитель: к.т.н., доцент *А.В. Гаврилов*

Рецензент: д.т.н., профессор *Е.В. Рабинович*

Работа подготовлена на кафедре вычислительной техники

ВВЕДЕНИЕ

Основной целью курса является формирование системного базового представления, первичных знаний, умений и навыков студентов по основам инженерии знаний и нейроинформатики, как двум направлениям построения интеллектуальных систем.

Ядро курса составляют методы представления и обработки знаний в системах искусственного интеллекта.

Для успешного изучения курса студенту необходимо знать основы информатики, теории вероятностей, математической логики, уметь программировать на каком-либо языке программирования.

1. ИСТОРИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ ИИ И ОСНОВНЫЕ ПОНЯТИЯ ИИ

Зарождение исследований в области искусственного интеллекта (ИИ). Два направления: логическое и нейрокибернетическое. Ранние исследования в 50-60-е годы (Н.Винер, Тьюринг, Мак-Каллок, Розенблатт, Саймон, Маккарти, Слэйджл, Сэмюэль, Гелернер, Н.Амосов). Появление первого развитого языка программирования LISP для построения систем ИИ. Появление в конце 60-х годов интегральных (интеллектуальных) роботов и первых экспертных систем. Успехи экспертных систем и застой в нейрокибернетике в 70-е годы. Новый бум нейрокибернетики в начале 80-х годов (модель Хопфилда). Появление логического программирования и языка PROLOG. Программа создания ЭВМ 5-го поколения. Стратегическая компьютерная инициатива США. Исследования по ИИ в СССР и России. Понятие знаний. Свойства знаний и отличие знаний от данных. Типы знаний: декларативные и процедурные, экстенсиональные и интенсиональные. Нечеткие знания. Виды и природа нечеткости. Проблема понимания смысла как извлечения знаний из данных и сигналов.

Литература – 2, 5, 6, 16.

С самого начала исследований в области моделирования процесса мышления (конец 40-х годов) выделились два до недавнего времени практически независимых направления: логическое и нейрокибернетическое.

Первое было основано на выявлении и применении в интеллектуальных системах различных логических и эмпирических приемов (эвристик), которые

применяет человек для решения каких-либо задач. В дальнейшем с появлением концепций "экспертных систем" (ЭС) (в начале 80-х годов) это направление вылилось в научно-технологическое направление информатики "инженерия знаний", занимающееся созданием т.н. "систем, основанных на знаниях" (Knowledge Based Systems). Именно с этим направлением обычно ассоциируется термин "искусственный интеллект" (ИИ).

Второе направление – нейрокибернетическое – было основано на построении самоорганизующихся систем, состоящих из множества элементов, функционально подобных нейронам головного мозга. Это направление началось с концепции формального нейрона Мак-Каллока-Питтса и исследований Розенблатта с различными моделями перцептрона – системы, обучающейся распознаванию образов. В связи с относительными успехами в логическом направлении ИИ и низким технологическом уровнем в микроэлектронике нейрокибернетическое направление было почти забыто с конца 60-х годов до начала 80-х, когда появились новые удачные теоретические модели (например, "модель Хопфилда") и сверхбольшие интегральные схемы.

Логическое направление можно рассматривать как моделирование мышления на уровне сознания или вербального или логического (целенаправленного) мышления. Его достоинствами являются:

- возможность относительно легкого понимания работы системы;
- легкость отображения процесса рассуждений системы на ее интерфейс с пользователем на естественном языке или каком-либо формальном языке;
- достижимость однозначности поведения системы в одинаковых ситуациях.

Недостатками этого подхода являются:

- трудность и неестественность реализации нечетких знаков (образов) (см. 2.4);
- трудность (или даже невозможность) реализации адекватного поведения в условиях неопределенности (недостаточности знаний, зашумленности данных, не точно поставленной цели и т.п.);
- трудность и неэффективность распараллеливания процесса решения задач.

Нейрокибернетическое направление (или нейроинформатика) можно рассматривать как моделирование образного мышления и мышления на подсознательном уровне (моделирование интуиции, творческого воображения, инсайта). Его достоинства – это отсутствие недостатков, свойственных логическому направлению, а недостатки – отсутствие его достоинств. Кроме того, в нейрокибернетическом направлении привлекает возможность (быть может, иллюзорная), задав базовые весьма простые алгоритмы адаптации и особенности структуры искусственной нейронной сети, получить систему, настраивающуюся на поведение сколь угодно сложное и адекватное решаемой задаче. Причем его сложность зависит только от количественных факторов модели нейронной сети.

Еще одним достоинством в случае аппаратной реализации нейронной сети является ее живучесть, т.е. способность сохранять приемлемую эффективность

решения задачи при выходе из строя элементов сети. Это свойство нейронных сетей достигается за счет избыточности. В случае программной реализации структурная избыточность нейронных сетей позволяет им успешно работать в условиях неполной или зашумленной информации.

Чем отличается понятие "знание" от понятия "данные" или "информация"? В последнее время ученые приходят к выводу, что наряду с веществом и энергией информация является объективно существующей неотъемлемой частью материального мира, характеризующей его упорядоченность (неоднородность) или структуру. Способность живых существ сохранять свою структуру (упорядоченность) в мире, где, вероятно, превалирует стремление к увеличению энтропии (однородности), обусловлена их способностью распознавать структуру окружающего мира и использовать результат распознавания (т.е. знания о мире) для целей выживания.

Таким образом, знания – это воспринятая живым существом (субъектом) информация из внешнего мира и в отличие от "информации" "знание" субъективно. Оно зависит от особенностей жизненного опыта субъекта, его истории взаимоотношения с внешней средой, т.е. от особенностей процесса его обучения или самообучения. На этом уровне абстракции знание уникально и обмен знанием между индивидуумами не может происходить без потерь в отличие от данных, в которых закодирована информация (неоднородность), и которые могут передаваться от передатчика к приемнику без потерь (не учитывая возможность искажения вследствие помех). Знание передается между субъектами посредством какого-либо языка представления знаний, наиболее типичным представителем которого является естественный язык. Создавая и используя естественный язык, человек с одной стороны стремился в нем формализовать и унифицировать знания для того, чтобы передавать их одинаковым образом наибольшему количеству людей с разным жизненным опытом, а с другой стороны, пытался дать возможность передавать все богатство личного знания. Первая тенденция привела к появлению различных формализованных специальных диалектов языка в различных областях знаний (математике, физике, медицине и т.д.). Вторая привела к появлению художественной литературы, в основе которой лежит стремление средствами языка вызвать ассоциации (переживания) в мозгу человека, т.е. заставить его думать и переживать на основе знаний, почерпнутых из прочтенного, и своих собственных знаний. По большому счету все разновидности искусства направлены на это – передачу знаний с использованием ассоциаций.

Если перейти от такого высокого уровня абстракции (философского) к более приземленному, то можно сравнивать знания и данные в их формализованном виде, что обычно и делается в литературе по искусственному интеллекту. Тогда можно сформулировать следующие отличия знаний от данных:

- знания более структурированы;
- в знаниях наибольшее значение имеют не атомарные элементы знаний (как в данных), а взаимосвязи между ними;

- знания более самоинтерпретируемы, чем данные, т.е. в знаниях содержится информация о том, как их использовать;
- знания активны в отличие от пассивных данных, т.е. знания могут порождать действия системы, использующей их.

Следует иметь в виду, что резкой границы между данными и знаниями нет, т.к. в последние двадцать лет разработчики систем управления базами данных все более делают их похожими на знания. Примером может служить применение семантических сетей (формализма для представления знаний) для проектирования баз данных, появление объектно-ориентированных баз данных, хранимых процедур (это делает в какой-то мере данные активными) и т.п. Таким образом, отличия знаний от данных, перечисленные выше, с развитием средств информатики сглаживаются.

В инженерии знаний различают следующие основные понятия о знаниях, заимствованные из семиотики – науки о знаковых системах:

- экстенсиональные знания – поверхностные или конкретные знания,
- интенсиональные знания – глубинные или абстрактные знания (знания о закономерностях),
- синтаксис – структура знаковой системы (данных или знаний),
- семантика – смысл знаковой системы (знаний), т.е. эквивалентное ее представление в другой парадигме представления знаний (внутренней),
- прагматика – цели, связанные со знаковой системой (например, цели или назначение предложения на естественном языке – команда, вопрос, пояснение и т.п.).

Контрольные вопросы

1. Чем отличаются знания от данных?

2. Основные свойства знаний.

3. В чем отличие нейрокибернетического подхода к построению ИИ от логического?

4. В чем недостатки логического/нейрокибернетического подхода?

5. Что такое "инженерия знаний"?

6. В чем идея теста Тьюринга на интеллектуальность компьютера или программы?

2. ПРИКЛАДНЫЕ СИСТЕМЫ ИИ

Прикладные системы ИИ – системы, основанные на знаниях. Понятие инженерии знаний. Экспертные системы. Их области применения и решаемые ими задач. Обобщенная структура экспертных систем. Интеллектуальные роботы. Их обобщенная структура. Системы общения на естественном языке и речевой ввод-вывод. Применение ИИ в системах управления производством. Применение ИИ в делопроизводстве и в сети Internet.

Литература – 1, 2, 5, 6, 10, 11, 16, 26.

Прикладные системы, основанные на знаниях, – это результат исследований в логическом направлении искусственного интеллекта. Методология и технология их создания изучаются и разрабатываются в так называемой инженерии знаний.

К прикладным системам ИИ можно отнести следующие имеющиеся сейчас на рынке ПО классы программных продуктов:

- экспертные системы;
- интеллектуальные роботы;
- системы интеллектуального анализа данных;
- системы речевого общения;
- системы распознавания текстов;
- системы идентификации личности по отпечаткам пальцев или по другим визуальным признакам;
- системы машинного перевода;
- диалоговые системы на естественном языке;
- системы поиска информации по запросу на естественном языке.

Экспертные системы – это прикладные системы ИИ, в которых база знаний представляет собой формализованные эмпирические знания высококвалифицированных специалистов (экспертов) в какой-либо узкой предметной области. Экспертные системы предназначены для замены при решении задач экспертов в силу их недостаточного количества, недостаточной оперативности в решении задачи или в опасных (вредных) для них условиях.

Обычно экспертные системы рассматриваются с точки зрения их применения в двух аспектах: для решения каких задач они могут быть использованы и в какой области деятельности. Эти два аспекта накладывают свой отпечаток на архитектуру разрабатываемой экспертной системы.

Можно выделить следующие основные классы задач, решаемых экспертными системами:

- диагностика;
- прогнозирование;
- идентификация;
- управление;
- проектирование (конфигурирование);
- мониторинг.

Наиболее широко встречающиеся области деятельности, где используются экспертные системы:

- медицина;
- вычислительная техника;
- военное дело;
- микроэлектроника;
- радиоэлектроника;
- юриспруденция;

- экономика;
- экология;
- управление технологическими процессами;
- геология (поиск полезных ископаемых).

Примеры широко известных и эффективно используемых (или использованных в свое время) экспертных систем:

DENDRAL – ЭС для распознавания структуры сложных органических молекул по результатам их спектрального анализа (считается первой в мире экспертной системой);

MOLGEN – ЭС для выработке гипотез о структуре ДНК на основе экспериментов с ферментами;

XCON – ЭС для конфигурирования (проектирования) вычислительных комплексов VAX 11 в корпорации DEC в соответствии с заказом покупателя;

MYCIN – ЭС диагностики кишечных заболеваний;

PUFF – ЭС диагностики легочных заболеваний;

MACSYMA – ЭС для символьных преобразований алгебраических выражений;

YES/MVS – ЭС для управления многозадачной операционной системой MVS больших ЭВМ корпорации IBM;

DART – ЭС для диагностики больших НМД корпорации IBM;

PROSPECTOR – ЭС для консультаций при поиске залежей полезных ископаемых;

POMME – ЭС для выдачи рекомендаций по уходу за яблоневым садом;

набор экспертных систем для управления планированием, запуском и полетом космических аппаратов многоразового использования;

AIRPLANE – экспертная система для помощи летчику при посадке на авианосец;

ЭСПЛАН – ЭС для планирования производства на Бакинском нефтеперерабатывающем заводе;

МОДИС – ЭС диагностики различных форм гипертонии;

МИДАС – ЭС для идентификации и устранения аварийных ситуаций в энергосистемах.

На рис. 1 изображена обобщенная структура экспертной системы.

База знаний предназначена для хранения экспертных знаний о предметной области, используемых при решении задач экспертной системой

База данных предназначена для временного хранения фактов или гипотез, являющихся промежуточными решениями или результатом общения системы с внешней средой, в качестве которой обычно выступает человек, ведущий диалог с экспертной системой.

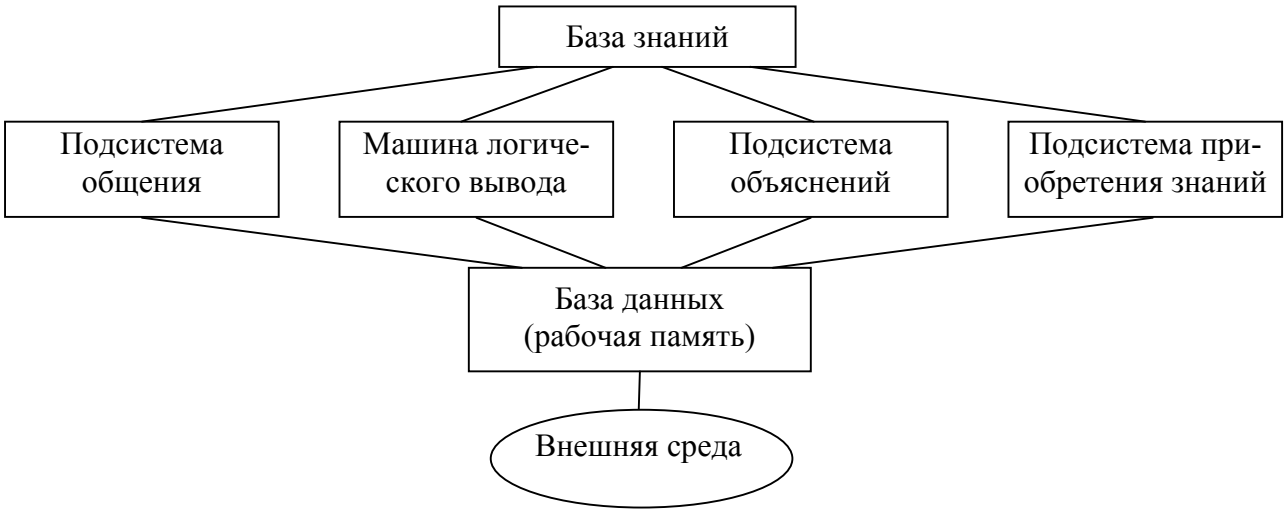

Рис. 1 . Структура экспертной системы

Машина логического вывода – механизм рассуждений, оперирующий знаниями и данными с целью получения новых данных из знаний и других данных, имеющихся в рабочей памяти. Для этого обычно используется программно реализованный механизм дедуктивного логического вывода (какая-либо его разновидность) или механизм поиска решения в сети фреймов или семантической сети.

Машина логического вывода может реализовывать рассуждения в виде:

1) дедуктивного вывода (прямого, обратного, смешанного);
2) нечеткого вывода;
3) вероятностного вывода;
4) унификации (подобно тому, как это реализовано в Прологе);
5) поиска решения с разбиением на последовательность подзадач;
6) поиска решения с использованием стратегии разбиения пространства поиска с учетом уровней абстрагирования решения или понятий, с ними связанных;
7) монотонного или немонотонного рассуждения;
8) рассуждений с использованием механизма аргументации;
9) ассоциативного поиска с использованием нейронных сетей;
10) вывода с использованием механизма лингвистической переменной.

Подсистема общения служит для ведения диалога с пользователем, в ходе которого ЭС запрашивает у пользователя необходимые факты для процесса рассуждения, а также, дающая возможность пользователю в какой-то степени контролировать и корректировать ход рассуждений экспертной системы.

Подсистема объяснений необходима для того, чтобы дать возможность пользователю контролировать ход рассуждений и, может быть, учиться у экспертной системы. Если нет этой подсистемы, экспертная система выглядит для пользователя как "вещь в себе", решениям которой можно либо верить, либо нет. Нормальный пользователь выбирает последнее, и такая ЭС не имеет перспектив для использования.

