МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информационных технологий
Кафедра систем информатики

Т. В. Батура

МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА И АВТОМАТИЧЕСКАЯ ОБРАБОТКА ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Учебное пособие

Новосибирск
2016

УДК 004.048:519.765
ББК 32.81
Б287

Рецензент
Н. В. Саломатина, канд. физ.-мат. наук,
Институт математики им. С. Л. Соболева СО РАН

Батура, Т. В.
Б287 Математическая лингвистика и автоматическая обработка текстов : учеб. пособие / Т. В. Батура ; Новосиб. гос. ун-т. – Новосибирск : РИЦ НГУ, 2016. – 166 с.

ISBN 978-5-4437-0548-4

В пособии рассмотрены формальные модели текстов на естественном языке, основанные на методах математической логики, линейной алгебры, теории вероятностей и математической статистики, а также изложены алгоритмы обработки текстов, применяемые для семантического анализа, классификации, поиска и извлечения информации.

Пособие предназначено для студентов и аспирантов ФИТ, ММФ и ГФ (отделение фундаментальной и прикладной лингвистики).

УДК 004.048:519.765
ББК 32.81

ISBN 978-5-4437-0548-4

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Пособие соответствует части курса лекций по дисциплине «Теоретические основы обработки информации», который читается студентам факультета информационных технологий НГУ.

Пособие состоит из введения и восьми глав. После каждой главы приводится список литературы по соответствующей теме. Главы 1–4 знакомят студентов с основами и общими принципами построения систем автоматической обработки текстовой информации. Они подготавливают студентов к дальнейшему обсуждению возникающих проблем и их решению. В главах 5–7 рассматриваются довольно сложные вопросы семантического анализа текстов, подходы и методы математической лингвистики, недостаточно или разрозненно освещенные в литературе. В главе 8 рассматриваются прикладные задачи автоматической обработки текстов: задача определения авторства текстов, задача определения тематической принадлежности и анализа тональности текстов.

При составлении пособия передо мной стояла нелегкая задача – собрать воедино понятия, инструменты и методы из классической математики, классической лингвистики, теории алгоритмов и программирования. У меня было несколько целей: изложить доступным языком накопленный материал, показать взаимосвязи между упомянутыми дисциплинами, важность и полезность направления компьютерной и математической лингвистики, продемонстрировать интересные задачи, способы их решения и пока еще нерешенные проблемы.

Хотя примеры и задачи в пособии разобраны как для русского, так и для английского языков, но мне бы очень хотелось, чтобы это пособие хоть в какой-то мере способствовало развитию именно отечественной компьютерной лингвистики.

ВВЕДЕНИЕ

Математическая лингвистика является ветвью науки об искусственном интеллекте. В 1950-х годах в США начали появляться первые языки программирования, были предприняты попытки проводить эксперименты с машинным переводом текстов научных журналов. С распространением Интернета количество информации, в том числе текстовой информации на естественном языке, стало расти чрезвычайно быстро. Стремительное развитие современного общества и компьютерных технологий требует постоянного совершенствования методов обработки информации.

Естественный язык – язык, используемый для общения людей и не созданный целенаправленно. Примерами естественных языков являются русский, английский, китайский, казахский и др.

Языки, предназначенные для общения людей, обычно противопоставляют **формальным языкам**. К формальным языкам относят язык математической логики; языки программирования; языки, порожденные регулярными выражениями, конечными автоматами, грамматикой Хомского и др.

Языки, созданные целенаправленно, называют **искусственными языками**. На данный момент их уже больше 1000, и постоянно создаются новые. Примеры искусственных языков: эсперанто, ложбан, токипона, эльфийские языки и др.

Целью математической лингвистики является использование математических моделей для описания естественных языков. В математической лингвистике акцент делается на абстрактные модели, позволяющие отображать феномены языка, а в компьютерной лингвистике – на прикладные методы описания и обработки языка для компьютерных систем. Основной деятельностью компьютерных лингвистов является разработка алгоритмов и прикладных программ для обработки текстовой информации.

В настоящее время выделяют следующие основные направления компьютерной лингвистики.

1. Информационный поиск (Information Retrieval). Создание информационно-поисковых систем (направление search engines), например, Google, Яндекс, Yahoo!, AltaVista, Sphinx и др. Для поиска необходимой информации в сети Интернет используются сетевые каталоги и полнотекстовые поисковые системы.

Каталоги устроены по принципу библиографических справочных систем. В них каждая книга или статья находится на определенном месте в предметном или авторском указателе. В сетевом каталоге ссылки рассортированы по тематическим рубрикам и сопровождаются аннотациями. Одной из важнейших функций этих служб является установление соответствия между сетевыми именами, допустим, пользователей или ресурсов, и

сетевыми адресами (или, иными словами, перевод одних в другие). Данная функция, называемая службой имен, позволяет работать с удобопонятными псевдонимами и переводить или отображать эти имена в машинные адреса.

В отличие от каталогов, хранящих только аннотации, поисковые системы Интернета хранят весь текст.

Первые версии программ полнотекстового поиска документа в базе данных предполагали сканирование всего содержимого всех документов в поиске заданного слова или фразы. При использовании такой технологии поиск занимал очень много времени (в зависимости от размера базы), а в Интернете был бы невыполним. Современные алгоритмы заранее формируют для поиска так называемый полнотекстовый индекс – словарь, в котором перечислены все слова и указано, в каких местах они встречаются. При наличии такого индекса достаточно осуществить поиск нужных слов в нем, и тогда сразу же будет получен список документов, в которых они встречаются.

2. Извлечение информации (Information Extraction). Автоматическое извлечение информации из текста (направления fact extraction, text mining). Результатом анализа текста являются выделенные из текста сущности – наименования организаций, персон, географические объекты, различные символьно-цифровые конструкции, классы сущностей; сеть синтактико-семантических отношений между сущностями текста; структуры данных, описывающие упомянутые в тексте события и факты. Пример такой системы – RCO Fact Extractor.

3. Машинный перевод (Machine Translation) – автоматический перевод текста с одного языка на другой. Среди переводчиков популярными сегодня являются, например, PROMT, Google Translate и др.

4. Автореферирование (Automatic Text Summarization) – извлечение наиболее важных сведений из одного или нескольких документов и генерация кратких аннотаций или отчетов на основе их содержимого. Например, TextAnalyst, Extractor, Text Miner.

5. Корпусная лингвистика (Corpus Linguistics) включает в себя создание и использование электронных корпусов текстов. Например, проект «Национальный корпус русского языка».

6. Экспертные системы (Expert Systems). Построение систем управления знаниями или так называемых экспертных систем. Например, WolframAlpha – база знаний и набор вычислительных алгоритмов, интеллектуальный «вычислительный движок знаний»; IBM Watson – суперкомпьютер фирмы IBM, способный понимать вопросы, сформулированные на естественном языке и находить на них ответы в базе данных. Другим примером является интеллектуальная поисковая система Нигма.

7. Вопросно-ответные системы (Question Answering Systems), как правило, содержат 4 основных модуля: классификатор вопросов; поиск в документах частей текста, потенциально содержащих ответ; выделение ответов; генерация текста в качестве ответа. Примеры вопросно-ответных систем: IBM Watson, AskMSR и др.

8. Создание электронных словарей, тезаурусов, онтологий. Словари используют, например, для автоматического перевода, проверки орфографии. Примеры: Lingvo, Мультитран, WordNet, Викисловарь и др.

9. Оптическое распознавание символов (Optical Character Recognition). Например, программа FineReader.

10. Автоматическое распознавание речи (Automatic Speech Recognition) – преобразование речи в текст, например, плагин в Google Chrome для ввода текста при помощи голоса.

11. Автоматический синтез речи (Text-To-Speech) – технология, которая дает возможность произнести текст голосом, приближенным к естественному. Например, есть в Google Translate.

Далее изложены подходы, методы и алгоритмы, используемые для решения задач первых восьми направлений. Последние три направления больше относятся к обработке изображений и сигналов, их рассмотрение выходит за рамки данного пособия.

ГЛАВА 1. **УСТРОЙСТВО СИСТЕМ АВТОМАТИЧЕСКОЙ ОБРАБОТКИ ТЕКСТОВ**

1.1. Основные этапы построения систем автоматической обработки текстов

Выбор формализма для представления лингвистических знаний определяется тремя взаимно-противоречивыми критериями [1]: лингвистическая естественность, формальная мощность и вычислительная эффективность. Под лингвистической естественностью понимается, с одной стороны, удобство отображения феноменов естественного языка, а с другой стороны – типологическая адекватность, т. е. возможность достаточно общим образом описывать феномены, относящиеся ко многим естественным языкам. Следствием этого является представление системы автоматической обработки текстов в виде набора последовательно работающих процессоров.

Компоненты, составляющие структуру систем анализа текстов, – это лингвистические процессоры, которые друг за другом обрабатывают входной текст. Вход одного процессора, как правило, является выходом другого. Выделяют следующие компоненты систем обработки текстов.

Основные этапы построения систем автоматической обработки текстов:

1) графематический анализ (осуществляется на уровне символов);
2) морфологический анализ (осуществляется на уровне слов);
3) фрагментационный анализ (осуществляется на уровне фраз, частей предложения);
4) синтаксический анализ (осуществляется на уровне предложений);
5) семантический анализ (осуществляется на уровне текста).

В последнее время часто упоминают прагматический анализ (дискурс-анализ) в качестве еще одного этапа анализа текста. Его основная задача – определить цель, которую преследует автор при изложении своих мыслей. Но в процесс построения систем автоматической обработки текстов данный этап не включен, т. к. формализация и автоматизация его пока остается только на уровне обсуждений.

Рассмотрим подробнее каждый из этапов.

Графематический анализ. Часто графематический анализ сводится к выполнению только лексического анализа и совмещен с морфологическим анализом. **Лексический анализ (токенизация)** – выделение в тексте слов, цифровых комплексов, знаков препинания, формул и т. д. Все эти выделенные элементы называют **лексемами (токенами).**

Иногда на этапе графематического анализа осуществляют деление текста на более крупные, чем отдельные слова, единицы (абзацы, предложе-

ния). На этом этапе ориентиром для разбивки являются пробелы, отступы, знаки препинания.

Подзадачей графематического анализа является **сегментация** – поиск границ между словами в тексте без пробелов (например, на китайском или японском языках).

Пример

Разбить строку *Itiseasytoreadwordswithoutspaces* на отдельные слова, чтобы получилась фраза.

Для решения задачи сегментации удобно воспользоваться алгоритмом Витерби.

Алгоритм Витерби

Вход:

- text текст;
- voc словарь со словами и их вероятностями.

Выход:

- наилучшее разбиение текста в виде списка слов.

Наилучшим считаем такое разбиение текста на слова, для которого получено наибольшее произведение вероятностей слов.

На рис. 1 приведен листинг алгоритма Витерби на языке Python.

```
def func(text):
    n = len(text)                   #длина исходного текста
    best = [1.0]                    #наилучшие вероятности слов
    words = []                      #список слов, являющихся наилучшими
                                        # для каждой позиции
    for i in range(n):
        best_est = 0.0
        best_word = ''
        for j in range(i+1):        #последовательно рассматриваем слова,
                                        # оканчивающиеся на символ в позиции i;
                                    #началом может быть любой символ в позиции j
            word = text[j:i + 1]
            if word in voc:
                notin = voc[word]  #по ключу, если он есть в словаре,
                                        # находим значение
            else:
                notin = 0
            cur_est = notin*best[j] #оценка текущего слова вычисляется как
                                        # вероятность текущего слова и
                                        # наилучшая вероятность предыдущих слов
            if cur_est > best_est:
                best_est = cur_est  #наилучшая оценка
                best_word = word    #наилучшее слово
        best.append(best_est)
        words.append(best_word)
    result = []                     #осуществляем "размотку" с конца;
                                        # получаем искомый список слов,
                                        # являющийся наилучшим разбиением
```

```
i = n - 1
while i >= 0:
    wordi = words[i]
    if wordi == '':
        return FuncResult(False) #эта функция позволяет обрабатывать
                                 # исключения, если невозможно
                                 # разбить исходный текст на слова
    result.insert(0, wordi)      #перед текущим словом добавляем следующее
    i = i - len(wordi)
  return FuncResult(result)
```

Рис. 1. Алгоритм Витерби

Морфологический анализ может состоять из четырех компонентов:

- стемматизация,
- лемматизация,
- приписывание граммем,
- получение парадигм.

Подробнее принципы построения морфологических анализаторов будут рассмотрены немного позже в данном разделе.

Фрагментационный анализ ставит своей целью деление предложения на неразрывные фрагменты (синтаксические единства), бóльшие или равные словосочетанию (синтаксической группе), и установление частичной иерархии на множестве этих единств. Возможные типы фрагментов: главные предложения, придаточные предложения в составе сложного, причастные, деепричастные и другие обособленные обороты.

Часто этот этап совмещен с синтаксическим анализом.

Действительно, установление иерархии, является важной задачей фрагментационного анализа. Любое простое предложение может быть «разорвано» «вклинивающимися» причастными или деепричастными оборотами или придаточными предложениями, которые, в свою очередь, тоже могут быть «разбиты» другими оборотами и придаточными. Порой «куски» цельного высказывания находятся на значительном расстоянии друг от друга, а глубина вложения таких «клиньев» теоретически не ограничена.

Пример

Вот дом,
Который построил Джек.
А это пшеница,
Которая в темном чулане хранится
В доме,
Который построил Джек.
А это веселая птица-синица,
Которая часто ворует пшеницу,
Которая в темном чулане хранится

В доме,
Который построил Джек.

Иерархия отражает тот факт, что в предложении некоторые фрагменты синтаксически зависимы от других. Так, фрагмент «причастный оборот» будет подчиняться фрагменту, содержащему определяемое слово; придаточное предложение – главному.

Часто этот этап совмещен с синтаксическим анализом. Тем не менее, иногда фрагментационный анализ выделяют как отдельный этап, для того чтобы сократить время работы синтаксического анализатора. В таком случае предложение поступает на вход синтаксическому анализу по фрагментам.

Синтаксический анализ. На этапе синтаксического анализа осуществляется построение групп в предложении.

Основная задача синтаксического анализатора – удаление значительной части «морфологического шума» и омонимичности словоформ. Важно, чтобы при синтаксическом анализе не появлялись лишние синтаксические связи, которые может допускать морфология. Это связано с морфологической неоднозначностью (см. главу 2). Например, слово *мыла* не является морфологически однозначным, т. к. в зависимости от контекста может выступать в роли существительного в единственном числе, в родительном падеже, или в роли глагола в прошедшем времени. Подробнее методы синтаксического анализа будут рассмотрены далее.

Семантический анализ. Цель семантического анализа – построить семантический граф текста. На этапе семантического анализа, в отличие от морфологического и синтаксического, появляется формальное представление смысла текста. В сферу семантического анализа входит построение семантической интерпретации слов и конструкций и установление «содержательных» семантических отношений между элементами текста, которые уже не ограничены размером одного слова.

Наиболее полное и законченное представление текста из всех возможных получается именно на этапе семантического анализа. Семантический анализ – самая труднорешаемая задача из пяти перечисленных в автоматической обработке текстов, для которой на сегодняшний день пытаются найти наилучшее приближение.

Наиболее полный список инструментов для автоматического анализа текстов приведен на сайте NLPub (http://nlpub.ru/Обработка_текста).

Существует много готовых морфологических и синтаксических анализаторов как с открытым, так и с закрытым кодом: MyStem, AOT, LinkGrammar и др.

Также есть большое разнообразие средств разработки для создания своих приложений: Solarix (http://www.solarix.ru); библиотеки для Python (pymorphy2, nltk); библиотеки для PHP (phpMorphy) и пр.

1.2. Принципы построения морфологических анализаторов

Для дальнейших рассуждений нам понадобятся некоторые понятия и термины из классической лингвистики.

Морфема – минимальная значимая часть слова.

Корень – морфема, несущая лексическое значение слова или основную часть этого значения.

Аффикс – морфема, видоизменяющая значение слова или отражающая отношение между словами. Например, префикс и постфикс, а по-простому – приставка и суффикс.

Пример

Алломорф – разновидность, вариант морфемы; каждая реализация алломорфа в речи – **морф**. Совокупность морфов – это целый класс вариантов морфемы. Имя класса выбирается из элементов класса.

Пример

Лексемы: морозы, изморозь, мороженое, вымораживать

Алломорфы: *мороз, мороз', морож, мораж*

Морфема: *мороз*

Парадигма (словоизменительная) – список всех форм слова. Например, существительное во всех падежах и числах (*стол, стола, столу, столом, столе, столы, столов, столам, столах, столами*).

По типу выражения грамматических характеристик обычно выделяют:

- аналитические языки;
- синтетические (флективные) языки.

В **аналитических** языках грамматические значения выражаются не формами слов, а отдельными служебными словами (местоимениями, предлогами, частицами). Например, английский и французский языки.

В **синтетических** языках грамматические значения выражаются с помощью словоизменения, т. е. аффиксами в составе словоформы. Например, русский и немецкий языки.

При аналитическом выражении грамматических значений слова типично состоят из малого числа морфем, при синтетическом – из нескольких.

Общая степень сложности морфологической структуры слова выражается количеством морфов, приходящимся в среднем на одну словоформу. Это называется **индексом синтетичности**, который вычисляется по формуле

$$M / W \text{, где}$$

M – количество морфов в отрезке текста;

W – количество слов в этом же отрезке текста.

Для подсчета берутся типичные тексты на соответствующем языке, длиной не менее 100 словоупотреблений.

Аналитическими являются языки с величиной индекса ниже 2, например, для английского получается 1,68 (на 100 слов 168 морфов).

Синтетическими являются языки с величиной индекса от 2 до 3, например, для русского языка – примерно от 2,33 до 2,45.

С точки зрения типов морфологической структуры выделяют языки:

- изолирующие (морфемы максимально отделены друг от друга);
- агглютинативные (морфемы семантически отделены, но реально объединены в слова);
- флективные (семантические и формальные границы между морфемами плохо различимы).

В **изолирующих** языках слово обычно равняется корню, а отношения между словами передаются прежде всего синтетически (порядком слов, служебными словами, ритмом, интонацией). К изолирующим языкам относится, например, китайский.

Структура слова в **агглютинативных** языках характеризуется большим количеством аффиксов, обычно прибавляемых к неизменяемой основе слова. Каждый из аффиксов имеет только одно, строго определенное значение. Располагаются аффиксы в определенном порядке, но вместе с тем довольно подвижны (могут быть пропущены). Примерами агглютинативных языков являются турецкий и казахский языки.

Для **флективных** языков характерна многозначность грамматических морфем и крепкая спаянность всех морфем в слове, не позволяющая им сравнительно свободно передвигаться внутри слова, как в языках агглютинативных.

Из сказанного ясно, что особую сложность представляет создание морфологических анализаторов, работающих с флективными языками. Менее трудной задачей является автоматизация морфологического анализа для агглютинативных языков.

Рассмотрим теперь принципы построения морфологических анализаторов. Как уже упоминалось ранее, морфологический анализатор может со-

стоять из четырех компонентов: стемматизации, лемматизации, приписывания граммем и получения парадигм.

Стемматизация (или **стемминг** от англ. stemming) – это процесс нахождения основы слова для заданного исходного слова. Отметим, что получаемая псевдооснова не обязана совпадать с грамматической основой рассматриваемой словоформы; достаточно, чтобы словоформы, соответствующие одной парадигме, получали в результате работы алгоритма одну и ту же псевдооснову.

Пример

Для словоформ *зеленый, зелень, зеленеть, зеленеющий* в результате стемматизации будет получена псевдооснова *зелен*.

Лемматизация – процесс приведения словоформы к лемме. **Лемма** – нормальная (словарная) форма слова.

Пример

Лемма имени существительного – это форма слова в единственном числе (если оно есть у существительного) и именительном падеже. Словоформе *столов* соответствует лемма *СТОЛ*.

Лемма имени прилагательного – это форма слова в единственном числе мужского рода. Словоформе *зеленых* соответствует лемма *ЗЕЛЕНЫЙ*.