Подсистема приобретения знаний служит для корректировки и пополнения базы знаний. В простейшем случае это – интеллектуальный редактор базы знаний, в более сложных экспертных системах – средства для извлечения знаний из баз данных, неструктурированного текста, графической информации и т.д.

Обобщенная структура задач, решаемых системой управления интеллектуального мобильного робота показана на рис. 2.

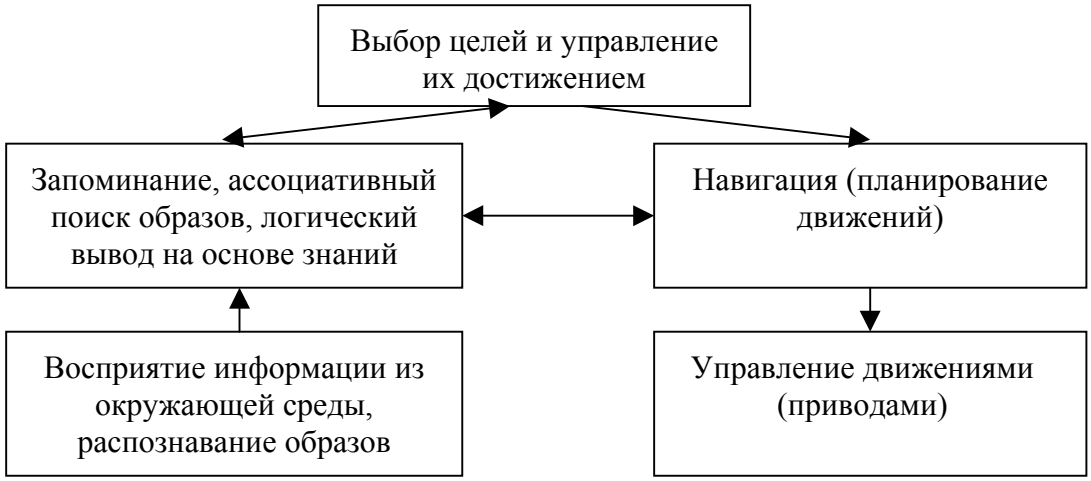

Рис. 2. Структура задач, решаемых системой управления мобильного робота

Система управления мобильного робота в общем случае выполняет следующие функции:

1) восприятие и распознавание информации, поступающей из внешнего мира от датчиков;
2) общение с человеком;
3) создание и корректировка модели мира путем обучения в процессе общения с человеком, восприятия сигналов с датчиков и выполнения действий;
4) планирование и перепланирование поведения;
5) управление выполнением действий;
6) управление приводами;
7) общение с другими роботами.

Контрольные вопросы

1. Какие Вы знаете прикладные системы искусственного интеллекта?
2. Что такое экспертная система?
3. Что такое база знаний?
4. Области применения экспертных систем.
5. Задачи, решаемые экспертными системами.
6. Примеры экспертных систем.
7. Области применения интеллектуальных роботов.
8. Задачи, решаемые мобильными роботами.

3. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ И РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

3.1. Логика предикатов 1-го порядка как метод представления знаний

Логические и эвристические методы представления знаний. Понятие предиката, формулы, кванторов всеобщности и существования. Интерпретация формул в логике предикатов 1-го порядка. Логика Хорна как основа языка логического программирования Prolog. Недостатки логики 1-го порядка как метода представления знаний.

Литература – 2, 5, 6, 9, 15, 16, 18, 24.

Логика предикатов является развитием алгебры логики (или логики высказываний). В логике высказываний для обозначения фактов используются буквы (имена или идентификаторы или фразы), не имеющие структуры (используемые как атомарные объекты), и принимающие значения "1" или "0" ("да" или "нет"). То, что фразы имеют атомарный характер, не позволяет обнаружить похожесть их смысла. Например, высказывания "расстояние от Земли до Солнца – 150 млрд. км" и "расстояние от Земли до Марса – 60 млн. км" имеют похожий смысл, но абсолютно разные в логике высказываний.

В логике предикатов факты обозначаются n-арными логическими функциями – предикатами $F(x_1,x_2,...,x_m)$, где F – имя предиката (функтор) и x_i – аргументы предиката. Имена предикатов неделимы, т.е. являются так называемыми атомами. Аргументы могут быть атомами или функциями $f(x_1,x_2,...,x_m)$, где f – имя функции, а $x_1,...,x_m$, так же как и аргументы предикатов являются переменными или константами предметной области. В результате интерпретации (подругому, конкретизации) предиката функторы и аргументы принимают значения констант из предметной области (строк, чисел, структур и т.д.). При этом следует различать интерпретацию на этапе описания предметной области (создания программ и баз знаний) и на этапе решения задач (выполнения программ с целью корректировки или пополнения баз знаний). В дальнейшем при работе с предикатами мы будем иметь дело с результатом их интерпретации в первом смысле, т. е. с их привязкой к некоторой предметной области.

Выше приведенные примеры высказываний в виде предикатов будут выглядеть как "расстояние(Земля, Солнце, 150000000000)" и "расстояние(Земля, Марс, 60000000)". Так как они имеют определенную структуру, их можно сравнивать по частям, моделируя работу с содержащимся в них смыслом.

Предикат с арностью $n > 1$ может используется в инженерии знаний для представления n-арного отношения, связывающего между собой n сущностей (объектов) – аргументов предиката. Например, предикат отец("Иван", "Петр Иванович") может означать, что сущности "Иван" и "Петр Иванович" связаны родственным отношением, а именно, последний является отцом Ивана или наоборот. Уточнение семантики (смысла) этого предиката связано с тем, как он

используется, т.е. в каких операциях или более сложных отношениях он участвует, и какую роль в них играют его 1-й и 2-й аргументы. Предикат "компьютер(память, клавиатура, процессор, монитор)" может обозначать понятие "компьютер" как отношение, связывающее между собой составные части компьютера, предикат "внутри(процессор_Pentium, компьютер)" – то, что внутри компьютера находится процессор Pentium.

Предикат с арностью $n = 1$ может представлять свойство сущности (объекта), обозначенного аргументом или характеристику объекта, обозначенного именем предиката. Например, кирпичный(дом), оценка(5), улица("Красный проспект"), дата_рождения("1 апреля 1965 г."), быстродействие("1 Мфлопс").

Предикат с арностью $n = 0$ (без аргументов) может обозначать событие, признак или свойство, относящееся ко всей предметной области. Например, "конец работы".

При записи формул (выражений) помимо логических связок "конъюнкция" (&), "дизъюнкция" ($\vee$), "отрицание" ($\neg$), "следование" ("импликация") ($\rightarrow$) , заимствованных из логики высказываний , в логике предикатов используются кванторы всеобщности ($\forall$) и существования ($\exists$). Например, выражение $\forall(x,y,z)$ (отец(x,y) & (мать(x,z)) -> родители(x,y,z) означает, что для всех значений x,y,z из предметной области справедливо утверждение "если y – отец и z – мать x, то y и z – родители x; выражение ($\exists x$) (студент(x) & должность(x,"инженер") означает, что существует хотя бы один студент, который работает в должности инженера.

Переменные при кванторах называются связанными переменными в отличии от свободных переменных. Например, в выражении

($\forall x$) (владелец(x,y) $\rightarrow$ частная_собственность(y))

x – связанная переменная, y – свободная переменная.

Логика предикатов 1-го порядка отличается от логик высших порядков тем, что в ней запрещено использовать выражения (формулы) в качестве аргументов предикатов.

Решение задач в логике предикатов сводится к доказательству целевого утверждения в виде формулы или предиката (теоремы), используя известные утверждения (формулы) или аксиомы.

В конце 60-х годов Робинсоном для доказательства теорем в логике предикатов был предложен метод резолюции, основанный на доказательстве "от противного". Целевое утверждение инвертируется, добавляется к множеству аксиом и доказывается, что полученное таким образом множество утверждений является несовместным (противоречивым). Для выполнения доказательства методом резолюции необходимо провести определенные преобразования над множеством утверждений, а именно, привести их к совершенной конъюнктивной нормальной форме (СКНФ). СКНФ представляет собой набор (конъюнкцию) дизъюнктов без кванторов. Кванторы всеобщности подразумеваются, а кванторы существования заменяются на перечисление формул (или предикатов) со всеми константами из предметной области, для которых формула истинна. Например, "отец(Иван, Петр)", "отец(Иван, Степан)" и т.д.

Логика предикатов 1-го порядка легла в основу языков логического программирования, самым распространенным из которых является Prolog (различные его диалекты). Точнее, язык Prolog основан на модифицированной логике предикатов 1-го порядка (логике Хорна или логике дизъюнктов). Логика Хорна отличается от классической логики предикатов 1-го порядка тем, что она оперирует уже почти преобразованными к применению метода резолюции формулами без кванторов всеобщности и существования, представляющими собой множество дизъюнктов (предложений или клауз Хорна). "Почти" объясняется тем, что клаузы Хорна содержат импликацию и выглядят как

$$A \to B,$$

где A – предикат;

B – предикат или конъюнкция или дизъюнкция предикатов (такое представление части B предложения возможно, т.к. конъюнкция и дизъюнкция рассматриваются как частные случаи предикатов.

Доказательство некоторого утверждения (целевого предиката) в логическом программировании сводится к процессу унификации, с помощью которого происходит рекурсивный перебор всех возможных подстановок значений переменных в целевом предикате, управляемый ограничениями, заданными множеством предложений. Множество предложений обычно в Прологе называется базой данных Пролога. База данных состоит из предложений-правил вывода вида $A \to B$ и предложений-фактов, представляющих собой отдельные предикаты. При этом в предикатах-фактах параметрами могут быть только константы, а в предикатах-правилах – константы и неконкретизированные (неопределенные) переменные. Последнее относится и к целевому предикату. В этом случае, если параметром является переменная, то это означает, что ее значение необходимо найти при доказательстве целевого предиката.

Унификация основана на сравнении (сопоставлении с образцом) целевого предиката, который надо доказать, с предикатами-фактами и предикатами-правилами из базы данных Prolog-программы. При этом успешность сопоставления двух предикатов определяется следующими условиями, упорядоченными в порядке их проверки:

- имена предикатов совпадают;
- количество параметров у предикатов совпадает;
- каждая пара сравниваемых параметров сопоставима.

Последнее условие для некоторой пары параметров истинно при трех вариантах:

- параметры являются константами (любого типа) и они равны;
- один параметр из пары является константой, а другой – переменной, в этом случае переменной присваивается значение константы;
- оба параметра являются неконкретизированными переменными, в этом случае эти переменные становятся "связанными", т.е. в дальнейшем при интерпретации программы рассматриваются как одна и та же переменная.

При унификации целевого предиката с правилом, сопоставлению подвергается сначала левая часть правила, а затем, в случае успешной унификации, последовательно проверяются предикаты, находящиеся в правой части. Т.е. правило в Прологе с точки зрения унификации рассматривается как предикат с именем ":-" (импликация) и с параметрами *A* и *B*, а *B* рассматривается в свою очередь как предикат "," (конъюнкция) или ";" (дизъюнкция) с параметрами-предикатами правой части правила.

К недостаткам логики предикатов 1-го порядка как метода представления знаний можно отнести следующее:

- монотонность логического вывода, т.е. невозможность пересмотра полученных промежуточных результатов (они считаются фактами, а не гипотезами);
- невозможность применения в качестве параметров предикатов других предикатов, т.е. невозможность формулирования знаний о знаниях;
- детерминированность логического вывода, т.е. отсутствие возможности оперирования с нечеткими знаниями.

Но логику предикатов 1-го порядка можно использовать как основу для конструирования более сложных и удобных логических методов представления знаний. В этом качестве она используется в модальных и псевдофизических логиках.

Контрольные вопросы

1. Какие преимущества имеет логика предикатов 1-го порядка как метод представления знаний по отношению к логике высказываний.

2. Чем логика предикатов 1-го порядка отличается от логик высших порядков?

3. Как можно представить в виде формулы логики предикатов 1-го порядка высказывание "Все люди смертны и обладают желаниями"?

4. В чем недостатки логики предикатов 1-го порядка как метода представления знаний?

5. В чем отличается логика Хорна от логики предикатов 1-го порядка?

6. Почему в языке Prolog (в логике Хорна) отсутствуют кванторы всеобщности и существования, и чем они заменяются?

7. Что такое "унификация" в языке Prolog?

3.2. Нечеткие и псевдофизические логики

Теория нечетких множеств – основа псевдофизических логик. Нечеткая логика. Понятие лингвистической переменной. Примеры псевдофизических логик: пространственная и временная логики.

Литература – 6, 8, 13, 15.

Для представления нечетких понятий и оперирования с ними американский ученый Л.Заде в 60-х годах разработал теорию нечетких множеств, а затем – нечеткую логику, базирующуюся на ней. В основе теории нечетких множеств

лежит интерпретация факта принадлежности элемента a множеству A как факта, который может быть истинным или ложным с некоторой оценкой истинности $\mu_A(a)$, пробегающей значения от 0 до 1. Эта оценка истинности называется функцией принадлежности элемента a множеству A.

Операции включения и равенства в теории нечетких множеств определяются обычно следующим образом (по Л.Заде):

$$F \subseteq G:$$
$$\forall a, \mu_F(a) \le \mu_G(a)$$
$$F = G:$$
$$\forall a, \mu_F(a) = \mu_G(a).$$

Дополнение множества F к G определяется так, что

$$\forall a, \mu_F(a) = 1 - \mu_G(a).$$

Пересечение и объединение множеств определяются следующим образом:

$$\forall a, \mu_{F \cap G}(a) = \min(\mu_F(a), \mu_G(a)),$$
$$\forall a, \mu_{F \cup G}(a) = \max(\mu_F(a), \mu_G(a)).$$

Эти определения не единственные, хотя они не противоречат интуитивным представлениям о соответствующих операциях над нечеткими множествами. Частным случаем теории нечетких множеств (при $\mu = 1$ или 0) является классическая теория множеств. Однако встречаются и другие определения операций над нечеткими множествами.

Так же как на основе классической теории множеств строится двоичная (булева) логика, так и на базе теории нечетких множеств строится теория нечетких множеств. Она оперирует с высказываниями, для которых функция принадлежности, описанная ранее, определена на множестве истинных высказываний. Функция принадлежности интерпретируется как мера истинности, уверенности или достоверности и отражает нечеткость знаний.

Предположим, существуют следующие высказывания:

"Иванов – хороший человек" с μ =0.8,
"Политик – хороший человек" с μ = 0.3.

Тогда конъюнкция этих двух высказываний (имеющая смысл как уточнение мнения об Иванове, когда стало известно, что он – политик) определяется функцией принадлежности μ = 0.3, а дизъюнкция – μ = 0.8.

Можно развить логику нечетких высказываний до логики нечетких предикатов, которая обычно рассматривается в рамках псевдофизических логик (см. ниже).

В теории нечетких множеств функция принадлежности может интерпретироваться как субъективное представление об истинности высказываний или объективная нечеткость знаний (информации). В первом случае описание нечетких высказываний является как бы снимком состояния некоторой интеллек-

туальной системы, обученной на примерах взаимодействия с внешней средой или заполненной субъективными знаниями экспертов. Во втором случае нечеткость является следствием каких-либо помех при поступлении информации в систему и интерпретации ее в виде знаний. В обоих случаях функцию принадлежности можно интерпретировать как вероятностную меру истинности и применять теорию вероятности к ее обработке и анализу. Это справедливо, т.к. интеллектуальная система работает с множеством разных субъектов, имеющих разные субъективные представления об истинности высказываний, или с множеством разных ситуаций, в которых разные помехи создают вероятностное описание истинности информации (знаний).

Недостатки классической логики и основанной на ней логики предикатов первого порядка как метода представления знаний об окружающем мире привели к появлению псевдофизических логик. В их основе лежит представление нечетких или размытых понятий в виде так называемых лингвистических переменных, придуманных Заде [9] для того, чтобы приблизить семантику (смысл) денотата (знака) к семантике, которая вырабатывается в мозгу человека в процессе его обучения (опыта). Для этого множество образов (десигнатов), с которыми должна оперировать интеллектуальная система, представляется в виде точек на шкалах. Например, можно рассматривать шкалы "возраст" (в годах), "расстояние до объекта" (в м или км) и т.п. С каждой шкалой связано множество знаковых значений лингвистической переменной. Например, со шкалой "возраст" могут быть связаны следующие значения одноименной лингвистической переменной: "юный", "молодой", "зрелый", "пожилой", "старый", "дряхлый". Со шкалой "расстояние" – "вплотную", "очень близко", "близко", "рядом", "недалеко", "далеко", "очень далеко", "у черта на куличиках". Взаимосвязь между этими двумя представлениями (множеством точек на шкале и множеством знаковых значений) задается с помощью функции принадлежности $\mu_x(t)$, где x – значение лингвистической переменной, t – значение на шкале. Значение функции принадлежности интерпретируется как вероятность того, что значение t на шкале можно заменить знаком x или наоборот. Очевидно, что можно пронормировать значения функции принадлежности в соответствии с формулой

$$\sum_x \mu_x(t) = 1$$

или в соответствии с

$$\sum_t \mu_x(t) = 1$$

На рис. 3 приведен пример описания лингвистической переменной возраст. Здесь каждая кривая описывает ее одно символьное значение.

Наиболее используемыми псевдофизическими логиками являются пространственная, временная и каузальная (причинно-следственная).

Пространственная логика может быть разбита на логики статическая и динамическая, взаимного расположения объектов, расположения объектов в пространстве (расстояний и направлений).

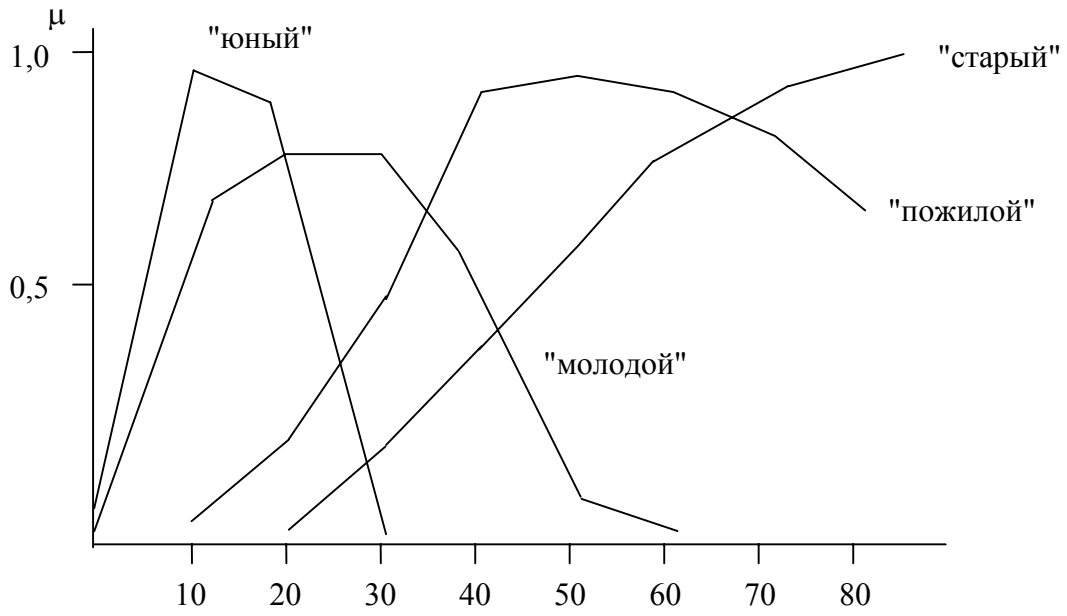

Рис. 3. Описание лингвистической переменной "Возраст"

Логики взаимного расположения объектов, расстояний и направлений делятся на метрическую и топологическую логики. В отличие от метрической топологическая логика не связана с метрической шкалой.

Метрические шкалы подразделяются на экзоцентрические и эндоцентрические, относительные и абсолютные. Экзоцентрические шкалы имеют началом координат точку, связанную с самой интеллектуальной системой. Примером такой шкалы является шкала для описания лингвистической переменной "Расстояние до объекта" в логике расстояний. Ее символьными значениями могут быть следующие: "совсем рядом", "рядом", "очень-очень близко", "очень близко", "близко", "не очень близко", "не близко", "недалеко", "не далеко, но и не очень близко", "не очень далеко", "неблизко", "далеко", "очень далеко" и т.п. Эндоцентричекая шкала имеет началом координат точку вне системы. Примером такой шкалы является шкала для описания лингвистической переменной "расстояние между двумя объектами" в той же логике расстояний. Относительные шкалы имеют изменяемую точку отсчета (начало координат), а абсолютные – неизменную (обычно, подразумеваемую, т.е. явно не заданную).

Логика направлений оперирует с понятиями "справа", "слева", "впереди", "сзади" или "на восток", "на запад" и т.п.

В логике взаимного расположения объектов описываются следующие базовые отношения: унарные – "иметь горизонтальное положение", "иметь вертикальное положение", бинарные – "находиться внутри", "находиться вне", находиться на поверхности", "находиться в центре", "находиться в середине", "быть там же, где..", "быть ненулевой проекцией", "находиться в ε-окрестности",

"быть частью", "находиться на одной прямой", "находиться вокруг", "быть на краю", "быть параллельно", "быть перпендикулярно", "быть симметрично", "находиться в n единицах от..", "иметь точку опоры на..", "иметь точку подвеса на..", "соприкасаться", "быть выше", "быть ниже", "находиться на одинаковом уровне", "быть дальше", "быть ближе", "быть равноудаленными", n-арное отношение – "быть между".

Из базовых отношений с помощью логических связок строятся производные отношения, такие как "не соприкасаться" (отрицание "соприкасаться"), "быть вместе.." (следствие от "находиться там же.."), "висеть" (конъюнкция "иметь вертикальное положение" и "висеть на…"), "стоять" (конъюнкция "иметь вертикальное положение" и "иметь точку опоры на..") и т.п. Эти отношения описываются в виде правил, определяющих с помощью импликации сложное отношение через базовые. Кроме того, в псевдофизической логике в виде правил описываются свойства отношений и взаимосвязи между ними. Например, свойства рефлексивности (например, рядом(x,x)), симметричности (например, рядом(x,y)→рядом(y,x)) и транзитивности (например, выше(x,y)→выше(x,z)&выше(z,y)).

Временная псевдофизическая логика имеет дело с отношениями "происходить одновременно", "пересекаться во времени" (n-арные), "быть раньше", "быть позже" (бинарные), "давно", "недавно", "скоро" (унарные) и т.п.

Контрольные вопросы

1. В чем причины (природа) нечеткости знаний?

2. В чем отличие теории нечетких множеств от классической теории множеств?

3. Что такое "функция принадлежности" и как она может интерпретироваться?

4. Что такое "лингвистическая переменная"?

5. Опишите в виде лингвистической переменной понятие "стоимость компьютера".

6. Как вычисляются функции принадлежности конъюнкции и дизъюнкции?

7. Какие Вы знаете псевдофизические логики?

8. Примеры отношений пространственной псевдофизической логики.

9. Нечеткие понятия, которые могут встречаться в пространственной/временной псевдофизической логике.

3.3. Правила-продукции

Структура правил-продукций. Типы ядер правил-продукций и варианты их интерпретаций. Методы логического вывода: прямой и обратный. Стратегии выбора правил при логическом выводе. Методы представления и обработки нечетких знаний в продукционных системах. Достоинства и недостатки правил-продукций как метода представления знаний.

Литература – 1, 2, 5, 6, 15, 16, 18, 24, 26.

Правило-продукция (или просто правило) в общем случае можно представить в виде

$$<I, S, P, A \rightarrow B, F>,$$

где: I – идентификатор правила (обычно порядковый номер);
S – область применимости;
P – условие применимости;
A – посылка правила;
B – заключение;
F – постусловие правила.

$A \rightarrow B$ является ядром правила-продукции и может по-разному интерпретироваться. Наиболее часто используемая форма интерпретации – логическая, при которой A является множеством элементарных условий, связанных логическими связками "И", "ИЛИ" и "НЕТ", B – множеством элементарных заключений. При этом правило считается сработавшим (выполняется заключение B), если посылка A истинна. Другой формой интерпретации ядра является вероятностная интерпретация, при которой правило срабатывает с некоторой вероятностью, зависящей от истинности посылки.

В качестве заключения обычно применяется операция добавления факта в базу данных интеллектуальной системы с указанием меры достоверности получаемого факта. В качестве постусловия могут использоваться какие-либо дополнительные действия или комментарии, сопровождающие правило.

Обычно при описании баз знаний или экспертных систем правила представляются в более наглядном виде, например:

ПРАВИЛО 1:
ЕСЛИ
Образование=Высшее И
Возраст=Молодой И
Коммуникабельность=Высокая
ТО
Шансы найти работу=Высокие КД=0.9.

При срабатывании этого правила в базу данных интеллектуальной системы (например, экспертной системы) добавляется факт, означающий, что шансы найти работу высоки с достоверностью 0.9 или 90 % (значение коэффициента достоверности КД). Понятия "Образование", "Возраст", "Коммуникабельность" служат для задания условия (в данном случае, конъюнкции), при котором срабатывает правило.

Факты хранятся в базе данных продукционной системы в форме

(Объект, значение, КД)

или

(Объект, атрибут, значение, КД).

Но могут использоваться и другие структуры для хранения фактов, такие как семантические сети или фреймы (см. 2.6 и 2.7). В этом случае говорят о

комбинации разных методов представления знаний или о гибридных интеллектуальных (экспертных) системах. При интерпретации (выполнении) правила в ходе проверки условия система проверяет факты, находящиеся уже в базе данных, и, если соответствующего факта нет, обращается за ним к источнику данных (пользователю, базе данных и т.д.) с вопросом (или запросом).

Кроме правил в продукционных базах знаний могут использоваться метаправила для управления логическим выводом. Пример метаправила для гипотетической базы знаний, пример из которой был приведен ранее:

ЕСЛИ
    Экономика = развивается
ТО
    Увеличить приоритет правила 1

Для представления нечетких знаний факты и правила в продукционных системах снабжаются коэффициентами достоверности (или уверенности), которые могут принимать значения из разных интервалов в разных системах (например, <0,1>, <0, 100>, <-1,+1>). Во втором случае можно говорить об уверенности в процентах, а в последнем случае – о задании коэффициентом уверенности меры ложности или истинности факта.

Существуют разные методы обработки нечеткости при интерпретации правил. Обычно для оценки истинности условия используются правила нечеткой логики (см. 2.3). Для оценки истинности факта, полученного при срабатывании правила, обычно также используется правило из нечеткой логики для оценки конъюнкции, аргументами которой являются условие и факт в заключении со своими коэффициентами достоверности.

Более разнообразные подходы для оценки истинности используются при формировании правилом факта, уже существующего в базе данных интеллектуальной системы. Ниже приводятся формулы, используемые в таком случае в экспертной системе MYCIN (в ней коэффициент принадлежности пробегает значения из интервала <–1,+1>):

$$КД = \begin{cases} ИП + РП(1 - ИП); ИП, РП > 0 \\ -(|ИП| + |РП|(1 - |ИП|); ИП, РП < 0 \\ \dfrac{|ИП| + |РП|}{1 - \min(|ИП|, |РП|)}; ИП * РП < 0, \end{cases}$$

где: КД – новое значение факта,
ИП – показатель истинности уже существующего факта (исходный показатель),
РП – показатель факта, формируемый исходя из истинности условия и заключения правила (результирующий показатель).

Легко проверить, что получающиеся значения не входят в противоречие с интуитивным представлением, о том, как должна меняться истинность факта при срабатывании правила, подтверждающего или опровергающего его.

Для решения задач в продукционной интеллектуальной системе существует два основных метода дедуктивного логического вывода: обратный и прямой. Может использоваться и комбинация этих двух методов. При обратном логическом выводе процесс интерпретации правил начинается с правил, непосредственно приводящих к решению задачи. В них в правой части находятся заключения с фактами, являющимися решением (целевыми фактами). При интерпретации этих правил в процесс решения могут вовлекаться другие правила, результатом выполнения которых являются факты, участвующие в условиях конечных правил и т.д.

В самом общем виде алгоритм обратного логического вывода, записанный на псевдокоде в виде функции, выглядит так (условие ограничено конъюнкцией элементарных условий):

```
функция Доказана_Цель(Цель): boolean;
Поместить Цель в стек целей.
пока стек целей не пуст
 цикл
        Выбор цели из стека целей и назначение ее текущей.
        Поиск множества правил, в правой части которых
         находится текущая цель (множества подходящих правил).
        Считать, что Цель не доказана.
        пока множество походящих правил не пусто
         и Цель не доказана
         цикл
         Выбор из этого множества одного текущего правила
         с использованием определенной стратегии.
         Считать текущим элементарным условием первое.
                пока не проверены все элементарные условия правила
                 и не надо прервать проверку условия
                 цикл
                        если в текущем элементарном условии
                                участвует факт,
                                встречающийся в правой части
                                какого-то правила
                                то
                                 если не Доказана_Цель(Этот факт)
                                        то
                                         Надо прервать проверку условия
                                 конец_если
                иначе
                 Запросить информацию о факте.
```

Проверить элементарное условие.
если элементарное условие истинно
то
Добавить факт в базу данных.
Перейти к следующему элементарному
Условию.
иначе
Надо прервать проверку условия.
конец_если
конец_если
конец_цикла
если условие правила истинно
то
Выполнить заключение.
Исключить Цель из стека целей.
Считать, что Цель доказана.
конец_если
конец_цикла
конец_цикла
конец_функции.

Существует много различных стратегий выбора правила из подходящих. Наиболее простой и часто встречающейся стратегией является "первая попавшаяся". При этой стратегии решение задачи зависит от порядка расположения (перебора) правил в базе знаний. Другие используемые стратегии выбора:

"стопки книг" (LIFO),
"по приоритету",
"наиболее свежие данные",
"наиболее длинное условие".

В прямом методе логического вывода интерпретация правил начинается от известных фактов, т.е. сначала выполняются правила, условия которых можно проверить с использованием фактов, уже находящихся в базе данных.

В общем виде алгоритм прямого вывода выглядит так:

пока Цель не доказана
цикл
Формирование множества подходящих правил
(по их условиям и наличию фактов).
Выбор одного правила из этого множества (с использованием
определенной стратегии выбора).
Считать текущим элементарным условием первое.
пока не проверены все элементарные условия правила
и не надо прервать проверку условия
цикл

если элементарное условие истинно
то
Перейти к следующему элементарному условию.
иначе
Надо прервать проверку условия.
конец_если
конец_цикла
Выполнить заключение.
если при формировании заключения появился целевой факт
то
Считать, что Цель доказана.
конец_если
конец_цикла.

Метод прямого логического вывода можно применять тогда, когда факты появляются в базе данных не зависимо от того, какую задачу сейчас требуется решить (какой целевой факт доказать) и в разные моменты времени. В этом случае можно говорить о том, что факты управляют логическим выводом (решением задачи). Кроме того, этот метод целесообразно применять для формирования вторичных признаков (фактов) из первичных для подготовки решения задачи в дальнейшем с применением обратного логического вывода.

Метод обратного логического вывода можно применять тогда, когда необходимо минимизировать количество обращений к источнику данных (например, пользователю), исключив из рассмотрения заведомо ненужные для решения задачи факты.

Достоинствами продукционного метода представления знаний являются следующие.

1. Наглядность и понятность знаний (по крайней мере, на уровне одного правила).
2. Возможность реализации немонотонного логического вывода и обработки противоречивых фактов.
3. Возможность введения различных модификаций в интерпретацию правил в соответствии с особенностями решаемых системой задач.
4. Возможность легкого наращивания базы знаний путем добавления новых правил.

Недостатками этого метода представления являются следующие.

1. Необозримость большой базы знаний и ее структуры.
2. Возможность легкого внесения серьезных искажений в базу знаний, приводящих к неправильному функционированию системы (если в системе нет развитых средств проверки целостности базы знаний).
3. Ориентация на последовательную обработку правил.