Приписывание словоформе множества наборов граммем – приписывание словоформе грамматических характеристик (грамматических признаков). **Граммема (грамматическая характеристика)** – это элементарный морфологический описатель, относящий словоформу к какому-то морфологическому классу.

Пример

Словоформе *стол* с леммой *СТОЛ* будет приписан следующий набор граммем: (мр, ед, им, неод) и (мр, ед, вн, неод).

Некоторые граммемы для глаголов:

св, нс – совершенный, несовершенный вид;

нст, прш, буд – настоящее, прошедшее, будущее время;

1л, 2л, 3л – первое, второе, третье лицо.

Большую роль здесь играет омонимичность словоформ.

Пример

У словоформы *стали* может быть две леммы: *сталь* – существительное и *стать* – глагол. Поэтому слову *стали* соответствует следующий набор граммем (жр, мн, им, неод); (жр, ед, рп, неод); (св, прш, 3л).

Получение парадигм – процесс построения всех форм слова по начальной форме.

Из предыдущего примера видно, что морфологического анализа явно не достаточно для выбора одной конкретной морфологической интерпретации слова. К тому же, выбор одной интерпретации может повлиять на выбор интерпретации для соседних слов. Поэтому программы работают с

целым набором возможных морфологических интерпретаций, постепенно выделяя наиболее вероятные на следующих этапах анализа.

Выделяют [2] два типа морфологических анализаторов: словарные и бессловарные.

Словарные анализаторы используют методы, основанные на словарях. В словарных анализаторах преобразование слова в лемму производится с помощью специальной таблицы (словаря), которая содержит отображение множества слов на множество лемм.

Бессловарные анализаторы используют аналитические методы. Бессловарные анализаторы содержат набор правил морфологических преобразований. Для русского языка это в основном таблицы суффиксов и условий их отсечения, с помощью которых данное слово преобразуется в некоторую нормальную форму.

Остановимся подробнее на каждом из этих типов анализаторов.

Методы, основанные на словарях, позволяют определять грамматические характеристики, без которых становится невозможно, например, проводить фрагментацию текстов. В морфологическом словаре приводятся все формы слов, каждой из которых сопоставлены все необходимые признаки (в частности, грамматические: число, падеж, лицо, время, наклонение и др.).

Самым наглядным примером морфологического словаря, используемого при автоматическом анализе, является словарь А. А. Зализняка (http://zaliznyak-dict.narod.ru). Он содержит около 100 тыс. базовых словоформ русского языка с их полным морфологическим описанием. Этот словарь является обратным – слова в нем упорядочены, начиная с последней буквы. Электронная версия этого словаря легла в основу большинства современных компьютерных программ, работающих с русской морфологией.

В словаре А. А. Зализняка для каждого слова указаны его грамматические характеристики. Для существительных указываются род, тип склонения, особенности изменения по падежам и т. д. Для глаголов – вид, особенности спряжения и т. д. В силу полноты морфологической информации этот словарь применяется в качестве основы для создания компьютерных морфологических словарей русского языка.

Словарные анализаторы опираются на принцип: у каждого слова существуют характерные окончания, по которым определяется часть речи. Отделив эти окончания (и, если потребуется, суффиксы), можно определить псевдооснову и основную форму слова. Иначе говоря, конкретным словам ставятся в соответствие векторы окончаний для каждой части речи. Иногда, правда, разные окончания требуют чередования в основе: *водить* –

вожу; есть супплетивные формы: *идти – шел, человек – люди*). Длина вектора окончаний равна числу словоформ данной части речи.

Есть методы определения грамматических характеристик по аналогии: сравниваем неизвестное слово посимвольно с конца со словарными формами, чтобы определить максимальную общую подпоследовательность неизвестного слова со словарным. Приписываем словарные грамматические признаки неизвестному слову.

Главный недостаток словарных анализаторов состоит в том, что получить какую-либо морфологическую информацию об анализируемой словоформе невозможно, если ее нет в словаре. Учитывая, что неизменяемое лексическое ядро естественного языка составляет порядка 80 % словарного запаса, применение только словарных анализаторов не решает проблему определения всех возможных слов в полном объеме.

Аналитические методы (не использующие словари) не решают все задачи морфологического анализа: трудно определить часть речи и грамматические признаки словоформы. Однако аналитические алгоритмы оказываются эффективны для задач индексации текстовых массивов, создания процедур работы со словарями естественных языков.

К наиболее популярным аналитическим методам относятся алгоритмы, основанные на отсечении аффиксов, или те, в которых последовательно применяются заранее определенные правила преобразования аффиксов, позволяющие получить сразу не основу, а нормальную форму или парадигмы. Так или иначе путь к морфологическому анализу лежит через стемматизацию. Рассмотрим более подробно следующие алгоритмы выделения основ:

- алгоритм Ловинса (Lovins, 1968);
- алгоритм Портера (Porter, 1980);
- алгоритм Пейса-Хаска (Paice/Husk, 1990).

Алгоритм Ловинса является самым быстрым алгоритмом выделения основ, но он требует работы лингвистов по составлению списка правил и исключений.

Определено 294 окончания, каждое из которых может быть удалено при выполнении одного из 29 условий.

Определяются 35 правил трансформации словоформ после отсечения окончаний. Алгоритм выбирает наиболее длинное окончание, которое можно отсечь при выполнении условий и ограничений.

Пример

У слова *nationally* получаем основу *nat.*

Существует два окончания, которые могут быть удалены: *ationally* и *ionally.*

Но первое не может быть удалено ввиду ограничения: выделенная основа должна быть длиннее трех символов.

Алгоритм Портера. В зависимости от выполнения условий может происходить одно из определенных преобразований окончаний, определяемое правилом. Правило имеет вид:

<условие><окончание> → <новое окончание>.

Алгоритм состоит примерно из 60 правил, каждое из которых последовательно применяется к поступившей на вход словоформе.

Пример

(m>0) eed → ee

agreed → agree

Используется правило: если в словоформе есть хотя бы одна гласная буква и словоформа оканчивается на согласную и окончание *–eed*, то окончание заменяется на *–ee*.

Для уменьшения ошибок выделения основы для форм, образующихся с исключением, вводят специальную таблицу исключений (например, для неправильных глаголов).

Ланкастерский алгоритм (алгоритм Пейса-Хаска). Используется таблица правил, каждое из которых задает преобразование окончания (удаление или замену). Замены используются для оптимизации алгоритма с целью сокращения числа дополнительных шагов, которые происходят уже после удаления суффиксов. Правила в таблице индексируются последними буквами соответствующих окончаний. Каждое правило состоит из следующих частей:

- окончание, представляемое в инвертированном виде;
- опциональный флаг целостности «*»;
- число, указывающее длину удаляемого окончания (включая 0);
- опциональная добавляемая строка длиной один или более символов;
- управляющие символы («>» или «.»).

Пример

Правило «nois4j>» указывает на то, что окончание *sion* должно быть заменено на *j*. Затем, так как в правиле указан символ продолжения, в таблице окончаний мы переходим на запись с индексом *j*. Если бы в правиле вместо символа «>» стоял символ окончания «.», то алгоритм завершил бы работу после применения этого правила.

Для повышения эффективности решения проблемы аналитического выделения основ на практике применяется смешанный подход, т. е. наряду с правилами преобразования аффиксов рассматриваются таблицы, напри-

мер, неправильных глаголов или исключений при образовании форм множественного числа имен существительных.

Рассмотренные алгоритмы аналитического выделения основ, базирующиеся на отсечении аффиксов, разрабатываются для каждого конкретного языка в соответствии с его грамматикой. Разработчики информационных систем склонны рассматривать это обстоятельство как существенное ограничение для создания процедур автоматической обработки текстов на естественных языках. Поэтому предпринимались попытки создания алгоритмов, которым были бы не нужны априорные знания о языке и которые можно было бы рассматривать как универсальные для решения задачи автоматического выделения основы. Такие алгоритмы используют частоты *n*-грамм, скрытые марковские модели и нейронные сети. Универсальные методы выделения основ, не опирающиеся на грамматику естественного языка, требуют обработки больших объемов текстовой информации для обучения системы или в процессе работы самой информационной системы. Это обстоятельство не позволяет достигать высоких скоростей и качества обработки текстов, что делает их плохо пригодными к использованию в реальных системах.

Эффективность алгоритмов аналитического выделения основ оценивается по определенным характеристикам: коэффициенту недостаточности отсечения; коэффициенту избыточности отсечения; количеству слов, относящихся к одной парадигме; коэффициенту сжатия индекса; модифицированному расстоянию Хэмминга.

Коэффициент недостаточности отсечения (Under-stemming Index) – процент анализируемых словоформ, для которых должна была быть получена одна и та же основа, но в результате применения алгоритма были получены различные основы.

Коэффициент избыточности отсечения (Over-stemming Index) – процент анализируемых словоформ, для которых должны были быть получены различные основы, но в результате применения алгоритма были получены одинаковые основы.

Для вычисления величин UI и OI требуется наличие заранее размеченного корпуса текстов.

Количество слов, относящихся к одной парадигме:

$$MVC = \frac{N}{S}, \text{ где}$$

N – количество уникальных словоупотреблений до применения алгоритма;

S – количество уникальных основ, полученных после применения алгоритма.

Коэффициент сжатия индекса показывает, насколько коллекция уникальных основ меньше коллекции уникальных словоформ:

$$ICF = \frac{N - S}{N}, \text{ где}$$

N – количество уникальных словоупотреблений до применения алгоритма;

S – количество уникальных основ, полученных после применения алгоритма.

Модифицированное расстояние Хэмминга. Классическое расстояние Хэмминга определяется как количество соответствующих позиций, в которых различаются две одинаковые по длине строки (последовательности). Если строки разной длины, то можно использовать модифицированное расстояние Хэмминга:

$$MHD = HD(1, P) + (Q - P), \text{ где}$$

$HD(1, P)$ – расстояние Хэмминга, посчитанное для первых P символов анализируемых строк $(P < Q)$.