Контрольные вопросы

1. *Что такое "ядро правила-продукции"?*
2. *Варианты интерпретации ядра правила.*
3. *Основные методы логического вывода в продукционных системах.*
4. *Какие бывают стратегии выбора очередного правила для интерпретации?*
5. *Достоинства и недостатки правил-продукций как метода представления знаний.*
6. *Как может представляться нечеткость знаний или данных в правилах-продукциях?*
7. *Как могут представляться факты, получаемые и используемые в процессе логического вывода в продукционных системах?*
8. *Чем отличаются прямой и обратный логические выводы?*
9. *Когда лучше использовать обратный, а когда прямой логические выводы?*
10. *Какие бывают стратегии выбора правил при логическом выводе?*

3.4. Семантические сети

Основные понятия. Типы отношений в семантических сетях. Принципы обработки (поиска) информации в семантических сетях. Связь семантических сетей с логикой 1-го порядка и псевдофизическими логиками.

Литература – 2, 5, 6, 11, 13, 15, 18, 24.

Семантической сетью называется ориентированный граф с помеченными вершинами и дугами, где вершинам соответствуют конкретные объекты, дугам – отношения между ними.

В семантических сетях используются три основных типа объектов: понятия, события и свойства.

Понятия представляют собой сведения об абстрактных или конкретных (физических) объектах предметной области.

События – это действия, которые могут внести изменения в предметную область, т.е. изменить состояние предметной области.

Свойства используются для уточнения понятий и событий. Применительно к понятиям свойства описывают их особенности или характеристики, например – цвет, размер, качество. Применительно к событиям свойства – продолжительность, место, время и т.д.

Семантические отношения условно делятся на четыре класса: лингвистические, логические, теоретико-множественные и квантифицированные. К наиболее распространенным лингвистическим отношениям относятся падежные и атрибутивные отношения. Падежными (или ролевыми) отношениями могут являться следующие:

– агент, отношение между событием и тем, что (или кто) его вызывает, например, отношение между "завинчиванием" (гайки) и рукой;

– объект, отношение между событием и тем, над чем производится действие, например, между "завинчиванием" и "гайкой";

– условие, отношение, указывающее логическую зависимость между событиями, например, отношение между "завинчиванием" (гайки) и "сборкой" (узла);

– инструмент, отношение между событием и объектом, с помощью которого оно совершается, например, между "завинчиванием" и "верстаком".

Атрибутивные отношения – это отношения между объектом и свойством, например, цвет, размер, форма, модификация и т.д. На рис. 4 приведен пример семантической сети с использованием атрибутивных отношений.

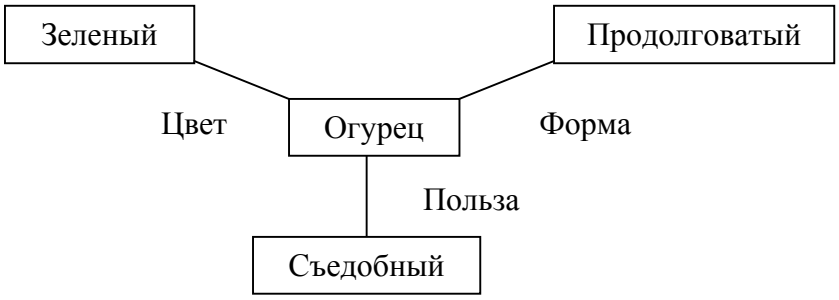

Рис. 4. Пример атрибутивных отношений

Логические отношения – это операции, используемые в исчислении высказываний: дизъюнкция, конъюнкция, импликация, отрицание.

Теоретико-множественные отношения или иерархические – это отношения между элементом множества (подмножества) и множеством, отношение части и целого, отношение между элементом класса и классом и т.п. Этот тип отношений используется для хранения в базе знаний сложных (составных или иерархических) понятий. Этот тип отношений иллюстрируется рис. 5.

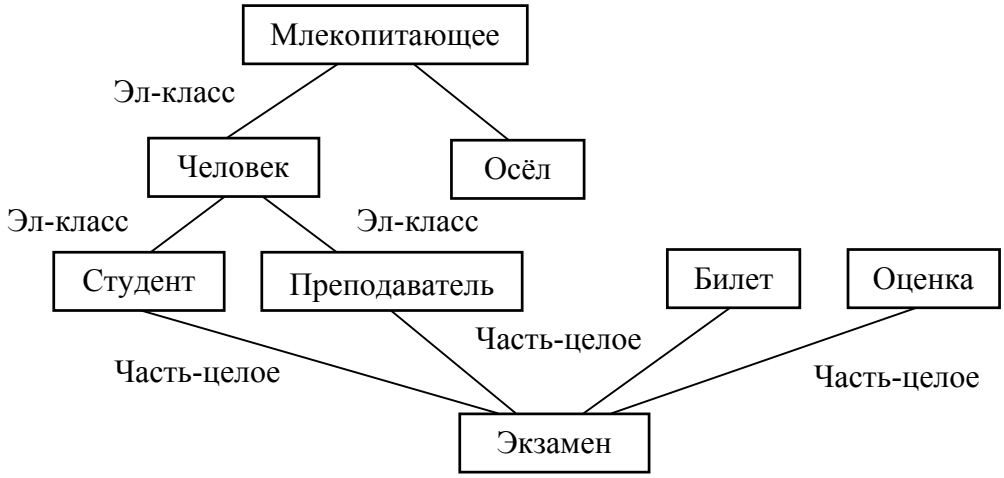

Рис. 5. Пример теоретико-множественных отношений

Квантифицированные отношения – это логические кванторы общности и существования. Они используются для представления знаний типа: "любой

студент должен посещать лабораторные занятия", "существует хотя бы один язык программирования, который должен знать любой выпускник НГТУ".

К базе знаний представленной семантической сетью, возможны следующие основные типы запросов:

1) запрос на существование;

2) запрос на перечисление.

При построении интеллектуальных банков знаний обычно используют разделение интенсиональных и экстенсиональных знаний. Экстенсиональная семантическая сеть (или К-сеть) содержит информацию о фактах, о конкретных объектах, событиях, действиях. Интенсиональная семантическая сеть (или А-сеть) содержит информацию о закономерностях, потенциальных взаимосвязях между объектами, неизменяемую информацию об объектах, т.е. модель мира. Экстенсиональные (конкретные) знания создаются и обновляются в процессе работы с банком данных, а интенсиональные (абстрактные) изменяются редко. Первые можно назвать экземпляром, а последние – моделью (схемой) базы данных.

Запрос к банку знаний, обрабатываемый системой управления базой знаний, представляет собой набор фактов (ситуацию), при описании которого допускается использование переменных вместо значений атрибутов, имен понятий, событий и отношений. Запрос можно представить в виде графа, в котором вершины, соответствующие переменным, не определены.

Поиск ответа сводится к задаче изоморфного вложения графа запроса (или его подграфа) в семантическую сеть.

Запрос на существование не содержит переменных и требует ответа типа ДА, если изоморфное вложение графа запроса в семантическую сеть удалось, и НЕТ – в противоположном случае. При обработке запроса на перечисление происходит поиск всех возможных изоморфных графу запроса подграфов в семантической сети, а также присваивание переменным в запросе значений из найденных подграфов.

Кроме того, на семантических сетях можно использовать методы доказательства, используемые в логике предикатов, т.к. семантическая сеть легко преоьразуется в логику предикатов 1-го порядка (каждое ребро можно представить в виде бинарного предиката).

Достоинством семантических сетей является их универсальность, достигаемая за счет выбора соответствующего применению набора отношений. В принципе с помощью семантической сети можно описать сколь угодно сложную ситуацию, факт или предметную область.

В виде семантической сети можно представить псевдофизическую логику. С другой стороны, семантическую сеть можно записать в виде набора предикатов 1-го порядка. В этом случае именами предикатов будут являться имена отношений (ребер графа), а атрибутами – связываемые отношением узлы семантической сети.

Недостатком семантических сетей является их практическая необозримость при описании модели мира реального уровня сложности. При этом появ-

ляется проблема размещения семантической сети в памяти ЭВМ. Если ее размещать всю в оперативной (виртуальной) памяти, на ее сложность накладываются жесткие ограничения. Если размещать во внешней памяти, появляется проблема, как подгружать необходимые для работы участки.

Частично эта проблема структуризации семантических сетей решается выделением фрагментов семантической сети, называемых обычно высказываниями. Ниже приводится фрагмент семантической сети, состоящей из двух высказываний, связанных отношением импликации (следования). Смысл этого фрагмента семантической сети можно выразить высказыванием (рис. 6) "Если студент учится в НГТУ, значит он – умный".

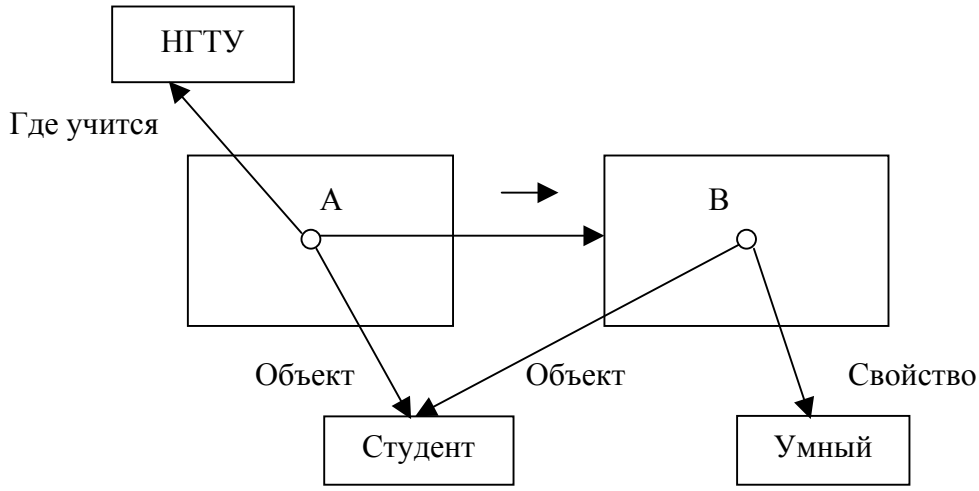

Рис. 6. Пример двух высказываний, связанных импликацией

Кроме того, в семантических сетях нельзя явно задавать наследование свойств, т.к. отношение "элемент класса – класс" является одним из многих типов отношений и его обработка (участие в процедуре поиска подходящих фрагментов) ничем не отличается от обработки других отношений.

Эта проблема структуризации семантических сетей и необходимости задания в них наследования свойств привела к идее структуризации семантических сетей, приведшей к появлению концепции фреймов.

Контрольные вопросы

1. Что такое "семантическая сеть"?

2. Классификация семантических отношений в семантических сетях.

3. Выразить в виде семантической сети высказывание "Все люди смертны и имеют желания".

4. Достоинства и недостатки семантических сетей как метода представления знаний.

5. Как осуществляется поиск информации в семантической сети?

6. Как можно представить семантическую сеть в виде логики предикатов 1-го порядка (или логики Хорна)? Приведите пример.

3.5. Фреймы

Основные понятия: слоты, присоединенные процедуры-слуги и процедуры-демоны, наследование свойств. Связь понятия фрейма и объекта в объетно-ориентированном программировании. Сети фреймов. Принципы обработки данных в сети фреймов. Связь фреймов с объектно-ориентированным программированием.

Литература – 5, 6, 10, 16, 18, 24.

В основе теории фреймов лежит восприятие фактов посредством сопоставления полученной извне информации с конкретными элементами и значениями, а также, с рамками, определенными для каждого концептуального объекта в памяти. Структура, представляющая эти рамки, называется фреймом.

Другими словами, фрейм – это структура, описывающая фрагмент базы знаний, который в какой-то степени рассматривается и обрабатывается обособленно от других фрагментов. Другие фрагменты, с которыми он связан, во фрейме представлены только их именами (идентификаторами) так же как и он в них.

В виде фрейма может описываться некоторый объект, ситуация, абстрактной понятие, формула, закон, правило, визуальная сцена и т.п.

Понятие фрейма неразрывно связано с абстрагированием и построением иерархии понятий.

Из понятия "фрейм", появившегося в конце 60-х годов в работах М. Минского, выросло в дальнейшем понятие объекта и объектно-ориентированного программирования с его идеями инкапсуляции данных в объекте, наследования свойств и методами, привязанными к описанию объектов.

Так как фрейм является более общей и гибкой концепцией, чем "объект", в дальнейшем будем использовать терминологию, сложившуюся при описании фреймов, подразумевая, что с некоторыми ограничениями они могут быть перенесены в среду объектно-ориентированного программирования.

Фреймы подразделяются на два типа: фреймы-прототипы (или классы) и фреймы-примеры (или экземпляры). Фреймы-прототипы используются для порождения фреймов-примеров.

В общем виде фрейм можно описать как структуру, состоящую из имени фрейма, множества слотов, характеризующихся именами и значениями, и множества присоединенных процедур, связанных с фреймом или со слотами:

$$F = (NF, P, (NS_1, VS_1, P_1), ..., (NS_i, VS_i, P_i), ..., (NS_n, VS_n, P_n)),$$

где NF – имя фрейма, NS – имя слота, VS – значение слота, P – присоединенная процедура.

Во фреймах различают два типа присоединенных процедур: процедуры-демоны и процедуры-слуги. Первые из них запускаются автоматически при наличии некоторых условий или событий. Процедуры-слуги запускаются явно.

В реальных системах, базирующихся на фреймах, структура, описанная выше, может быть более сложной за счет иерархической структуры значений слотов (наличия более глубоких уровней в описании фрейма).

Ниже приводится пример фрейма-прототипа, описывающего земельный участок в виде многоугольника:

Имя фрейма:Земельный участок
Количество сторон: (4)
Длины сторон:
Размеры углов:
Площадь: IF_NEEDED: Вычисление площади
IF_ADDED: Вычисление цены
Цена:

Здесь слот "Количество сторон" имеет значение "по умолчанию", равное 4, т.к. подавляющее большинство земельных участков имеет форму четырехугольника.

К слоту "Площадь" присоединены процедуры-демоны вычисляющие площадь и цену, соответственно, запускаемые при запросе слота (событие "IF_NEEDED") и добавлении значения слота "Площадь" (событие "IF_ADDED").

В инженерии знаний существуют языки представления знаний, базирующиеся на концепции фреймов. Наиболее известные из них: KRL и FRL. Оба эти языка являются расширением языка LISP – функционального языка программирования.

Контрольные вопросы

1. Что такое фрейм?

2. В чем сходство и отличие использования фреймов от объектно-ориентированного программирования?

3. В чем отличие фрейма-прототипа от фрейма–примера, процедур-слуг от процедур-демонов?

4. Приведите пример наследования свойств при использовании фреймов.

5. В чем преимущества использования сети фреймов вместо семантической сети?

6. Как решаются задачи при использовании сети фреймов?

4. ТЕХНОЛОГИЯ ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ

Условия применимости экспертных систем. Типы экспертных систем в зависимости от степени завершенности и особенностей использования: демонстрационные, исследовательские, промышленные, коммерческие. Этапы построения экспертных систем: идентификация, концептуализация, формализация, реализация, тестирование. Трудности при создании экспертных систем. Инструментальные средства для создания ЭС: CLIPS, KEE, ESWin. Проблемы извлечения и формализации знаний.

Литература – 1, 2, 4, 5, 6, 10, 11, 15, 16, 20, 22, 23, 24, 26, 28.

4.1. Когда целесообразно использование экспертных систем?

Экспертные системы целесообразно использовать тогда, когда: 1) разработка ЭС возможна; 2) оправдана; 3) методы инженерии знаний соответствуют решаемой задаче.

Рассмотрим более подробно эти условия.

Разработка ЭС возможна когда:

- существуют эксперты в данной области;
- эксперты должны сходиться в оценке предлагаемого решения;
- эксперты должны уметь выразить на естественном языке и объяснить используемые методы;
- задача требует только рассуждений, а не действий;
- задача не должна быть слишком трудной, ее решение должно занимать у эксперта до нескольких часов или дней, а не недель или месяцев;
- задача должна относиться к достаточно структурированной области;
- решение не должно использовать в значительной мере здравый смысл (т.е. широкий спектр общих сведений о мире и о способе его функционирования).

Разработка ЭС оправдана, если:

- решение задачи принесет значительный эффект;
- использовать человека-эксперта невозможно из-за ограниченного количества экспертов или из-за необходимости выполнения экспертизы одновременно во многих местах;
- при передаче информации эксперту происходит значительная потеря времени или информации;
- необходимо решать задачу в окружении, враждебном человеку.

Методы инженерии знаний соответствуют задаче, если задача обладает следующими характеристиками:

- может быть естественным образом решена посредством манипуляции с символами, а не с числами;
- имеет эвристическую природу, т.е. не годится задача, которая может быть решена гарантированно с помощью некоторых формальных процедур;
- должна быть достаточно сложной, чтобы оправдать затраты, но не чрезмерно сложной;
- должна быть достаточно узкой, но практически значимой.

4.2. Этапы создания экспертных систем

В проектировании экспертных систем можно выделить следующие этапы.

1. Идентификация.

1.1. Определение участников и их ролей в процессе создания и эксплуатации экспертной системы.

В процессе создания экспертной системы могут участвовать следующие специалисты: инженеры по знаниям, эксперты, программисты, руководитель проекта, заказчики (конечные пользователи). При реализации сравнительно

простых экспертных систем программистов может не быть. Роль инженера по знаниям – выуживание профессиональных знаний из экспертов и проектирование базы знаний экспертной системы и ее архитектуры. Программист необходим при разработке специализированного для данной экспертной системы программного обеспечения, когда подходящего стандартного (например, оболочки для создания экспертных систем) не существует или его возможностей не достаточно и требуются дополнительные модули.

В процессе эксплуатации могут принимать участие конечные пользователи, эксперты, администратор.

1.2. Идентификация проблемы.

На этом этапе разработчики должны ответить на ряд вопросов, определяющих особенности решаемых экспертами, а, следовательно, будущей экспертной системой, задач. Эти особенности определят и особенности архитектуры экспертной системы, формируемой на последующих этапах. К этим вопросам относятся следующие:

- какой класс задач должна решать ЭС;
- как эти задачи могут быть охарактеризованы или определены;
- какие можно выделить подзадачи;
- какие исходные данные должны использоваться для решения;
- какие понятия и взаимосвязи между ними используются при решении задачи экспертами;
- какой вид имеет решение и какие концепции используются в нем;
- какие аспекты опыта эксперта существенны для решения задачи;
- какова природа и объем знаний, необходимых для решения задачи;
- какие препятствия встречаются при решении задач;
- как эти помехи могут влиять на решение задачи.

1.3. Определение необходимых ресурсов – временных, людских, материальных.

1.4. Определение целей.

В качестве целей, преследуемых при создании экспертных систем, могут быть: повышение скорости принятия решения, повышение качества решений, тиражирование опыта экспертов и т.п.

2. Концептуализация.

На этом этапе разработчики должны ответить на следующие вопросы:

- какие типы данных нужно использовать;
- что из данных задано, а что должно быть выведено;
- имеют ли подзадачи наименования;
- имеют ли стратегии наименования;
- имеются ли ясные частичные гипотезы, которые широко используются.

3. Формализация.

На этом этапе разработчики имеют дело с описанием (формализацией) пространства гипотез, моделей процессов и характеристик данных.

Чтобы понять структуру пространства гипотез необходимо формализовать концепции и определить как они связываются между собой, образуя гипотезы. При этом надо ответить, например, на следующие вопросы:

– выгодно ли описывать концепции как структурированные объекты или рассматривать их как простые понятия;

– являются ли причинно-следственные связи между концепциями важными и следует ли представлять их в явном виде;

– конечно или бесконечно пространство гипотез;

– состоит ли из заранее определенных классов или должно генерироваться из концепций по некоторой процедуре;

– полезно или нет рассматривать гипотезы в иерархическом виде;

– присутствует ли неопределенность в каком-либо виде в конечных или промежуточных гипотезах;

– следует ли использовать разные уровни абстракции.

Модели процессов могут быть поведенческими или математическими (аналитической или статистической).

При понимании природы данных необходимо ответить на следующие вопросы:

– являются ли связи данных с гипотезами причинно-следственными, дефинитивной (т.е. вытекающей из определений) или корреляционной;

– являются ли данные редкими и недостаточными или обильными и избыточными;

– имеется ли неопределенность в данных;

– зависит ли логическая интерпретация данных от порядка их появления во времени;

– какова стоимость приобретения данных;

– как данные приобретаются или извлекаются, какие вопросы надо задать, чтобы получить данные;

– как некоторые характеристики данных можно получить на основе обработки непрерывного потока данных – текста на естественном языке, рисунков, графиков;

– являются ли данные надежными, точными, однозначными или они ненадежны, неточны, неоднозначны (размыты);

– являются ли данные непротиворечивыми и понятными для решения поставленных задач.

Результатом этого этапа является выбор инструментальных средств для реализации экспертной системы и формализация базы знаний в терминах методов представления, поддерживаемых в них.

4. Реализация прототипной версии.

5. Тестирование.

6. Перепроектирование прототипной версии.

4.3. Прототипы и жизненный цикл экспертных систем

По степени готовности к использованию и распространению различают четыре прототипа экспертных систем:

1) демонстрационный; предназначен для демонстрации возможностей будущей экспертной системы, основных архитектурных решений, пользовательского интерфейса, для уточнения требований к пользовательскому интерфейсу и функциям, выполняемым экспертной системой, содержит демонстрационную далеко неполную базу знаний;

2) исследовательский; предназначен для исследования направлений дальнейшего совершенствования экспертной системы и для пополнения базы знаний, может использоваться для решения реальных задач в ограниченных пределах;

3) промышленный; предназначен для использования, как правило, в организации, где был разработан, в нем возможны ограничения, условности, специализация, свойственные для данной организации;

4) коммерческий; предназначен для широкого распространения, обладает гибкостью, удобством в эксплуатации, адаптируемостью к конкретным задачам и требованиям пользователя.

Жизненный цикл экспертной системы состоит из этапов разработки и сопровождения. На этапе разработки создается программное обеспечение и база знаний экспертной системы, на этапе сопровождения происходит исправление выявленных ошибок и пополнение базы знаний без участия разработчиков (если последнее допускается архитектурой экспертной системы).

Применение экспертной системы с базой знаний, неизменяемой в процессе эксплуатации, возможно при достаточно стабильной в течение длительного времени предметной области, в которой решаются задачи. Примерами таких предметных областей являются разделы математического анализа, описание правил диагностики различных заболеваний.

Примерами областей применения, требующих гибкости со стороны создания и пополнения базы знаний, являются: планирование производства, проектирование и диагностика в области электроники, вычислительной техники и машиностроения.

4.4. Инструментальные средства для разработки экспертных систем

Инструментальные средства можно классифицировать по уровню методов представления и обработки знаний следующим образом:

1. Традиционные (в том числе объектно-ориентированные) языки программирования типа С, С++ (как правило, эти инструментальные средства используются не для создания ЭС, а для создания более развитых инструментальных средств).

2. Символьные языки программирования (например, Lisp, Prolog и их разновидности). Эти ИС в последнее время, как правило, не используются в реальных приложениях в связи с тем, что они плохо приспособлены к объединению с программами, написанными на языках традиционного программирования.

3. Инструментарий, содержащий многие, но не все компоненты ЭС. Эти средства предназначены для разработчика, от которого требуются знание программирования и умение интегрировать компоненты в программный комплекс. Примерами являются такие средства, как OPS 5, ИЛИС и др.

4. Оболочки ЭС общего назначения, содержащие все программные компоненты, но не имеющие знаний о конкретных предметных средах. Средства этого и последующего типов не требуют от разработчика приложения знания программирования. Примерами являются ЭКО, Leonardo, Nexpert Object, Kappa, ESWin и др. Надо иметь в виду, что в последнее время термин "оболочка" (shell) используется реже, его заменяют на более широкий термин "среда разработки" (development environment). Если хотят подчеркнуть, что средство используется не только на стадии разработки приложения, но и на стадиях использования и сопровождения, то употребляют термин "полная среда" (complete environment). Примерами таких средств для создания статических ЭС являются: Nexpert Object , ProKappa, ART*Enterprise, Level 5 Object и др.

5. Проблемно/предметно-ориентированные оболочки (среды):

- проблемно-ориентированные средства (problem-specific), ориентированные на некоторый класс решаемых задач и имеющие в своем составе соответствующие этому классу альтернативные функциональные модули (примерами таких классов задач являются задачи поиска, управления, диагностики, планирования, прогнозирования и т.п.);
- предметно-ориентированные средства (domain-specific), включающие знания о некоторых типах предметных областей, что сокращает время разработки БЗ.

6. Пустые экспертные системы, получающиеся исключением базы знаний из разработанной для какого-либо конкретного применения экспертной системы.

В приведенной классификации инструментальные средства перечислены в порядке убывания трудозатрат, необходимых на создание с их помощью конкретной экспертной системы.

Контрольные вопросы

1. При каких условиях целесообразно использовать экспертные системы?

2. Из каких этапов состоит процесс проектирования и построения экспертной системы?

3. Какие существуют типы экспертных систем в зависимости от степени их готовности к использованию?

4. Основные характеристики и особенности архитектуры экспертных систем.

5. Задачи, решаемые инженером по знаниям.

6. Классификация инструментальных средств для построения экспертных систем.

7. Целесообразно ли разрабатывать экспертную систему для определения авторства живописного художественного произведения? Обоснуйте ответ.

8. Целесообразно ли разрабатывать экспертную систему для прогнозирования исхода сдачи экзамена? Обоснуйте ответ.

9. На какие основные вопросы должен ответить разработчик экспертной системы на этапе идентификации предметной области?

10. Какие характеристики данных, используемых экспертной системой надо учитывать при ее разработке?

5. ПРИОБРЕТЕНИЕ ЗНАНИЙ

Основные понятия методов обучения. Классификация методов обучения по способу обучения: эмпирические и аналитические, по глубине обучения – символьные (поверхностные) и на основе знаний (глубинные). Связь этой классификации с понятиями индуктивного вывода, вывода по аналогии, обучения на примерах. Сведение задачи приобретения знаний к задаче обобщения. ДСМ-метод. Определение индуктивного вывода.

Литература – 4, 5, 6, 8, 15, 16, 22, 25.

Основной проблемой при разработке современных экспертных систем является проблема приобретения знаний, т.е. преобразование разного вида информации (данных) из внешнего представления в представление в виде знаний, пригодное для решения задач, для которых создается экспертная система. Эту проблему часто называют проблемой извлечения знаний из данных (в более общем виде, из внешнего мира), которая сводится к задаче обучения интеллектуальной системы.

Примерами задач приобретения знаний являются:

1) выявление причинно-следственных связей между атрибутами реляционной базы данных и формирование их в виде правил в продукционной экспертной системе;

2) формирование программы (или правил) решения задачи (например, планирования производственного процесса или поведение робота) на основе примеров удачного планирования, вводимых в компьютер;

3) выявление информативных признаков для классификации объектов, существенных с точки зрения решаемой задачи.

Обучающиеся системы можно классифицировать по двум признакам: уровень, на котором происходит обучение и применяемый метод обучения. По первому признаку различают обучение на символьном уровне (SLL – symbol level learning), при котором происходит улучшение представления знаний на основе опыта, полученного при решении задач, и обучение на уровне знаний (KLL – knowledge level learning), при котором происходит формирование новых знаний из существующих знаний и данных.

На символьном уровне обучение сводится к манипулированию уже существующими структурами, представляющими знание, например, корректировка коэффициентов достоверности правил-продукций, изменение порядка расположения (просмотра) правил-продукций в базе знаний вводимого пользовате-

лем описания решения задачи на достаточно формализованном языке, не сильно отличающимся от языка, на котором представляются знания в системе.

На уровне знаний обучение сводится к выявлению и формализации новых знаний. Например, из фактов

журавль умеет летать,
воробей умеет летать,
синица умеет летать,
журавль есть птица,
воробей есть птица,
синица есть птица

система может сформулировать правило-продукцию

Если X есть птица
то X умеет летать.

По признаку применяемого метода обучения различают системы, в которых используются аналитические или эмпирические методы обучения. Аналитические в свою очередь делятся на использующие глубинные (knowledge-rich) или поверхностные (knowledge-drizen) знания.

Эмпирические делятся на использующие знания (knowledge-learning) или данные (data-drizen).

С другой стороны в инженерии знаний известны три основных подхода к приобретению знаний: индуктивный вывод, вывод по аналогии и обучение на примерах. В основе индуктивного вывода лежит процесс получения знаний из данных и/или других знаний (в продукционных системах – правил из фактов и/или других правил). Вывод по аналогии основан на задании и обнаружении аналогий между объектами (ситуациями, образами, постановками задачи, фрагментами знаний) и применением известных методов (процедур) к аналогичным объектам. В основе обучения на примерах лежит демонстрация системе и запоминание ей примеров решения задач. Резкой границы между этими методами не существует, т.к. все они базируются на обобщении, реализованной в той или иной форме, т.е. реализуют переход от более конкретного знания (фактов) к более абстрактному знанию.

На рис. 7 показана классификация обучающихся систем и взаимосвязи между понятиями, связанными с приобретением знаний.

Наиболее известными методами приобретения знаний являются ДСМ-метод (обычно относится к индуктивным методам) и нейронные сети (в них реализовано в наиболее явном виде обучение на примерах).

В ДСМ-методе используется представление знаний об экспериментах (наблюдениях), подтверждающих причинно-следственные связи между факторами, в виде матрицы гипотез

Рис. 7. Классификация обучающихся систем

$$M^{+}= \begin{array}{cccc} a\backslash b & b_1\ldots & b_j & b_m \\ a_1 & q_{11} & q_{1j} & q_{1m} \\ \ldots & & & \\ a_i & q_{i1} & q_{ij} & q_{im} \\ \ldots & & & \\ a_n & q_{n1} & q_{nj} & q_{nm} \end{array}$$

где: a_i – факторы-причины,

b_j – факторы-следствия,

q_{ij} – оценки истинности (силы) причинно-следственной связи между соответствующими факторами.

Оценка истинности определяется в процессе обучения (экспериментов или наблюдений) как k^+/k, k-общее количество экспериментов (примеров), а k^+ – количество экспериментов, подтверждающих причинно-следственную связь (положительных примеров).

Таким образом, после обучения мы имеем матрицу со значениями q_{ij} в пределах интервала (0,1). Если значение $q_{ij} \approx 1$, это означает, что между факторами a_i и b_j есть причинно-следственная связь и ее можно записать в виде правила.

Иногда в ДСМ-методе используется и матрица отрицательных примеров M^-.

Для определения индуктивного вывода необходимо определить следующие его составляющие:

1) метод и форма предъявления данных (фактов или обучающих примеров);
2) метод индуктивного вывода (алгоритмы обработки фактов);
3) что и в каком виде мы хотим получить (вывести);
4) условия, при которых можно считать, что вывод завершен.

Контрольные вопросы

1. В чем отличие методов приобретения знаний на символьном уровне и на уровне знаний?

2. Что нужно определить, чтобы определить индуктивный вывод?

3. В чем отличие обучения на одном и на множестве примеров?

4. В чем суть ДСМ-метода приобретения знаний?

6. НЕЙРОННЫЕ СЕТИ

Задачи, решаемые с помощью нейронных сетей. Формальная модель нейрона Мак-Каллока-Питса. Многослойные персептроны. Сведение функционирования нейронной сети к задаче минимизации целевой функции. Алгоритм обучения обратным распространением ошибки. Нейронная сеть как ассоциативная память. Модель Хопфилда. Модель Кохонена. Модель Гросберга-Карпентера (ART-1). Программная и аппаратная реализации нейронных сетей. Использование нейронных сетей для прогнозирования.

Литература – 3, 6, 7, 8, 12, 14, 17, 19, 27.

6.1. Задачи, решаемые нейронными сетями

Ниже перечисляются некоторые проблемы, решаемые с помощью ИНС и представляющие интерес для ученых и инженеров.

Классификация/распознавание образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови, распознавание отпечатков пальцев, а также, лиц.