Пример

В результате применения алгоритма аналитического выделения основы для словоформы *parties* была получена основа *party*. В этом случае модифицированное расстояние Хэмминга равно: $MHD(party, parties) = 1 + 2 = 3$, потому что $HD(1, P) = 1$, где $P = 5$, определяется в результате анализа двух строк: *party* и *parti*.

Список литературы

1. Сокирко А. В. Семантические словари в автоматической обработке текста // Дисс. канд. тех. наук, МГПИИЯ, Москва, 2000. 108 с.
2. Болховитянов А. В., Чеповский А. М. Алгоритмы морфологического анализа компьютерной лингвистики : учеб. пособие. М. : МГУП имени Ивана Федорова, 2013. 198 с.
3. Норвиг П., Рассел С. Искусственный интеллект: современный подход. М. : Вильямс, 2007. 1424 с.
4. Jurafsky D., Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. 2008. 1024 p.

ГЛАВА 2. **МЕТОДЫ ЗАДАНИЯ СИНТАКСИЧЕСКОЙ СТРУКТУРЫ ПРЕДЛОЖЕНИЙ**

После того как произведен морфологический анализ каждого слова, система автоматической обработки начинает осуществлять синтаксический анализ, позволяющий определять взаимосвязи между отдельными словами и частями предложения.

Как правило, синтаксическая структура предложения – это граф, узлами которого выступают слова предложения. При этом, если два слова связаны каким-либо образом, то соответствующие им вершины графа связаны дугой с определенной окраской. Возможные окраски дуг зависят от языка, на котором написано предложение, а также от выбранного способа представления синтаксической структуры предложения.

При синтаксическом анализе предложений русского языка для окраски дуг можно использовать вопросы, задаваемые от одного слова к другому. Некоторым словам (например, предлогам) вообще не соответствует ни одна из вершин графа, но эти слова влияют на вопросы, задаваемые от одного слова к другому.

В общем понимании **синтаксический анализ** – это процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой.

Результатом синтаксического анализа является синтаксическая структура предложения, представленная в виде системы составляющих (дерева составляющих) или дерева зависимостей (дерева подчинения).

Остановимся подробнее на этих двух методах задания синтаксической структуры предложений.

2.1. Системы составляющих

Пусть x – произвольная непустая цепочка в словаре V. Множество C отрезков цепочки x называется **системой составляющих** этой цепочки, если оно удовлетворяет следующим двум условиям:

1) C содержит отрезок, состоящий из всех точек цепочки x, и все одноточечные отрезки x;
2) любые два отрезка из C либо не пересекаются, либо один из них содержится в другом.

Элементы C мы будем называть **составляющими**. Одноточечные отрезки называются **точечными составляющими.** Отрезок, состоящий из всех точек цепочки, называется **полной составляющей**. Полную и точечные составляющие называют **тривиальными**, остальные составляющие – **нетривиальными.**

Для наглядного изображения системы составляющих можно пользоваться следующим простым способом: заключать каждую нетривиальную составляющую в скобки, причем левую и правую скобки, отвечающие одной составляющей, помечать одной и той же меткой так, чтобы разные пары скобок были помечены разными метками; в качестве меток можно использовать натуральные числа. Можно обойтись и без меток, так как для каждой левой скобки можно однозначно указать соответствующую ей правую.

Если цепочку интерпретировать как предложение естественного языка, то система составляющих может быть использована в качестве способа выражения информации о его синтаксической структуре.

Пример

Предложение *Онегин, добрый мой приятель, родился на брегах Невы* допускает следующую «естественную» систему составляющих: *(Онегин, (добрый (мой приятель))),(родился (на (брегах Невы)))*.

Среди многочисленных систем составляющих, которые имеет предложение естественного языка, лишь весьма немногие «правильны», т. е. адекватно отражают синтаксическое строение предложения. При этом понятие «правильной» системы составляющих не абсолютно. Оно зависит от соглашений лингвистического характера, отражающих определенные содержательные представления о синтаксической структуре предложения данного языка.

При фиксированной системе соглашений предложение также может иметь несколько «правильных» систем составляющих. Нередко эти системы соответствуют различным толкованиям смысла предложения. В лингвистике такое явление – наличие у одного предложения двух или более правильных «синтаксических анализов» – известно под названием синтаксической омонимии.

Пример

Допустим, есть две возможных системы составляющих для одного и того же предложения:

(Все (эти известия)) (произвели (на меня)(удручающее впечатление));

(Все (эти известия)) произвели (на меня) (удручающее впечатление).

Первая система отвечает традиционному представлению о подлежащем и сказуемом как о главных членах предложения. Вторая – соответствует взгляду на подлежащее и дополнение как на «актанты», равноправно подчиненные сказуемому (*произвели* – главное слово, к нему равноправно относятся *все эти известия*, *удручающее впечатление* и *на меня*).

Если A и B – составляющие некоторой системы C, то выражение «B **непосредственно вложена в** A» (B **непосредственная составляющая** A, будем писать в этом случае $B \subset\subset A$) означает, что $B \subset A$ и в C нет составляющей, вложенной в A, содержащей B и отличной от A и от B.

Нетрудно видеть, что граф $\langle C, \subset\subset \rangle$ является деревом, корнем которого служит полная составляющая, а висячими узлами – точечные составляющие.

В ранее приведенных примерах была использована скобочная запись. На рис. 2 представлена система составляющих в виде бинарного дерева. Система составляющих C – это целое множество отрезков, в том числе отдельные слова и предложение целиком.

A и B – непосредственные составляющие.

A и D – нет, т. к. $(\exists B \in C : B \subset A \wedge D \subset B \wedge A \neq B \wedge D \neq B)$.

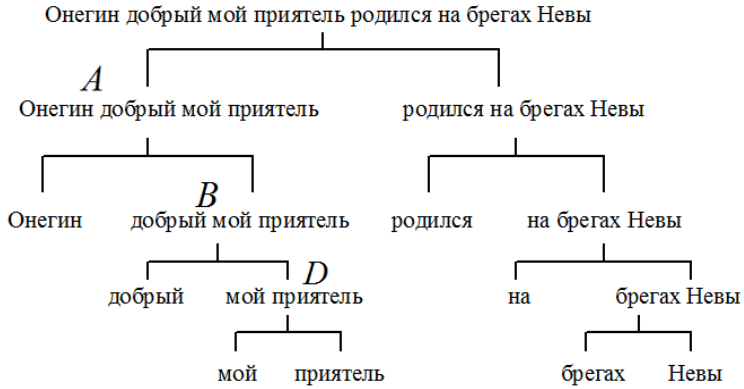


Рис. 2. Пример системы составляющих, представленной в виде дерева

Другими словами, основная идея грамматики составляющих заключается в следующем: всякая сложная грамматическая единица складывается из двух более простых и не пересекающихся единиц, называемых ее непосредственными составляющими.

Рассматривая составляющую как узел дерева составляющих, можно определить ее различные характеристики так же, как для обычных деревев (в теории графов). В частности, **ранг** – максимальное число дуг пути, соединяющих начальную вершину с выбранной. Ранги всегда определены, если граф направленный. Систему составляющих, имеющую бинарное дерево, называют **бинарной.**

Важными характеристиками составляющей являются также степень левого ветвления и степень правого ветвления. Известна гипотеза В. Ингве, согласно которой в естественных языках имеются специальные механизмы, обеспечивающие для большинства предложений ограниченность степеней левого ветвления, в то время как степени правого ветвления ничем в принципе не ограничены. Имеются ввиду конструкции следующего типа:

Вот кот,
Который пугает и ловит синицу,
Которая часто ворует пшеницу,
Которая в темном чулане хранится
В доме,
Который построил Джек.

Существуют, однако, языки, для которых это предположение об ограниченности степеней левого ветвления заведомо не подтверждается, например, венгерский.

Чтобы добавить в систему составляющих дополнительную информацию, рассматривается отображение этой системы составляющих во множество всех подмножеств некоторого конечного множества, элементы которого называются метками.

Упорядоченную тройку $\langle C, W, \varphi \rangle$ будем называть **размеченной системой составляющих**, где

C – система составляющих,

W – множество меток,

φ – отображение C в 2^W.

Составляющая, включающая более одного слова, называется **группой**, а слово, соответствующее корневому узлу в дереве, которое описывает группу, – **вершиной группы.**

Метки содержательно интерпретируются как разновидности синтаксических групп слов и словосочетаний. Обычно выделяют следующие виды синтаксических групп (фразовые категории):

- именная группа (группа существительного, ИГ; англ. noun phrase, NP) – возглавляется существительным;
- группа прилагательного (ГПрил; англ. adjectival phrase, AP) – возглавляется прилагательным;
- наречная группа (НарГ; англ. adverbial phrase, AdvP) – возглавляется наречием;
- предложная группа (ПрГ; англ. prepositional phrase, PP) – возглавляется предлогом;
- глагольная группа (ГГ; англ. verb phrase, VP) – возглавляется глаголом;
- предложение (П; англ. sentence, S).

Некоторые фразовые категории, в частности, именная группа и предложение, обладают свойством рекурсивности – способностью включать в себя составляющие той же фразовой категории.

Множество меток W может содержать следующие элементы:

ПРЕДЛ – «предложение»;

$\tilde{V}^{t}_{xyvw}$ – «группа переходного глагола (согласуется в существительным в ВП без предлога: *читать книгу*) в роде *x*, числе *y*, времени *v* и лице *w*»;

V^{t}_{xyvw} – «переходный глагол в роде *x*, числе *y*, времени *v* и лице *w*»;

$\tilde{V}^{i}_{xyvw}$ – «группа непереходного глагола в роде *x*, числе *y*, времени *v* и лице *w*»;

V^{i}_{xyvw} – «непереходный глагол в роде *x*, числе *y*, времени *v* и лице *w*»;

$\tilde{S}_{xyz}$ – «группа существительного рода *x* в числе *y* и падеже *z*»;

S_{xyz} – «существительное рода *x* в числе *y* и падеже *z*»;

A_{xyz} – «прилагательное в роде *x,* числе *y* и падеже *z*»;

НА – «предложная группа с предлогом *на*»;

НА – «предлог *на*» и т. д.