Кластеризация/категоризация. При решении задачи кластеризации, которая известна также как классификация образов "без учителя" отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

Аппроксимация функций. Предположим, что имеется обучающая выборка (пары данных вход-выход), которая генерируется неизвестной функцией (x), искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции (x). Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

Предсказание/прогноз. Пусть заданы n дискретных отсчетов в последовательные моменты времени t, Задача состоит в предсказании значения $y(t)$ в некоторый будущий момент времени $t > n + 1$.

Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза. Предсказание является также основной задачей, решаемой сомообучаемыми мобильными автономными системами в условиях адаптации к незнакомой окружающей среде.

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, относящаяся к классу *NP*-полных, является классическим примером задачи оптимизации.

Ассоциативная память. Содержимое ассоциативной памяти или памяти, адресуемой по содержанию, может быть вызвано по частичному входу или искаженному содержанию. Ассоциативная память полезна при создании мультимедийных информационных баз данных. А также, она является основой системы управления обучаемых мобильных роботов.

Управление. Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ является входным управляющим воздействием, а $y(t)$ – выходом системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия $u(t)$, при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

Все задачи, решаемые нейронными сетями, можно свести к двум основным:

- распознавание (классификация);
- регрессия.

Задача классификации заключается в формировании нейронной сетью в процессе обучения гиперповерхности в пространстве признаков, разделяющей

признаки на классы. И выходы обученной нейронной сети соответствуют распознанному классу входного вектора (набора признаков).

Задача регрессии заключается в аппроксимации нейронной сетью произвольной нелинейной функции. В этом случае значение функции снимается с выхода нейронной сети, а входами являются аргументы. Существует теорема, доказывающая, что многослойный персептрон может аппроксимировать любую нелинейную функцию от n аргументов с какой угодно заданной точностью.

6.2. Основные сведения об искусственных нейронных сетях

В нейронных сетях знания содержатся в состояниях множества так называемых нейроподобных элементов (или просто нейронов) и связей между ними. Направление, которое во главу угла ставит связи между нейронами, называется коннективизмом.

Формальная модель нейрона Мак-Каллока-Питтса, которая и сейчас является наиболее применяемым формализмом для описания отдельного нейрона в нейронной сети, показана на рис. 8.

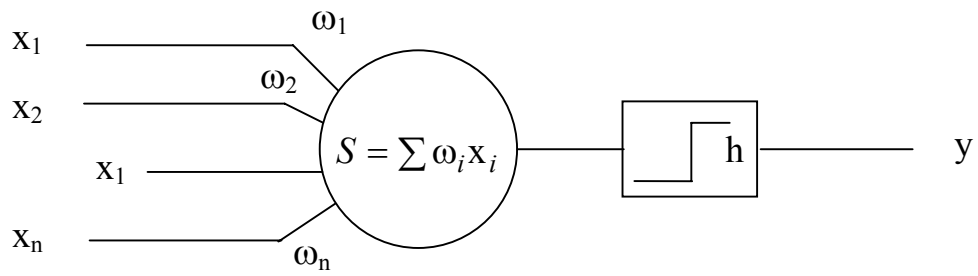

Рис. 8. Формальная модель нейрона Мак-Каллока – Питтса

Здесь: x_i – сигнал на i-м входе (синапсе) нейрона;
ω_i – вес i-го входа (синапса) нейрона;
y – выход нейрона;
h – порог срабатывания нейрона.

В модели взвешенная сумма сигналов на входах нейрона сравнивается с пороговым значением h, и на выходе есть сигнал, если она превышает порог. В современных моделях нейронов пороговая функция в общем случае заменяется на нелинейную функцию $y = f(S)$, называемую передаточной функцией или функцией активации нейрона. В качестве этой функции может использоваться, одна из сигмоидальных функций, например, рациональная сигмоида (рис. 9)

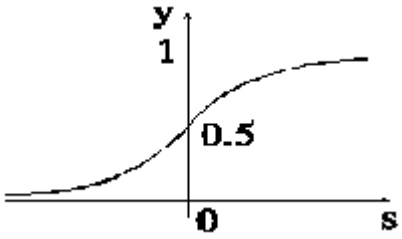

Рис. 9. Сигмоидальная функция

$$f(S) = \frac{S}{S + \alpha}.$$

Параметр α обычно называется смещением. Таким образом, иногда говорят, что нейрон состоит из умножителей (на веса), сумматора и нелинейного элемента.

Сигмоидальная функция имеет то преимущество перед пороговой, что она дифференцируема на всей своей области определения. Это ее свойство используется в алгоритмах обучения (в частности, обучения обратным распространением ошибки).

Из связанных определенным образом нейронов (узлов) строится нейронная сеть с определенным количеством входов и выходов. Обычно различают три типа узлов (нейронов) – входные (входной слой нейронов или Input layer), выходные (выходной слой или Output layer) и скрытые слои нейронов (Hidden layers) (рис. 10).

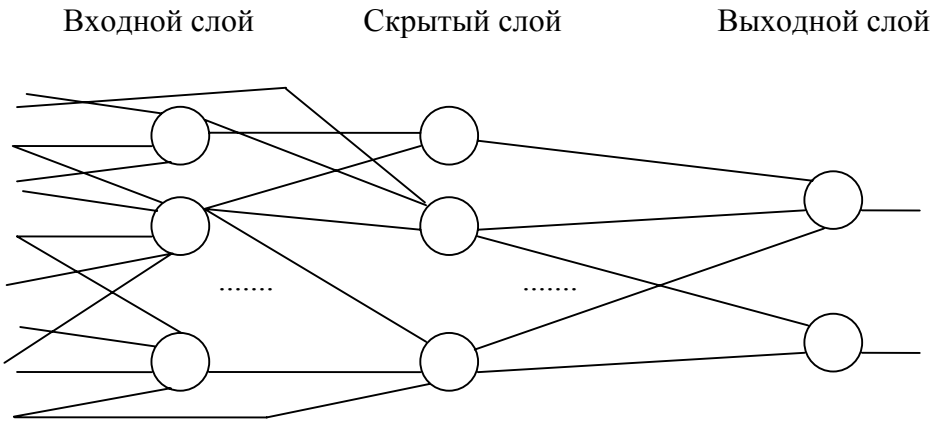

Рис. 10. Нейронная сеть с прямыми связями

Функционирование нейронной сети состоит из двух этапов: обучения сети "правильному" или адекватному реагированию на входную информацию (входной вектор) и использования обученной сети для распознавания входных векторов. Последний этап часто называют тестированием. Другими словами, сеть учится распознаванию входных векторов, т.е. формированию выходных векторов, соответствующих распознанному классу входных векторов. При этом знания о соответствии входных векторов выходным сохраняются в весах синапсов и порогах нейронов. В случае коннективистского подхода (коннекцианизма) – только в весах синапсов. Иногда под входным вектором понимается конкатенация входного и выходного вектора и не все разряды этого вектора могут задаваться при обучении и тестировании сети. В некоторых моделях нейронных сетей (например, модели Хопфилда) входные и выходные сигналы не различаются и соответствующие им входы (выходы) сети могут меняться ролями в процессе функционирования сети.

Функционирование нейронной сети часто описывают в терминах задач оптимизации. Ее обучение можно представить как формирование n-мерной гиперповерхности (где n – размерность входного вектора), определенной целевой функции, обычно называемой энергией сети. Описание этой гиперповерхности хранится в карте весов синапсов и в порогах нейронов. Например, в случае модели Хопфилда энергетическая функция описывается формулой

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\, x_i\, x_j ,$$

где: N – количество нейронов (размерность входного вектора);
w_{ij} – вес связи между i-м и j-м нейронами;
x_i – состояние i-го нейрона;

При тестировании (использовании) обученной нейронной сети происходит процесс поиска ближайшего минимума целевой функции. При этом происходит восстановление искаженных разрядов входного вектора или "вспоминание" неизвестных разрядов, ассоциативно связанных с заданными (известными) разрядами.

6.3. Классификация нейронных сетей

В настоящее время существует большое разнообразие моделей нейронных сетей. Их различают по структуре сети (связей между нейронами), особенностям модели нейрона, особенностям обучения сети.

По структуре нейронные сети можно разделить на неполносвязные (или слоистые) и полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями.

Неполносвязные нейронные сети описываются неполносвязным ориентированным графом. Наиболее распространенным типом таких нейронных сетей являются персептроны: однослойные (простейшие персептроны) и многослойные, с прямыми, перекрестными и обратными связями. В нейронных сетях с прямыми связями нейроны j-ого слоя по входам могут соединяться только с нейронами i-ых слоев, где $j > i$, т.е. с нейронами нижележащих слоев. В нейронных сетях с перекрестными связями допускаются связи внутри одного слоя, т.е. выше приведенное неравенство заменяется на $j \geq i$. В нейронных сетях с обратными связями используются и связи j-ого слоя по входам с i-ым при $j < i$. Кроме того, по виду связей различают персептроны с регулярными и случайными связями. Нейронные сети с обратными связями называют рекуррентными.

По используемым на входах и выходах сигналам нейронные сети можно разделить на аналоговые и бинарные.

По моделированию времени нейронные сети подразделяются на сети с непрерывным и дискретным временем. Для программной реализации применяется, как правило, дискретное время.

По организации обучения разделяют обучение нейронных сетей с учителем (supervised neural networks) и без учителя (nonsupervised). При обучении с учителем предполагается, что есть внешняя среда, которая предоставляет обучающие примеры (значения входов и соответствующие им значения выходов) на этапе обучения или оценивает правильность функционирования нейронной сети и в соответствии со своими критериями меняет состояние нейронной сети или поощряет (наказывает) нейронную сеть, запуская тем самым механизм изменения ее состояния. Под состоянием нейронной сети, которое может изменяться, обычно понимается:

– веса синапсов нейронов (карта весов – map) (коннекционистский подход);

– веса синапсов и пороги нейронов (обычно в этом случае порог является более легко изменяемым параметром, чем веса синапсов);

– установление новых связей между нейронами (свойство биологических нейронов устанавливать новые связи и ликвидировать старые называется пластичностью).

Кроме того, есть так называемые "растущие" нейронные сети, в которых количество нейронов изменяется в процессе обучения. Алгоритмы обучения таких сетей называются конструктивными.

По способу обучения разделяют обучение по входам и по выходам. При обучении по входам обучающий пример представляет собой только вектор входных сигналов, а при обучении по выходам в него входит и вектор выходных сигналов, соответствующий входному вектору.

По способу предъявления примеров различают предъявление одиночных примеров и "страницы" примеров. В первом случае изменение состояния нейронной сети (обучение) происходит после предъявления каждого примера. Во втором – после предъявления "страницы" (множества) примеров на основе анализа сразу их всех.

6.4. Персептроны и алгоритм обратного распространения ошибки

Обычно в настоящее время под персептроном понимают нейронную сеть, стоящую из множества слоев (рис. 10), объединенных прямыми связями, т.е. каждый нейрон *i*-го слоя связан по входам только с нейронами (*i* – 1)-го слоя (со всеми). Такие сети называют еще нейронными сетями прямого распространения.

Функционирование многослойной сети прямого распространения выполняется в соответствии с формулами:

$$S_{i_m} = \sum_{i_{m-1}-1}^{N_{m-1}} W_{i_m i_{m-1}} y_{i_{m-1}} - b_{i_m}, i_m = 1,2,...,N_m, m = 1,2,...,L$$

$$y_{t_m} = f(S_{i_m}), i_m = 1,2,...,N_m, m = 1,2,...,L$$

где: m – номер слоя,

y_i – выход i-го нейрона,

W_{ij} – вес связи между входом i-го и выходом j-го нейронов,

b_i – смещение i-го нейрона,

i_m – i-ый нейрон m-го слоя.

Обучение сети методом обратного распространения ошибки разбивается на следующие этапы.

1.Инициализация сети.

Весовым коэффициентам и смещениям сети присваиваются малые случайные значения из диапазонов и соответственно.

2. Определение элемента обучающей выборки - обучающего примера (<текущий вход>, <желаемый выход>).

Текущие входы (x_0, x_1... x_{N-1}), должны различаться для всех элементов обучающей выборки. При использовании многослойного персептрона в качестве классификатора желаемый выходной сигнал (d_0, d_1... d_{N-1}) состоит из нулей за исключением одного единичного элемента, соответствующего классу, к которому принадлежит текущий входной сигнал.

3. Вычисление текущего выходного сигнала:

Текущий выходной сигнал определяется в соответствии с традиционной схемой функционирования многослойной нейронной сети.

4. Настройка синаптических весов:

Для настройки весовых коэффициентов используется рекурсивный алгоритм, который сначала применяется к выходным нейронам сети, а затем проходит сеть в обратном направлении до первого слоя. Синаптические веса настраиваются в соответствии с формулой:

$$w_{i,j}(t+1) = w_{i,j}(t) + rg_j x_i,$$

где w_{ij} – вес от нейрона i или от элемента входного сигнала i к нейрону j в момент времени t;

x_i' – выход нейрона i или i-ый элемент входного сигнала;

r – шаг обучения;

g_j – значение ошибки для нейрона j.

Если нейрон с номером j принадлежит последнему слою, то

$$g_j = y_j(1-y_j)(d_j - y_j),$$

где d_j – желаемый выход нейрона j;

y_j – текущий выход нейрона j.

Если нейрон с номером j принадлежит одному из слоев с первого по предпоследний, то

$$g_j = x_j'(1-x_j')\sum_k g_k w_{jk},$$

где k пробегает все нейроны слоя с номером на единицу больше, чем у того, которому принадлежит нейрон j.

5. Если ошибка функционирования сети при обработке примера

$$E = \frac{1}{2}\sum_j (d_j - y_j)^2,$$

больше заданной величины ε, то перейти к шагу 4.

6. Если не обработаны все обучающие примеры, перейти к шагу 2.

7. Завершение обучения.

Классический метод обратного распространения относится к алгоритмам с линейной сходимостью. Для увеличения скорости сходимости необходимо использовать матрицы вторых производных функции ошибки.

Существуют многочисленные модификации алгоритма обратного распространения, которые связаны с использованием различных функций ошибки, различных процедур определения направления и величины шага.

Обратное распространение один из самых популярных алгоритмов обучения, с его помощью были решены и решаются многочисленные практические задачи. Персептроны с обучением обратным распространением ошибки используются для решения задач классификации/распознавания и прогнозирования. Главным их недостатком является медленное обучение.

6.5. Модель Хопфилда

Модель разработана Хопфилдом в 1982 году. С тех пор были предложены многочисленные ее модификации. Данная модель используется как ассоциативная память, классификатор и для решения некоторых задач оптимизации. Эта модель относится к полносвязным рекуррентным сетям.

В дальнейшем будем рассматривать бинарную модель Хопфилда.

Исходными данными для расчета значений синаптических весов сети являются векторы – образцы классов. Сеть функционирует циклически. Выход каждого из нейронов подается на входы всех остальных нейронов. Нейроны сети имеют жесткие пороговые функции (рис. 11).

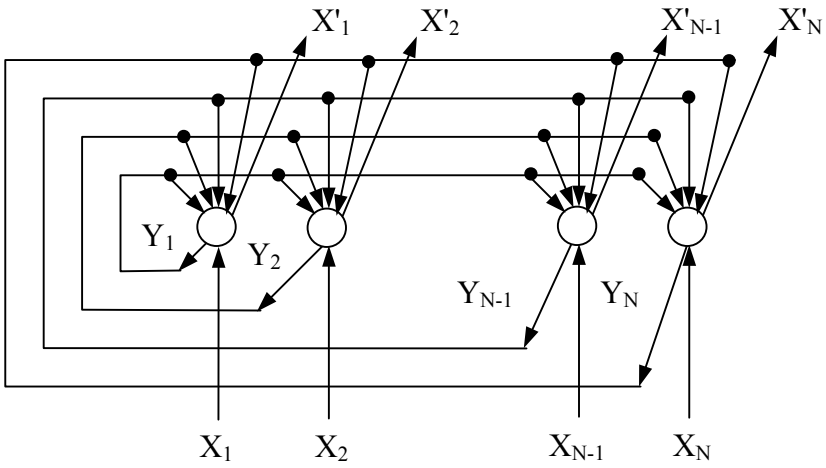

Рис. 11. Сеть Хопфилда

Ниже используются следующие условные обозначения:

w_{ij} – i-й синаптический вес j-го нейрона;

x_i – i-й элемент входного сигнала сети;

x_{ij} – i-й элемент j-го вектора-образца;

x_i' – i-й элемент выходного сигнала сети,;

y_j – выход j-го нейрона;

N – количество элементов (размерность) входного сигнала, количество нейронов в сети;

M – количество векторов-образцов.

Состояние сети характеризуется симметричной матрицей весов синапсов

$$w_{ij} = w_{ji} \forall i, j = \overline{1, N}.$$

Входные и выходные сигналы биполярные и принимают значения –1 и +1. Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации – технологическими возможностями. Размерности входных и выходных сигналов совпадают.

Сеть, содержащая N нейронов может запомнить не более $M = 0.15*N$ образов. При этом запоминаемые образы не должны быть сильно коррелированы. Мера коррелированности:

$$K = \sum_{j,k i=1}^{M} \sum^{N} x_{ij} x_{ik},$$

$$j \neq k.$$

Тип передаточной функции: жесткая пороговая.

Число синапсов в сети: $M*(M-1)$.

Формирование синаптических весов (обучение) сети осуществляется по формуле:

$$w_{ij} = \sum_{s=1}^{M} x_{is} x_{js}, \quad i \neq j.$$

Другими словами, веса после предъявления каждого примера вычисляются по формуле (правило Хебба):

$$w_{ij} = w_{ij} + x_i x,$$
$$w_{ij} = 0, \quad i = j,$$
$$1 \leq i, \quad j \leq N.$$

Функционирование сети описывается следующими формулами:

$$y_j(0) = x_j, 1 \leq j \leq N,$$

$$y_j(t+1) = f\left(\sum_{i=1}^{N} w_{ij} y_i(t) \right), \quad 1 \leq j \leq N.$$

Функционирование заканчивается, если на некотором шаге T для всех j:

$$y_j(T) = y_j(T-1).$$

Нелинейная функция f обычно выглядит как пороговая, принимающая значения 1 при аргументе > 0 и – 1 – в противном случае. Если используются значения 0 и 1 для кодирования состояний нейронов, то сравнение производится не с нулем, а с порогом – константой, одинаковой для всех нейронов.

Выходной сигнал сети:

$$x_j' = y_j(T).$$

В процессе функционирования может использоваться процедура "замора-

живания" состояний некоторых нейронов, которые рассматриваются как входные (т.е. на которые поступает входной сигнал). "Замороженные" нейроны не меняют своего состояния. Такое функционирование может рассматриваться как восстановление вектора по его фрагменту, а обученная нейронная сеть Хопфилда – как ассоциативная память.

В процессе функционирования уменьшается энергетическая функция:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} x_i x_j .$$

Другими словами, состояние нейронной сети "скатывается" в минимум гиперповерхности, сформированной в процессе обучения.

Существует много разновидностей модели Хопфилда, например, с отжигом, т.е. наряду с детерминированным алгоритмом функционирования, описанным выше, работает так называемый "алгоритм отжига", обеспечивающий возможность сети выбраться из локального экстремума энергетической функции и перейти в глобальный экстремум с использованием случайного изменения весов или состояний нейронов.

6.6. Достоинства и недостатки нейронных сетей как средства для обработки знаний

Если рассматривать нейронную сеть как способ представления знаний, то в ней хранятся знания об ассоциативных связях между стимулами (входными векторами) и откликами (выходными векторами). Знания хранятся (формируются в процессе обучения) обычно в форме весов связей между нейронами.

Недостатками нейронных сетей в качестве метода представления знаний являются:

- трудности вербализации результатов работы нейронной сети и объяснений, почему она приняла то или иное решение;
- невозможность гарантировать повторяемость и однозначность получения результатов.

Преимущества нейронных сетей для представления и обработки знаний:

- отсутствие необходимости формализации знаний, формализация заменяется обучением на примерах;
- естественное представление и обработка нечетких знаний примерно так, как это осуществляется в естественной интеллектуальной системе – мозге;
- ориентация на параллельную обработку, что при соответствующей аппаратной поддержке обеспечивает возможность работы в реальном времени;
- отказоустойчивость и живучесть при аппаратной реализации нейронной сети;
- возможность обработки многомерных (размерности больше трех) данных и знаний без увеличения трудоемкости (но в этом случае затруднено объяснение результатов, т.к. человек с трудом воспринимает многомерность).

Контрольные вопросы

1) В чем преимущества и недостатки использования нейронных сетей в интеллектуальных системах?

2) Модель нейрона Мак-Каллока-Питтса.

3) Классификация моделей нейронных сетей.

4) Почему использовать сигмоидальную активационную функцию во многих случаях лучше, чем пороговую?

5) В чем суть алгоритма обучения персептрона обратным распространением ошибки?

6) Что такое энергетическая функция нейронной сети?

7) Как функционирование нейронной сети связано с решением задачи оптимизации?

8) Приведите примеры моделей нейронных сетей, в которых используется обучение с учителем/без учителя?

9) В чем заключаются проблемы обучения нейронных сетей?

7. ИСПОЛЬЗОВАНИЕ ЕСТЕСТВЕННОГО ЯЗЫКА В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Трудности моделирования понимания естественного языка в искусственных системах. Уровни понимания. Синтаксически- и семантически-ориентированные подходы к анализу естественного языка. Методы анализа естественного языка.

Литература – 1, 6, 10, 16, 21.

7.1.Основные понятия о системах, использующих естественный язык

Системы, в которых используется естественный язык (ЕЯ) можно разделить на классы как показано на рис. 12.

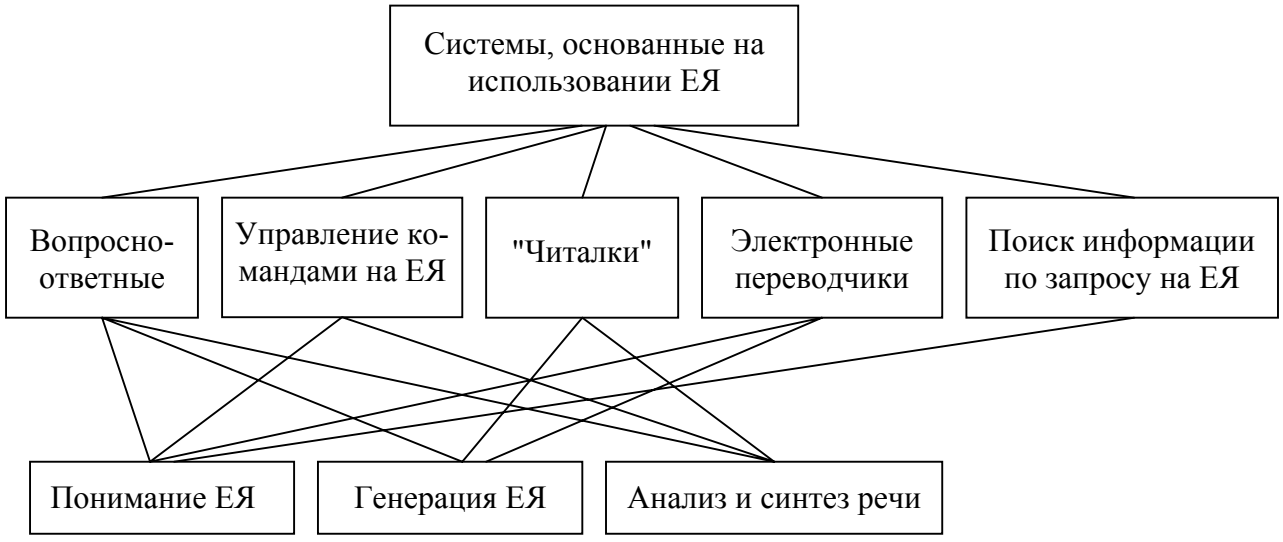

Рис. 12. Основные классы прикладных систем, основанных на ЕЯ, и решаемые при их создании задачи

Говоря о возможных решениях задачи анализа естественного языка (моделирования понимания ЕЯ), можно выделить следующие два основных подхода.

1. Психолингвистический подход – состоит в моделировании психологических механизмов человека, обеспечивающих полноценное понимание естественно-языковых текстов.

2. Утилитарный подход – состоит в создании технических средств, обеспечивающих взаимодействие на естественном языке с компьютерными системами различного назначения, решающими какие-то свои задачи, сами по себе, чаще всего нелингвистические.

Первый из них связан с большими трудностями, т.к. естественный язык и диалог на нем отражает в себе все многообразие и сложность мышления в естественном мозге. Это – использование контекста, умолчаний, анафорических ссылок, метафор, здравого смысла, внелингвистических методов передачи информации в процессе диалога (жесты, мимика). Таким образом, его реализацию можно приравнять к реализации искусственного разума в полном объеме.

Второй подход связан с разработкой систем, понимающих ограниченный или похожий на естественный язык (деловой прозы).

Существуют два основных подхода к реализации систем, моделирующих понимание естественного языка (ЕЯ) – синтаксически- и семантически-ориентированный. В синтаксически-ориентированном подходе строго выдерживается следующая последовательность этапов анализа:

1) морфологический анализ – анализ структуры слов, т.е. распознавание корня и аффиксов (приставок, суффиксов, окончаний), с использованием словарей корней и аффиксов;

2) синтаксический анализ – анализ структуры предложения, т.е. частей предложения (или ролей слов в нем) с использованием грамматики языка;

3) семантический анализ – анализ смысла предложения, т.е. интерпретация его в терминах представления смысла, с использованием базы знаний о предметной области и знаний о синтаксисе представления смысла;

4) прагматический анализ – анализ целей предложения или ожиданий и желаний его источника с целью планирования реакции на анализируемое предложение.

Недостатками синтаксически-ориентированного подхода являются:

- расточительность в использовании ресурсов – времени и памяти за счет необходимости использования огромной грамматики ЕЯ и емких словарей;
- сложность и трудоемкость обнаружения и исправления на последующих этапах ошибок анализа, допущенных на предыдущих этапах с учетом того, что морфологические и синтаксические ошибки в предложениях ЕЯ вполне естественны.

В отличие от этого подхода в семантически-ориентированном анализе

главным и первым этапом анализа является анализ семантики (смысла), иногда, предварительный, т.к. далее смысл может уточняться с использованием уже синтаксического и морфологического анализа. В этом случае можно говорить не об анализе, а о распознавании смысла предложения.

7.2. Технологии анализа естественного языка

Под технологией анализа ЕЯ подразумевается перевод некоторого выражения на ЕЯ во внутреннее представление. Фактически все системы анализа ЕЯ могут быть распределены на следующие категории: подбор шаблона (Pattern Matching), синтаксический анализ, семантические грамматики, анализ с помощью падежных фреймов, "жди и смотри" (Wait And See), словарный экспертный (Word Expert), коннекционистский, "скользящий" (Skimming) анализ. Ниже пойдет речь о некоторых наиболее распространенных методах анализа ЕЯ.

Подбор шаблона. Сущность данного подхода состоит в интерпретации ввода в целом, а не в интерпретации смысла и структуры его отдельных составляющих на более низком уровне. При использовании этого метода происходит сравнение уже имеющихся в системе шаблонов-образцов с текстом, поступившим на вход. Обычно шаблоны представлены в виде простого списка соответствий между классами высказываний и интерпретациями. Иногда они дополнены семантическими элементами или другими компонентами более высокого уровня. Построенная таким образом, система ничего не понимает, а лишь поддерживает диалог, сравнивая реплики пациента с шаблонами и присвоенными им соответствующими ответными репликами, такими, как:

X всегда Y:	Вы можете привести какой-нибудь пример?
	Когда?
	Действительно, всегда?
X Вы Y меня:	Почему Вы думаете, что я Y Вас?
	Вам нравится думать, что я Y Вас, не правда ли?
	Представьте, что я Y Вас. Что Вы об этом думаете?

В результате система способна вести такой диалог:

Человек	Люди **всегда** пристально смотрят на меня.	подбор по 1-му шаблону с учетом *всегда*
Система	Вы можете привести какой-нибудь пример?	первый ответ по 1-му шаблону
Человек	Например, сейчас ты **смотришь** на меня.	подбор по 2-му шаблону, Y=*смотреть*
Система	Почему Вы думаете, что я смотрю на Вас?	первый ответ по 2-му шаблону
Человек	Потому что ты **ненавидишь** меня.	подбор по 2-му шаблону, Y=*ненавидеть*
Система	Вам нравится думать, что я ненавижу Вас, не правда ли? и т.д.	второй ответ по 2-му шаблону

Синтаксический анализ. При использовании синтаксического анализа происходит интерпретация отдельных частей высказывания, а не всего высказывания в целом. Обычно сначала производится полный синтаксический анализ, а затем строится внутренне представление введенного текста, либо производится интерпретация.

Деревья анализа и свободно-контекстные грамматики. Большинство способов синтаксического анализа реализовано в виде деревьев. Одна из простейших разновидностей – свободно-контекстная грамматика, состоящая из правил типа *S = NP + VP* или *VP = V + NP* и полагающая, что левая часть правила может быть заменена на правую без учета контекста. Свободно-контекстная грамматика широко используется в машинных языках, и с ее помощью созданы высокоэффективные методы анализа. Недостаток этого метода – отсутствие запрета на грамматически неправильные фразы, где, например, подлежащее не согласовано со сказуемым в числе. Для решения этой проблемы необходимо наличие двух отдельных, параллельно работающих грамматик: одной – для единственного, другой – для множественного числа. Кроме того, необходима своя грамматика для пассивных предложений и т.д. Семантически неправильное предложение может породить огромное количество вариантов разбора, из которых один будет превращен в семантическую запись. Всё это делает количество правил огромным и, в свою очередь, свободно-контекстные грамматики непригодными для *NLP*.

Трансформационная грамматика. Трансформационная грамматика была создана с учетом упомянутых выше недостатков и более рационального использования правил ЕЯ, но оказалась непригодной для *NLP*. Трансформационная грамматика создавалась Хомским как порождающая, что, следовательно, делало очень затруднительным обратное действие, т.е. анализ.

Расширенная сеть переходов. Расширенная сеть переходов была разработана Бобровым (Bobrow), Фрейзером (Fraser) и во многом Вудсом (Woods) как продолжение идей синтаксического анализа и свободно-контекстных грамматик в частности. Она представляет собой (рис. 13) узлы и направленные стрелки, "расширенные" (т.е. дополненные) рядом тестов (правил), на основании которых выбирается путь для дальнейшего анализа. Промежуточные результаты записываются в ячейки (регистры). Ниже приводится пример такой сети, позволяющей анализировать простые предложения всех типов (включая пассив), состоящие из подлежащего, сказуемого и прямого дополнения, таких, как *The rabbit nibbles the carrot (Кролик грызет морковь)*. Обозначения у стрелок означают номер теста, а также либо признаки, аналогичные применяемым в свободно-контекстных грамматиках (*NP*), либо конкретные слова (by). Тесты написаны на языке LISP и представляют собой правила типа *если условие=истина, то присвоить анализируемому слову признак X и записать его в соответствующую ячейку.*

Разберем алгоритм работы сети на вышеприведенном примере. Анализ начинается слева, т.е. с первого слова в предложении. Словосочетание *the rabbit*

проходит тест, который выясняет, что оно не является вспомогательным глаголом (*Aux*, стрелка 1), но является именной группой (*NP*, стрелка 2). Поэтому *the rabbit* кладется в ячейку *Subj,* и предложение получает признак *TypeDeclarative*,

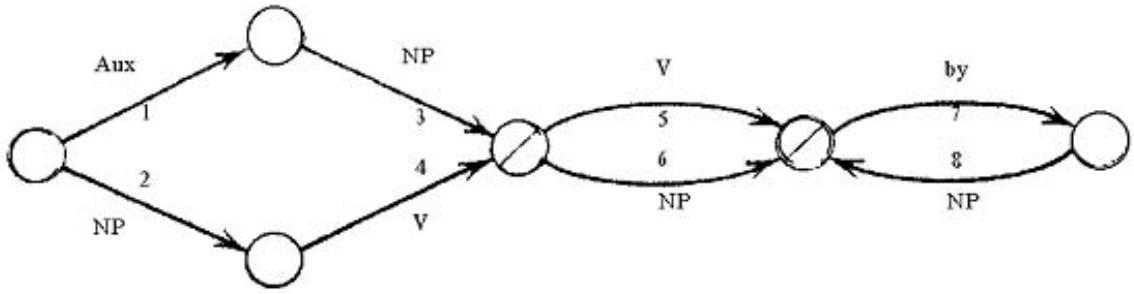

Рис. 13. Расширенная сеть переходов

т.е. *повествовательное*, и система переходит ко второму узлу. Здесь дополнительный тест не требуется, поскольку он отсутствует в списке тестов, записанных на LISP. Следовательно, слово, стоящее после *the rabbit* – т.е. *nibbles* – глагол-сказуемое (обозначение *V* на стрелке), и *nibbles* записывается в ячейку с именем *V*. Перечеркнутый узел означает, что в нем анализ предложения может в принципе закончиться. Но в нашем примере имеется еще и дополнение *the carrot*, так что анализ продолжается по стрелке 6 (выбор между стрелками 5 и 6 осуществляется снова с помощью специального теста), и словосочетание *the carrot* кладется в ячейку с именем *Obj*. На этом анализ заканчивается (последний узел был бы использован в случае анализа такого пассивного предложения, как *The carrot was nibbled by the rabbit*). Таким образом, в результате заполнены регистры (ячейки) *Subj*, *Type*, *V* и *Obj*, используя которые, можно получить какое-либо представление (например, дерево).