Одна из проблем грамматики составляющих – в снятии неоднозначностей (синтаксической омонимии). В русском языке существует относительно свободный порядок слов в предложении, поэтому одному и тому же предложению будут соответствовать несколько синтаксических деревьев. Для таких случаев грамматика составляющих подходит не очень хорошо, гораздо удобнее использовать грамматику зависимостей.

2.2. Деревья зависимостей

В грамматике зависимостей порядок слов в предложении не важен. Главное – знать, от какого слова зависит каждое слово в предложении и каким типом связи обозначена эта зависимость.

Пусть x – произвольная непустая цепочка в словаре V, X – множество всех элементов цепочки x.

Произвольное бинарное отношение $\rightarrow$ на X такое, что граф $\langle X, \rightarrow \rangle$ является деревом, называется **отношением зависимости** (или **отношением синтаксического подчинения**) для x.

Другими словами, если считать слова элементами цепочки, то цепочка x – это последовательность слов в предложении, а X – это множество всех слов предложения.

Дерево зависимости можно естественно изобразить графически

(рис. 3). Для дерева зависимости обычным образом определяются зависимость, группа зависимости, ширина дерева, ранг узла и дерева.

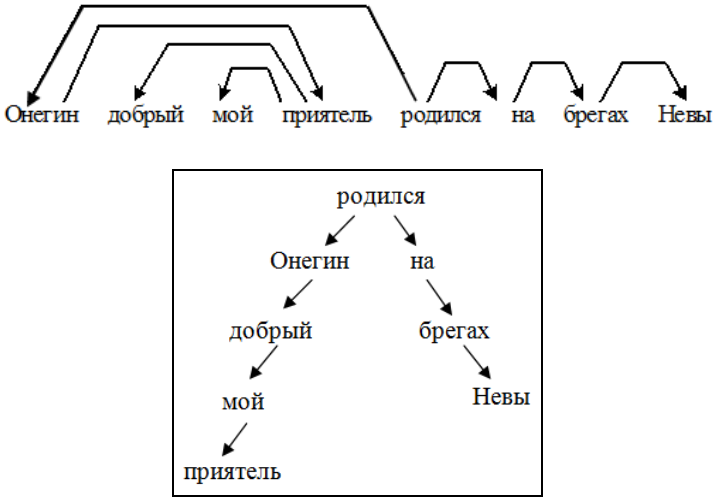


Рис. 3. Дерево зависимостей для предложения

Для анализа предложений естественного языка используются размеченные деревья зависимостей (рис. 4).

Размеченное дерево зависимостей для цепочки x – это четверка $\langle X, \rightarrow, Z, \psi \rangle$, где

$\langle X, \rightarrow \rangle$ – дерево зависимостей для x,

Z – конечное множество (его элементы называются **метками**),

ψ – отображение множества дуг дерева $\langle X, \rightarrow \rangle$ в Z.

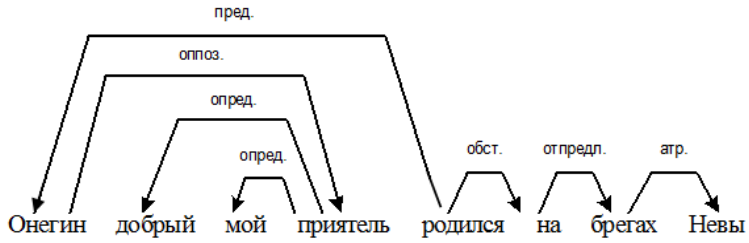


Рис. 4. Размеченное дерево зависимостей

В классе всевозможных деревьев зависимостей можно выделить подкласс, который содержит подавляющее большинство деревьев для пред-

ложений реальных языков. Это класс так называемых проективных деревьев. В них наблюдается некоторая упорядоченность слов.

Дерево зависимостей $\langle X, \rightarrow \rangle$ для цепочки x называется **проективным**, если для любых трех точек α, β, γ цепочки x из того, что $\alpha \rightarrow \beta$ и γ лежит между α и β, следует, что γ зависит от α.

Еще один важный класс деревьев зависимостей (являющийся расширением предыдущего) – класс слабо проективных деревьев.

Дерево зависимостей $\langle X, \rightarrow \rangle$ для цепочки x называется **слабо проективным**, если для любых четырех точек $\alpha, \beta, \gamma, \delta$ цепочки x из $\alpha \rightarrow \beta$ и $\gamma \rightarrow \delta$ следует, что пары α, β и γ, δ не разделяют друг друга.

При графическом способе изображения деревьев зависимостей слабая проективность равносильна возможности провести все стрелки так, чтобы никакие две из них не пересекались.

Содержательный смысл условий проективности и слабой проективности может быть охарактеризован приблизительно так: при выполнении этих условий слова, близкие синтаксически, близки и по положению в тексте. При этом проективность обеспечивает «более тесную» текстовую близость.

В художественной литературе, особенно в поэзии, отклонения от слабой проективности и тем более от проективности вполне обычны. В русской разговорной речи возможен свободный порядок слов, и при определенной интонации предложение все равно может быть понятным и допустимым, т. е. условие слабой прективности также не соблюдается. Напротив, в научной и деловой прозе (по крайней мере, русской) деревья подчинения подавляющего большинства предложений слабо проективны и даже проективны.

Программы синтаксического анализа включают в себя косвенно или в явном виде фильтр на непроективность. Требование проективности синтаксической структуры предложения универсально для большинства индоевропейских языков.

Для приведенного на рис. 3 дерева зависимостей выполняются условия проективности и слабой проективности. К примеру, условие проективности выполняется, если возьмем α = *Онегин*, β = *добрый* или β = *мой*, γ = *приятель*; условие слабой проективности выполняется, если возьмем α = *Онегин*, β = *родился*, γ = *брегах*, δ = *Невы*.

На рис. 5 приведено сравнение деревьев составляющих и деревьев зависимостей. Основные отличия можно сформулировать следующим образом.

1. Деревья зависимостей минимальны по сравнению с деревьями составляющих, т. к. обычно содержат меньшее число вершин.

2. Деревья зависимостей отражают не реальный порядок слов в предложении, а только иерархию связей.
3. Грамматика составляющих опирается на бинарное разбиение (выделяется именная и глагольная группы), а в грамматике зависимостей глагол является основой всей структуры предложения.

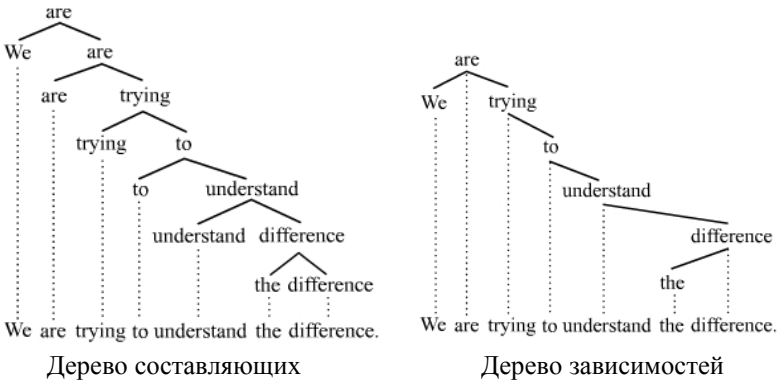


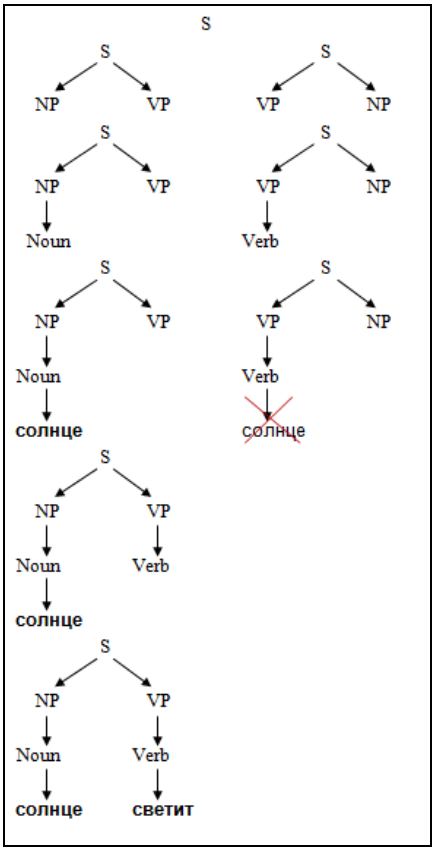
Рис. 5. Сравнение деревьев составляющих и деревьев зависимостей

Таким образом, грамматика зависимостей имеет два главных преимущества. Во-первых, условия проективности и слабой проективности в некоторых случаях помогают «предсказывать» связи между словами или исключать невозможные варианты разбора. Во-вторых, грамматика зависимостей хорошо отражает специфику языков с произвольным порядком слов и позволяет формально описать строение языковых конструкций.

Выделяют два типа алгоритмов синтаксического анализа: нисходящий и восходящий, как для языков программирования. В нисходящем парсере (top-down parser) продукции грамматики раскрываются, начиная со стартового символа, до получения требуемой последовательности токенов. Например, метод рекурсивного спуска, LL-анализатор и др. В восходящем парсере (bottom-up parser) продукции восстанавливаются из правых частей, начиная с токенов и кончая стартовым символом. Например, LR-анализатор, GLR-парсер и др.

Нисходящий синтаксический анализ (рис. 6), или анализ через синтез, начинает с выдвижения предположений о крупномасштабной структуре предложения, а затем уточняет и детализирует это предположение, рекурсивно опускаясь на уровень конкретных слов. Слово «нисходящий» в определении структурного парсера подчеркивает фундаментальную особенность данного алгоритма. Он выполняет разбор предложения с помощью выдвижения альтернативных гипотез и их доказательства или опро-

вержения. Разбор начинается с выдвижения гипотез самого общего вида по поводу структуры предложения (например, вопрос, утверждение, вежливая побудительная конструкция). Для каждой из выдвинутых гипотез определяется набор гипотез, чье доказательство необходимо для их подтверждения или опровержения. Процесс доказательства гипотез продолжается до тех пор, пока не будет найдена необходимая комбинация токенов. Процесс доказательства гипотез может выполняться рекурсивно. Например, при разборе группы существительного с правым причастным оборотом может возникнуть необходимость разобрать входящие в состав оборота другие, более мелкие группы существительного, которые также могут содержать причастный оборот, и так далее.



Набор правил:

$S \rightarrow NP\ \ VP$

$S \rightarrow VP\ NP$

$NP \rightarrow Noun$

$VP \rightarrow Verb$

$Noun \rightarrow солнце \mid луна$

$Verb \rightarrow светит \mid всходит$

Рис. 6. Нисходящий синтаксический анализ

Восходящий синтаксический анализ (рис. 7) начинает разбор с конкретных слов. Сначала связываются пары слов, затем к этим парам присоединяются новые слова или другие связанные пары. Постепенно все слова предложения оказываются связаны в единую структуру.

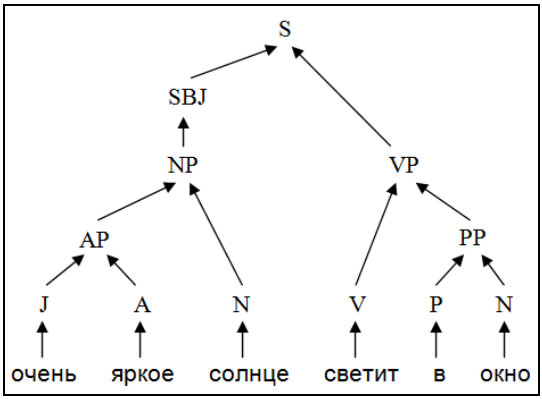


Рис. 7. Восходящий синтаксический анализ

На рис. 7 приняты следующие обозначения: J – наречие; A – прилагательное; N – существительное; V – глагол; P – предлог; AP – группа прилагательного; NP – группа существительного; NP-SBJ – именная группа – субъект (это разновидность именной группы); PP – предложная группа (предлог с существительным); VP – глагольная группа; S – предложение.

Оба алгоритма недетерминированы, то есть могут выдавать множество альтернативных вариантов синтаксического разбора. Это крайне важно для текстов на естественном языке, в которых грамматическая трактовка токенов зависит от контекста заранее неизвестной ширины.

2.3. Проблемы синтаксического анализа

В рамках синтаксического анализа предложения на сегодняшний день успешно решена и уже нашла применение в производстве задача автоматического выделения именных групп. В частности, при извлечении информации из текста нередко возникает необходимость распознать именованные элементы (имена людей, названия организаций, географические названия, события, временны́е и денежные обозначения и пр.) или определить отношения между выделенными сущностями. Однако созданию качественного автоматического синтаксического парсера препятствует ряд проблем. К ним относятся:

- морфологическая омонимия (part-of-speech tagging, word-category disambiguation);
- синтаксическая омонимия (syntactic homonymy);
- лексическая многозначность (word sense disambiguation);
- синтаксическая синонимия (syntactic synonymy);
- разрешение кореферентности (coreference resolution).

Для анализа предложения и тем более текста каждая из них в настоящее время остается не решенной. Познакомимся с ними подробнее.

1. Морфологическая омонимия – совпадение одной или нескольких грамматических форм слов, принадлежащих разным частям речи, в написании и произношении.

Пример

Я траву косил *косой*,
Дождик вдруг пошел *косой*.
Бросил я тогда косить
И на Стешу стал косить.
Ну а Стеша, ох, краса,
Как огонь ее коса!

Явление морфологической омонимии весьма негативно отражается на скорости работы программы синтаксического анализа. На длинных предложениях количество комбинаторных вариантов иногда достигает нескольких сотен, поэтому используются разного рода математические и лингвистические ухищрения, позволяющие избежать анализа всех комбинаторно возможных вариантов.

Для сравнения (в системе АОТ, о которой мы еще будем говорить): скорость морфологического анализатора составляет 6000 слов в секунду, синтаксического – 300 слов в секунду.

Существует два подхода для снятия морфологической омонимии: детерминированный и вероятностный.

Детерминированный подход (развивается с 60-х годов) основан на локальном и глобальном синтаксическом разборе, синтаксических словарях и на правилах согласования слов.

Например, в предложении *Кошка пила молоко* есть три существительных, стоящих в именительном падеже, и одно из них может быть глаголом. Как быть? Есть правило: для слова, имеющего признаки и существительного, и глагола (у нас это *пила*) мы должны найти существительное, согласованное с глаголом в роде и числе или только в числе (в случае составного подлежащего или простого подлежащего во множественном числе). Таких правил имеется целый набор.

Вероятностный подход (развивается последние 20 лет) использует статистику совместной встречаемости грамматических признаков слов в больших корпусах, омонимия в которых снята заранее. Здесь предполагается использование методов машинного обучения системы (с учителем или без).

Практически все существующие алгоритмы снятия омонимии включаются в состав синтаксического анализа, что создает трудноразрешимое противоречие, когда для успешного снятия омонимии необходимы точные результаты синтаксического анализа, для получения которых, в свою очередь, нужно предварительно снять омонимию.

Пример

Рассмотрим грамматически правильное предложение, иллюстрирующее, как омонимы могут быть использованы для создания сложных конструкций:

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.

Фразу можно перевести так: «Баффальские буйволы, запуганные (другими) баффальскими буйволами, запугивают баффальских буйволов».

Припишем каждому слову часть речи:

Buffalo.a buffalo.n Buffalo.a buffalo.n buffalo.v buffalo.v Buffalo.a buffalo.n, где

«a» – имя прилагательное, Buffalo = город в США, штат Нью-Йорк (здесь прил. баффальский по названию местности);

«n» – существительное, buffalo = буйвол;

«v» – глагол, buffalo = запугивать, озадачивать.

Желаемый результат синтаксического разбора приведен на рис. 8. Это предложение не может корректно разобрать ни один из существующих в наши дни синтаксических парсеров.

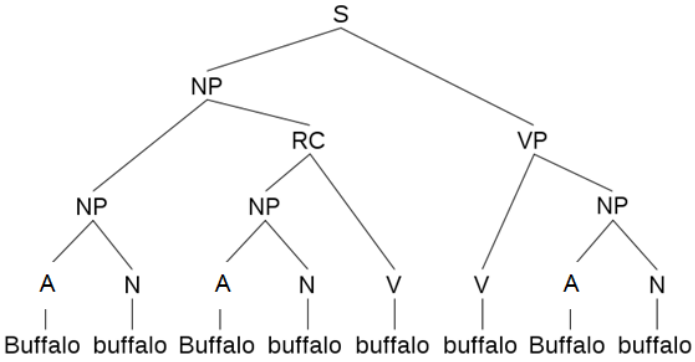


Рис. 8. Желаемый результат синтаксического разбора

2. Синтаксическая омонимия – неоднозначность, возникающая из-за неясности синтаксических связей между словами в предложении.

Пример

Преподаватель предложил прийти на зачет во вторник (*предложил во вторник* или *прийти во вторник?*).

Мать любит дочь (кто кого любит?).

Ему нужно отдать долг (*ему должны отдать* или *он должен отдать?*).

3. Лексическая омонимия и полисемия – совпадение одной или нескольких грамматических форм слов в написании и произношении; слова при этом принадлежат одной части речи, но имеют несколько различных значений.

Пример

Ключ – инструмент для открывания; *ключ* – источник воды.

Ключ подошел, дверь открылась.

Я напился из ключа.

Жизнь бьет ключом.

Пример

Bark – кора; *bark* – лаять.

The dogs bark at the tree.

В процессе работы над проблемой разрешения лексической многозначности было обнаружено большое количество трудностей, чаще всего обусловленных свойствами человеческой психологии и речи. Кроме того, значения слов сильно зависят от контекста.

Существует два подхода для снятия лексической омонимии: глубокий и поверхностный.

Глубокий подход обычно использует отнологии и предполагает наличие доступа к знаниям о мире. Несмотря на то, что при данном подходе знания о мире возможно хранить в удобном для компьютерной обработки формате, глубокий подход считается не слишком результативным на практике. Связано это с тем, что онтологии, как правило, содержат информацию о небольших областях жизни и не могут применяться к любого рода исследованиям.

При **поверхностном подходе** не ставится задача понимания текста, но производится анализ близлежащих слов. Например, рассмотрим слово *bass*, имеющее по крайней мере два значения: *рыба окунь* и *низкий голос*. Если рядом со словом *bass* в тексте присутствуют слова *sea* или *fishing*, скорее всего, что в данном случае имеет место первое значение, относящееся к рыбам. Подобные правила могут быть автоматически извлечены при использовании корпуса текстов с размеченными значениями слов. Пусть

этот подход и не покрывает по мощности предыдущий, но по эффективности на практике легко его обгоняет.

4. Синтаксическая синонимия – явление, при котором синтаксические конструкции имеют близкие значения и способны в определенных контекстах заменять друг друга.

Пример

John fell silent not knowing what to say.
John fell silent as he didn't know what to say.
John fell silent without knowing what to say.

The Byron poems.
The poems by Byron.
Byron's poems.
The poems of Byron.

5. Разрешение кореферентности – определение отношений между компонентами высказывания, в котором имена ссылаются на один и тот же объект. Благодаря кореферентности текст получается связным.

Пример

Книга лежит на столе. *Она* тяжелая.
Миша был зол на Диму, потому что *он (Дима)* украл *его (Мишин)* обед.
Миша был зол на Диму, поэтому *он (Миша)* украл *его (Димин)* обед.
Carol told *Bob* to attend the party. *They* arrived together.
If *they* are angry about the music, *the neighbors* will call the cops.

Существует несколько способ выражения референции:

- графический: *фотоконтент – фото-контент – фото контент*;
- транслитерация: *Yandex – Яндекс*;
- аббревиация: *ФМШ – физико-математическая школа – СУНЦ*;
- синонимия: *больница – госпиталь*.

Определение отношений между сущностями опирается на использование онтологий. Факты можно представлять как строки в таблице, а в столбцах размещать объекты и отношения между ними. Определение отношений между сущностями часто осуществляется одновременно с разрешением кореференций, либо с распознаванием именованных элементов.