Расширенная сеть переходов имеет свои недостатки:

- немодульность;
- сложность при модификации, вызывающая непредвиденные побочные эффекты;
- хрупкость (когда единственная неграмматичность в предложении делает невозможным дальнейший правильный анализ);
- неэффективность при переборе с возвратами, т.к. ошибки на промежуточных стадиях анализа не сохраняются;
- неэффективность с точки зрения смысла, когда с помощью полученного синтаксического представления оказывается невозможным создать правильное семантическое представление.

Семантические грамматики. Анализ ЕЯ, основанный на использовании семантических грамматик, очень похож на синтаксический, с той разницей, что вместо синтаксических категорий используются семантические. Естественно, семантические грамматики работают в узких предметных областях. Примером служит система *Ladder*, встроенная в базу данных американских судов. Ее грамматика содержит записи типа:

S -> <present> the <attribute> of <ship>
<present> -> what is|[can you] tell me
<ship> -> the <shipname>|<classname> class ship

Такая грамматика позволяет анализировать такие запросы, как *Can you tell me the class of the Enterprise?* (*Enterprise* – название корабля). В данной системе анализатор составляет на основе запроса пользователя запрос на языке базы данных.

Недостатки семантических грамматик состоят в том, что, во-первых, необходима разработка отдельной грамматики для каждой предметной области, а во-вторых, они очень быстро увеличиваются в размерах. Способы исправления этих недостатков – использование синтаксического анализа перед семантическим, применение семантических грамматик только в рамках реляционных баз данных с абстрагированием от общеязыковых проблем и комбинация нескольких методов (включая собственно семантическую грамматику).

Анализ с помощью падежных фреймов. С созданием падежных фреймов связан большой скачок в развитии NLP. На сегодняшний день падежные фреймы – один из наиболее часто используемых методов NLP, т.к. он является наиболее компьютерно-эффективным при анализе как снизу вверх (от составляющих к целому), так и сверху вниз (от целого к составляющим).

Падежный фрейм состоит из заголовка и набора ролей (падежей), связанных определенным образом с заголовком. Фрейм для компьютерного анализа отличается от обычного фрейма тем, что отношения между заголовком и ролями определяется семантически, а не синтаксически, т.к. в принципе одному и то же слово может приписываться разные роли, например, существительное может быть как инструментом действия, так и его объектом [2].

Общая структура фрейма такова:

[Заголовочный глагол
  [падежный фрейм
    агент: <активный агент, совершающий действие>
    объект: <объект, над которым совершается действие>
    инструмент: <инструмент, используемый при совершении действия>
    реципиент: <получатель действия – часто косвенное дополнение>
    направление: <цель (обычно физического) действия>
    место: <место, где совершается действие>
    бенефициант: <сущность, в интересах которой совершается действие>
    коагент: <второй агент, помогающий совершать действие>
]]

Например, для фразы *Иван дал мяч Кате* падежный фрейм выглядит так:

[Давать
  [падежный фрейм
    агент: Иван
    объект: мяч
    реципиент: Катя]
  [грам

время: прош
залог: акт]
]

Существуют обязательные, необязательные и запрещенные падежи. Так, для глагола *разбить* обязательным будет падеж *объект* – без него высказывание будет незаконченным. *Место* и *коагент* будут в данном примере необязательными падежами, а *направление* и *реципиент* – запрещенными.

Часто в NLP бывает полезным использовать семантическое представление в как можно более канонической форме. Наиболее известным способом такой репрезентации являются метод **концептуальных зависимостей**, разработанный Шенком для глаголов действия [1]. Он заключается в том, что каждое действие представлено в виде одного или более простейших действий.

Например, для предложений *Иван дал мяч Кате (1)* и *Катя взяла мяч у Ивана (2),* различающихся синтаксически, но оба обозначающих акт передачи, могут быть построены следующие репрезентации с использованием простейшего действия *Atrans*, применяющегося в грамматике концептуальных зависимостей:

(1)
[Atrans
отн: обладание
Агент: Иван
объект: мяч
источник: Иван
реципиент: Катя]

(2)
[Atrans
отн: обладание
агент: Катя
объект: мяч
источник: Иван
реципиент: Катя]

С помощью такого представления легко выявляются сходства и различия фраз.

Для облегчения анализа также используется деление роли на лексический маркер и заполнитель. Так, для роли *объект* может быть установлен маркер *прямое дополнение*, для роли *источник* – маркер вида *<маркер-из>=из|от|...*

В общем случае анализ текста с помощью падежных фреймов состоит из следующих шагов:

1) используя существующие фреймы, подобрать подходящий для заголовка. Если такого нет, текст не может быть проанализирован;

2) вернуть в систему подходящий фрейм с соответствующим заголовком-глаголом;

3) попытаться провести анализ по всем обязательным падежам. Если один или более обязательных заполнителей падежей не найдены, вернуть в систему код ошибки. Такой случай может означать наличие эллипсиса, неверный выбор фрейма, неверно введенный текст или недостаток грамматики. Следующие шаги используются уже для анализа и исправления таких ситуаций;

4) провести анализ по всем необязательным падежам;

5) если после этого во введенном тексте остались непроанализированные элементы, выдать сообщение об ошибке, связанной с неправильным вводом,

недостаточностью данного анализа или необходимостью провести другой, более гибкий анализ.

Преимущества использования падежных фреймов таковы:

- совмещение двух стратегий анализа (сверху вниз и снизу вверх);
- комбинирование синтаксиса и семантики;
- удобство при использовании модульных программ.

Обработка предложений ЕЯ с помощью нейронных сетей. В последнее десятилетие для обработки ЕЯ начали использовать нейронные сети.

Нейронные сети используются для решения следующих задач:

1) распознавание грамматически правильных предложений ЕЯ (используются рекуррентные нейронные сети);

2) представление и распознавание смысла, заключенного в предложении ЕЯ (семантические нейронные сети);

3) кластеризация слов и фраз ЕЯ с целью поиска документов по содержимому (модель Хэмминга, самоорганизующиеся карты Кохонена).

В первом случае происходит обучение сети на обучающей выборке из правильных предложений. При этом в процессе обучения нейронная сеть формирует в своем состоянии и своей структуре грамматику языка.

Во втором случае при обучении происходит формирование нейронных сетей, узлами которых являются слова, словосочетания и концепции, взятые из обработанных предложений ЕЯ.

В третьем случае в процессе обучения происходит разбиение предложений (фраз, слов) по степени их похожести в пространстве признаков.

Контрольные вопросы

1) Какие особенности естественного языка затрудняют его использование в искусственных интеллектуальных системах?

2) В чем природа неоднозначности или нечеткости ЕЯ, как механизма передачи знаний?

3) Чем отличаются синтаксически- и семантически- ориентированные подходы к анализу ЕЯ?

4) Какие недостатки имеет синтаксически-ориентированный анализ ЕЯ?

5) Чем отличается синтаксическая и семантическая грамматика?

6) Что такое падежный фрейм и как он используется при анализе предложения ЕЯ?

7) Примеры прикладных систем, использующих анализ или синтез ЕЯ.

8. СОВРЕМЕННЫЕ ТЕНДЕНЦИИ И ПЕРСПЕКТИВЫ ИИ

Онтологии. Гибридные интеллектуальные системы. Понятие интеллектуального агента и мультиагентных систем. Распределенные интеллектуальные системы.

Литература – 1, 4, 5, 7, 8.

Онтология – это путь дальнейшей структуризации больших баз знаний. Онтология представляет собой по существу самостоятельную базу знаний (или экспертную систему) с декларативными и процедурными знаниями, описанием пользовательского интерфейса, средствами пополнения онтологии. В то же время онтология может рассматриваться как часть более сложной базы знаний, являющейся деревом онтологий. Особенное значение онтологии имеют для размещения баз знаний в Internet.

Существуют специальные инструментальные средств для создания и сопровождения онтологий.

В настоящее время (с конца 80-х годов) наметились тенденции объединения в одной системе логических и/или эмпирических методов представления знаний (вербальных или символьных) с ассоциативными, использующими нейронные сети. В ходе этих исследований появились такие парадигмы как:

- семантические нейронные сети;
- нечеткие нейронные сети;
- "двухполушарные" экспертные системы.

Целью этих исследований является создание систем ИИ, способных обучаться примерно так, как это делает человек, создавая при обучении ассоциативные связи между параметрами внешней среды и символьными понятиями и иерархии понятий, и решать задачи, комбинируя и чередуя ассоциативный поиск и логический вывод. Такие системы искусственного интеллекта получили название гибридных.

Интеллектуальный агент – это интеллектуальная система, моделирующая поведение, мышление и коммуникацию субъектов деятельности. Его особенностью является реализация полного цикла: восприятие – познание – исполнение в определенной внешней среде и модели знаний.

Термином "распределённый искусственный интеллект" обычно обозначают область искусственного интеллекта, в которой уделяется особое внимание способности агентов работать совместно для решения проблем, которые требуют коллективных усилий. Ресурсы, которыми располагают агенты, такие как знания, способности, информация и т.д. распределены в многоагентной среде, из-за чего агенты не могут выполнить задание самостоятельно, или, по крайней мере, могут выполнить его быстрее, сотрудничая с другими агентами.

При решении задач от агентов требуется как умение слаженно взаимодействовать с другими агентами, так и хорошо выполнять те задачи, для которых данный агент предназначен.

Пример распределенной системы, состоящей из нескольких интеллектуальных агентов – команда роботов в "RoboCup" (ежегодный чемпионат мира по футболу среди роботов). Девиз "RoboCup" – к 2050 году создать команду полностью автономных гуманоидных роботов, которые могут обыграть в футбол команду людей. Но такие системы известны и для решения практически значимых задач в области экономики.

9. ТЕМЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

В качестве контрольной работы можно выбрать одну из трех тем:

1. Постановка задачи и идентификация предметной области для разработки экспертной системы.

В этом случае студент должен конкретизировать решаемую задачу, а также, в рамках этапа идентификации должен ответить на соответствующие вопросы (см. выше), а также, привести возможные варианты выбора инструментальных средств:

2. Постановка задачи и анализ предметной области по применению нейронных сетей.

В этом случае студент должен конкретизировать решаемую задачу, определить входные и выходные данные нейронной сети, выбрать модель нейронной сети, продумать кодировку входной и выходной информации.

3. Расширенная постановка задачи и анализ предметной области по применению естественного языка.

В этом случае студент должен конкретизировать решаемую задачу с использованием естественного языка (ЕЯ), роль ЕЯ при решении этой задачи (диалог, синтез или анализ предложений ЕЯ), выбрать метод анализа или синтеза ЕЯ, проработать структуру используемых словарей и структур данных для описания семантики и/или синтаксиса ЕЯ.

В качестве предметной области студент выбирает какую-либо область, связанную с его профессиональной деятельностью. , согласовывая выбор с преподавателем. Взаимодействие с преподавателем осуществляется по электронной почте.

При выполнении контрольной работы студент пользуется (помимо литературы по экспертным системам) электронным учебным пособием А.В.Гаврилов, Ю.В.Новицкая "Разработка экспертных систем", размещенным на сайте кафедры ВТ НГТУ по адресу http://ermak.cs.nstu.ru/metod/ai1.htm и другим методическим обеспечением, расположенным на сайте http://ermak.cs.nstu.ru/islab/

10. ТЕМЫ ЛАБОРАТОРНЫХ РАБОТ

1. Изучение модели многослойного перцептрона на примере решения задачи распознавания образов
2. Изучение модели Хопфилда на примере решения задачи оценки недвижимости
3. Изучение модели Кохонена на примере решения задачи классификации образов
4. Изучение модели Гроссберга-Карпентера (ART-1) на примере решения задачи кластеризации образов

11. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература

1. А.В. Гаврилов. Гибридные интеллектуальные системы. – Новосибирск: НГТУ, 2003.
2. А.В. Гаврилов. Системы искусственного интеллекта. Уч. пособие, ч. 1. – Новосибирск, НГТУ, 2000, 2001.
3. А.В. Гаврилов. Лабораторный практикум по нейронным сетям. Ч.1. – Новосибирск, НГТУ, 2000.
4. Т.А. Гаврилова, В.Ф. Хорошевский. Базы знаний интеллектуальных систем. – СПБ: Питер, 2000.
5. Джексон П. Введение в экспертные системы. – М., СпБ., Киев: "Вильямс", 2001.
6. Дж.Ф. Люгер. Искусственный интеллект: стратегии и методы решения сложных проблем. – М.: "Вильямс", 2003.
7. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2002.
8. Н.Г. Ярушкина. Основы теории нечетких и гибридных систем. – М.: Финансы и статистика, 2004.

Дополнительная литература

9. И. Братко. Программирование на языке ПРОЛОГ для искусственного интеллекта. – М.: Мир, 1990.
10. Искусственный интеллект. Применение в интегрированных производственных системах. Под ред. Э.Кьюсиака. – М.: Машиностроение, 1991.
11. Искусственный интеллект. Справочник в 3-х томах. – М.: Радио и связь, 1990.
12. Р. Каллан Основные концепции нейронных сетей. – М.: "Вильямс", 2001.
13. Е.Ю. Кандрашина, А.В. Литвинцева, Д.А. Поспелов. Представление знаний о времени и пространстве в интеллектуальных системах. – М.: Наука, 1989.
14. В.В. Круглов, В.В. Борисов. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия-Телеком, 2001.
15. Логический подход к искусственному интеллекту. – М.: Мир, 1990.
16. Ж.-Л. Лорьер. Системы искусственного интеллекта. – М: Мир, 1991.
17. А.В. Назаров, А.И. Лоскутов. Нейросетевые алгоритмы прогнозирования и оптимизации систем. – СПб.: Наука и техника, 2003.
18. Обработка знаний. – М: Мир, 1990.
19. С. Осовский. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002.
20. Э.В. Попов. Экспертные системы. – М:, Наука, 1987.
21. Э.В. Попов. Общение с ЭВМ на естественном языке. – М:, Наука, 1986.
22. Э.В. Попов, И.Б. Фоминых, Е.Б. Кисель, М.Д.Шапот. Статические и динамические экспертные системы. – М.: Финансы и статистика, 1996.
23. Построение экспертных систем. Под ред. Ф. Хейес-Рота, Д. Уотермена, Д. Лената. – М.: Мир, 1987.
24. Представление и использование знаний. – М: Мир, 1989.
25. Приобретение знаний. – М: Мир, 1990.
26. Тельнов Ю.Ф. Интеллектуальные информационные системы в экономике. (Учебное пособие) – М., 2002.
27. Уоссермен Ф. Нейрокомпьютерная техника : Теория и практика. – М.: Мир. 1992.
28. Д. Уотерман. Руководство по экспертным системам. – М.: Мир, 1989.

Более полный список литературы по ИИ, электронные учебные и методические пособия, а также, ссылки на ресурсы Internet – http://ermak.cs.nstu.ru/islab/

СОДЕРЖАНИЕ