Среди методов решения задачи распознавания именованных сущностей (Named Entity Recognition) особую популярность получили методы машинного обучения с учителем. В некоторой степени точность обучающихся систем выделения сущностей зависит от наличия дополнительных ресурсов: словарей сущностей, коллекций размеченных текстов и т. д. Главный минус машинного обучения в том, что если ошибиться с выбором

критериев, по которым происходит обучение, то полученный результат будет неудовлетворительным. Система будет часто делать ошибки, точность будет очень низкая, а усилий затрачено уже слишком много. Кроме того, инструменты для автоматической разметки русскоязычных текстов пока не очень развиты, а существующие не всегда легко доступны. Следует учитывать, что обучающая выборка должна быть достаточно объемная, размечена верно, единообразно и полностью. А это достаточно трудоемкий процесс.

Другим перспективным направлением в решении задачи распознавания именованных сущностей является использование онтологий и словарей. В частности, хорошим ресурсом имен известных людей и названий компаний является Википедия, в которой, помимо прочего, содержатся возможные варианты написания сущностей. Неоспоримым преимуществом данного направления является то, что словарные методы исключают ложные срабатывания и дают максимальную точность.

В настоящее время активно ведется работа над созданием гибридных методов, опирающихся на машинное обучение с учителем и одновременно использующих Википедию в качестве источника дополнительной информации о сущностях.

Список литературы

1. Hudson R. Language Networks: The new word grammar. Oxford University Press, 2007.
2. Кобзарева Т. Ю. В поисках синтаксической структуры: автоматический анализ русского предложения с опорой на сегментацию. М. : РГГУ, 2015. 371 с.
3. Батура Т. В., Мурзин Ф. А. Машинно-ориентированные логические методы отображения семантики текста на естественном языке : моногр. / Институт систем информатики им. А. П. Ершова СО РАН. Новосибирск : Изд. НГТУ, 2008. 248 с.
4. Temperley D. An Introduction to the Link Grammar Parser, 2014. URL: http://www.abisource.com/projects/link-grammar/dict/introduction.html#1.

ГЛАВА 3. **ПРИНЦИПЫ ПОСТРОЕНИЯ СИНТАКСИЧЕСКИХ АНАЛИЗАТОРОВ**

Морфологический и синтаксический анализ – важные этапы автоматической обработки перед поиском, переводом или другими операциями с текстом. Морфологический анализ был подробно рассмотрен в разделе 1. Теперь остановимся на синтаксическом анализе. Рассмотрим работу синтаксических анализаторов для русского (АОТ) и английского (LINK) языков.

3.1. Проект «Автоматическая Обработка Текста» (АОТ)

Более детальная информация имеется на сайте проекта [1]. В проекте «Автоматическая обработка текста» обратим внимание на этапы фрагментационного и синтаксического анализа.

На этапе фрагментационного анализа

- осуществляется определение типа фрагмента;
- применяются правила снятия омонимии при определении типа фрагмента;
- применяются правила, устанавливающие иерархию.

Синтаксический анализ основан на последовательном применении синтаксических правил для построения синтаксических групп и соблюдении принципа проективности для синтаксических групп. Принцип проективности был сформулирован в предыдущем разделе. Суть его в следующем: из того, что две синтаксические группы пересекаются, следует, что одна лежит в другой. При графическом способе изображения деревьев зависимостей слабая проективность равносильна возможности провести все стрелки так, чтобы никакие две из них не пересекались.

Согласно определению из раздела 1, задача фрагментационного анализа состоит в том, чтобы выделить в предложении фрагменты (синтаксические группы) и установить иерархию на множестве этих единств, не используя семантической информации и информации о модели управления. Иерархия отражает тот факт, что в предложении некоторые фрагменты синтаксически зависимы от других.

На вход фрагментационного анализа поступает текст, разбитый на предложения. Каждое предложение разбито на слова и знаки препинания. Каждому слову приписана морфологическая информация (все возможные пары <грамматическая характеристика, лемма>, которым удовлетворяет слово).

На выходе имеется текст, состоящий из предложений, разбитых на линейно неразрывные фрагменты. На множестве фрагментов установлена

иерархия, т. е. про каждый фрагмент известно, какие фрагменты в него непосредственно вложены и в какие он непосредственно вложен. Каждому фрагменту приписано множество типов.

1. Определение типа фрагмента

Типом фрагмента может быть ровно одно значение из списка: глагол в личной форме, краткое причастие, краткое прилагательное, предикативное слово, причастие, деепричастие, инфинитив, вводное слово. Если ни одно из предыдущих значений не подходит, в качестве типа фрагмента выбирается пустое значение.

Начиная с первого значения из списка, по порядку проверяется, есть ли в данном фрагменте слово этой части речи. Если такое слово найдено и у него нет омонимов других частей речи, то дальнейшие поиски прекращаются, и тип фрагмента – это значение, на котором сделана остановка. Если для данного значения из списка не нашлось неомонимичных (с точностью до части речи) подходящих слов, но есть омонимичные, тогда для фрагмента не устанавливается однозначно тип, а постулируется несколько вариантов, которые либо уничтожатся на уровне семантики, либо останутся в выходной структуре.

Пример

на этот раз она не права

Для этого фрагмента есть два варианта. Тип фрагмента – «краткое прилагательное» (*права* – ж. р., ед. ч. от *правый*). Тип фрагмента – «пусто» (*права* – и. п./в. п., мн. ч.; р. п., ед. ч. от *право*).

Пример

мои права забрали в милиции

Для этого фрагмента тип определяется однозначно, т. к. *забрали* – неомонимичный глагол в личной форме. Глагол в личной форме стоит в списке на первом месте, дальнейшие поиски возможных вершин фрагмента не ведутся.

2. Правила снятия омонимии при определении типа фрагмента

В том случае, если тип фрагмента не определяется однозначно или если имеется несколько слов, которые могли бы быть вершиной фрагмента, в системе АОТ применяется определенный набор правил для снятия омонимии. Для наглядности рассмотрим несколько примеров.

Первое правило

Если есть слово, один из омонимов которого – краткое прилагательное или краткое причастие, и во всем предложении нет существительного или местоимения в именительном падеже того же числа и рода (если число единственное), то этот омоним уничтожается.

Пример

Права он получил только с пятой попытки.

Так как во фрагменте нет существительного женского рода в единственном числе и в именительном падеже, значит, у слова *права* уничтожается омоним – краткое прилагательное женского рода единственного числа от *правый*.

Второе правило

Если во фрагменте есть неомонимичная предикация (глагол в личной форме, краткое прилагательное, краткое причастие, предикативное слово, причастие или деепричастие), то во всех остальных словах данного фрагмента уничтожаются омонимы этих частей речи.

Пример

Мыла на кухне она не нашла.

Так как *нашла* – неомонимичный глагол, то у слова *мыла* уничтожается омоним – прошедшее время, единственное число, женский род от глагола *мыть*.

3. Правила, устанавливающие иерархию – это правила, вкладывающие один фрагмент в другой или объединяющие фрагменты на основании стандартной информации о структуре фрагмента и сведений об отдельных словах. Рассмотрим пример правила, применяемого к фрагментам с типом «причастие».

Пример

1. ***Написанная в спешке*** *программа выполнила недопустимую операцию.*

Проверяем, не стоит ли причастие – вершина фрагмента – перед определяемым словом: осуществляем поиск существительного, совпадающего с причастием по роду, числу и падежу. Если такое существительное найдено, то данный фрагмент (причастный оборот) вкладывается в соседний правый, и осуществляется выход из правила.

2. *Программа,* ***написанная в спешке****, выполнила недопустимую операцию.*

Поиск в соседнем левом фрагменте существительного или местоимения, совпадающего с причастием в роде числе и падеже. Если такое слово найдено, то данный фрагмент (причастный оборот) вкладывается в соседний левый, и осуществляется выход из правила.

Любое простое предложение может быть «разорвано» причастными или деепричастными оборотами или придаточными предложениями, которые, в свою очередь, тоже могут быть «разбиты» другими оборотами и придаточными. Иногда части цельного высказывания находятся на значительном расстоянии друг от друга, а глубина вложения теоретически не ограничена. Поэтому для определения семантики высказывания особенно

важен корректно проведенный процесс сбора воедино фрагментов предложения.

На этапе синтаксического анализа осуществляется построение синтаксических групп при помощи последовательного применения синтаксических правил (около 40 правил, все правила упорядочены). Причем соблюдается принцип проективности для синтаксических групп.

Пример

рубить дрова; есть кашу; читать книгу

Правило для построения групп: глагол + прямое дополнение (ПРЯМ_ДОП).

Рассматриваются цепочки: глагол + существительное в винительном падеже.

Главная группа: глагольная группа.

Пример

радостно сообщил; тихо и смирно сидит; хорошо знаю

Правило для построения групп: наречие + глагол (НАРЕЧ_ГЛАГОЛ).

Рассматриваются цепочки: одиночное наречие + одиночный глагол; группа наречий + одиночный глагол; одиночное наречие + группа инфинитивов.

Главная группа: глагольная группа.

Пример

краше тебя; уютнее твоего дома

Правило для построения отсравнительной группы (ОТСРАВН).

Рассматриваются цепочки: две группы, у первой группы главное слово – сравнительное прилагательное, у второй группы главное слово – существительное в родительном падеже.

Главная группа: группа прилагательного.

3.2. Синтаксический анализатор LINK

Синтаксический анализатор Link Grammar Parser [2] был разработан в 1990-х годах в университете Корнеги-Мелона. Данный подход отличается от классической теории синтаксиса. Система приписывает предложению синтаксическую структуру, которая состоит из множества помеченных связей (коннекторов), соединяющих пары слов. Link Grammar Parser использует информацию о типах связей между словами.

Анализатор имеет словари, включающие около 60000 словарных форм. Он позволяет анализировать большое число синтаксических конструкций, включая многочисленные редкие выражения и идиомы. Link Grammar Parser довольно устойчив, он может пропустить часть предложения, которая ему непонятна, и определить некоторую структуру оставшейся части предложения. Анализатор способен работать с неизвестной лексикой и

делать разумные предположения (на основе контекста и написания) о синтаксической категории неизвестных слов.

Анализ в системе проходит в два этапа.

1. Построение множества синтаксических представлений одного предложения. На этом этапе рассматриваются все варианты связей между словами, и выбираются среди них те, которые удовлетворяют
   - **критерию проективности** (связи не должны пересекаться);
   - **критерию минимальной связности** (получившийся граф должен содержать наименьшее число компонент связности. Компонента связности графа – некоторое множество вершин графа такое, что для любых двух вершин из этого множества существует путь из одной в другую, и не существует пути из вершины этого множества в вершину не из этого множества.).
2. Постобработка предназначена для работы с уже построенными альтернативными структурами предложения.

Получаемые диаграммы по сути являются аналогом деревьев подчинения. В деревьях подчинения от главного слова в предложении можно задать вопрос к второстепенному. Таким образом, слова выстраиваются в древовидную структуру. Синтаксический анализатор может выдать две или более схемы разбора одного и того же предложения. Это явление называется синтаксической синонимией.

Главной причиной, по которой анализатор называют семантической системой, можно считать уникальный по полноте набор связей (около 100 основных, причем некоторые из них имеют 3–4 варианта). В некоторых случаях тщательная работа над разными контекстами привела авторов системы к переходу к почти семантическим классификациям, построенным исключительно на синтаксических принципах. Так, выделяются следующие классы английских наречий: ситуационные наречия, которые относятся ко всему предложению в целом (clausal adverb); наречия времени (time adverbs); вводные наречия, которые стоят в начале предложения и отделены запятой (openers); наречия, модифицирующие прилагательные и т. д.

Из достоинств системы нужно отметить, что организация самой процедуры нахождения вариантов синтаксического представления очень эффективна. Построение идет не сверху вниз (top-down) и не снизу вверх (bottom-up), а все гипотезы отношений рассматриваются параллельно: сначала строятся все возможные связи по словарным формулам, а потом выделяются возможные подмножества этих связей. Это, во-первых, приводит к алгоритмической непрозрачности системы, поскольку очень трудно проследить за всеми отношениями сразу, а во-вторых – не к линейной зависимости скорости алгоритма от количества слов, а к экспоненциальной, поскольку множество всех вариантов синтаксических структур на предложе-

нии из N слов в худшем случае равномощно множеству всех основных деревьев полного графа с N вершинами.

Последняя особенность алгоритма заставляет разработчиков использовать таймер для того, чтобы вовремя останавливать процедуру, которая работает слишком долго. Однако все эти недостатки с лихвой компенсируются лингвистической прозрачностью системы, в которой с одинаковой легкостью прописываются валентности слова, причем порядок сбора валентностей внутри алгоритма принципиально не задается, связи строятся как бы параллельно, что полностью соответствует нашей языковой интуиции.

Правила соединения слов описаны в наборе словарей. В таблице 1 приведены примеры различных типов словарей, с которыми работает анализатор.

Таблица 1

Примеры типов словарей, с которыми работает Link Grammar Parser

words.n.1	исчисляемые существительные в единственном числе (*book*)
words.n.2.s	существительные во множественном числе, оканчивающиеся на «s» (*books*)
words.n.2.x	существительные во множественном числе, не оканчивающиеся на «s» (*man – men*)
words.n.3	неисчисляемые существительные (*air*)
words.n.4	существительные, которые могут быть исчисляемыми или неисчисляемыми (*tea, coffee*)

Для каждого слова в словаре записывается, какими коннекторами оно может быть связано с другими словами предложения. Коннектор состоит из имени типа связи, в которую может вступать рассматриваемая единица анализа. Например, пометка S соответствует связи между субъектом и предикатом, O – между объектом и предикатом. Только основных, наиболее важных связей имеется более 100. Для обозначения направления связи справа к коннектору присоединяется знак «+», слева – знак «–» . Левонаправленный и правонаправленный коннекторы одного типа образуют связь (link).

Например, если слову W1 приписан коннектор A+, а слову W2 – коннектор A–, то в синтаксической структуре предложения, состоящего из двух слов W1 W2, будет проведена связь A между словами W1 и W2. Предложение же W2 W1 не получит никакой интерпретации, поскольку W2 приписан коннектор A–, который образует связь только влево, а слову W1 приписан A+, который образует связь только вправо (рис. 9).

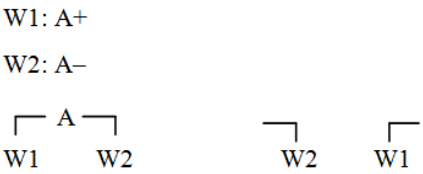


Рис. 9. Приписывание коннекторов словам в словаре

Одному слову может быть приписана формула коннекторов, составленная с помощью определенных связок.

& – несимметричная конъюнкция. Например, если слову W приписана формула A+ & B+ (обозначается «W: A+ & B+»), то некоторое слово X, с которым слово W образует связь A, должно стоять раньше по тексту, чем слово Y, с которым слово W образует связь B.

or – дизъюнкция. Если W: A+ or B-, то слово W может образовывать либо связь A вправо, либо связь B влево.

{} – факультативность. Если W: A+ & {B+}, то после того, как слово W образовало правую связь A, оно может образовывать или не образовывать связь B.

@ – неограниченность означает, что связь может строиться неограниченное число раз.

Пример

На рис. 10 изображен пример разбора предложения *I am sitting on a chair* анализатором Link Grammar Parser.

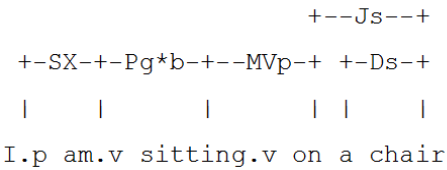


Рис. 10. Результат разбора предложения анализатором Link Grammar Parser

Использованы следующие обозначения:

.p – местоимение (pronoun);

.v – глагол (verb);

.n – существительное (noun).

В разборе предложения участвуют связи, перечисленные в таблице 2.

Таблица 2

Типы связей Link Grammar Parser

SX	соединеняет местоимение «I» с формами глагола «to be»
Pg*b	соединяет форму глагола «to be» с предлогом, прилагательным или причастием наст. вр. (**Pg**) или прош. вр. (**Pv**)
MV	соединяет глаголы и прилагательные с модифицирующими фразами, такими как именные группы, предложные группы, наречия, временные выражения и т. д. (**MVp** соединяет предлоги с глаголами; **MVa** соединяет наречия с глаголами и др.)
Js	соединяет предлог с его дополнением
D	соединяет определитель («a», «the», «some», «this», «each», «many», «much») с существительным (**Ds** соединяет сущ. с артиклем, **Dmc** – с «many», **Dmu** – с «much» и др.)

Отметим также недостатки Link Grammar Parser.

1. Практическое тестирование системы показывает, что при анализе сложных предложений, длина которых превышает 25–30 слов, возможен комбинаторный взрыв. В этом случае результатом работы анализатора становится «панический» граф, как правило, случайный вариант синтаксической структуры, с лингвистической точки зрения неадекватной.

2. Применение описанных выше идей затруднено для флективных языков типа русского, ввиду значительно возрастающего объема словарей, которые возникают в силу морфологической развитости флективных языков. Каждая морфологическая форма должна описываться отдельной формулой, где нижний индекс входящего в нее коннектора должен обеспечивать процедуру согласования. Это приводит к усложнению набора коннекторов и к увеличению их количества.

3.3. Морфологический и синтаксический анализ в поисковых системах

Рассмотрим для начала устройство поисковой системы на примере системы Яндекс, для того чтобы выяснить, какая роль отводится морфологическому и синтаксическому анализу.

Поисковая система – упорядоченная совокупность документов, предназначенная для хранения и поиска информации – текстов (документов) или данных (фактов).

Ежедневно Яндекс обрабатывает более 100 млн. запросов [3]. Такая нагрузка требует от поисковой системы высокой отказоустойчивости и скорости ответа. Данные требования определяют архитектуру поиска. Основными компонентами поисковой системы являются: базовый поиск и метапоиск (рис. 11).

Рис. 11. Общий порядок обработки запроса

В основе поиска лежит индекс. **Индекс** – база данных, содержащая сведения о словах и их позициях в документах (на страницах), представленных в Интернете. Ясно, что объем индекса огромный. Актуальность базы обеспечивается поисковым роботом, который обходит и выкачивает ресурсы Интернета по предопределенному маршруту с заданной периодичностью. Выкачанные документы разбиваются на непересекающиеся части и очищаются от разметки. Для каждой полученной группы документов по очищенному тексту строится соответствующая часть индекса.

Индексация страниц может включать в себя:

1) получение документов (страниц) из web по указанным ссылкам, их начальную обработку (морфологический и синтаксический анализ), накопление и сохранение для каждого документа таблиц частот ключевых слов и пр.;
2) построение ссылочных метрик (типа PageRank);
3) построение прямого индекса (**прямой индекс** – это таблица, в которой по ID документа можно получить список слов и всю информацию об их вхождениях в документ);
4) добавление новой информации к существующему индексу;
5) сортировку по полученным метрикам (например, PageRank) и построение обратного индекса (**обратный индекс** – это таблица, в которой каждому слову сопоставлен отсортированный по релевантности и ссылочной метрике список документов, содержащих это слово).

Базовый поиск – это программа, обеспечивающая поиск по своей части индекса. Поиск выполняется параллельно по каждой части. За счет этого сокращается время формирования ответа. Для повышения надежности и оптимального распределения нагрузки каждый базовый поиск представлен несколькими экземплярами.

Метапоиск – это программа, обеспечивающая

- прием и разбор (например, лингвистический) поисковых запросов;
- выбор базовых поисков и передачу им запросов;
- кэширование страниц с результатами поиска;

– агрегацию и ранжирование найденных документов.

Согласно введенным определениям общий порядок обработки запроса можно представить в следующем виде.

1. Оценка текущей загруженности поисковой системы. Пользовательский запрос отправляется на наименее загруженный сервер.
2. Анализ поискового запроса метапоиском.
3. Проверка кэша на наличие поискового ответа. Если ответ найден, процедура поиска прекращается, ответ передается пользователю. Если ответ отсутствует, запрос передается выбранным базовым поискам.
4. Обработка запроса базовыми поисками. Каждый базовый поиск ищет в своей части индекса документы, соответствующие поисковому запросу, и ранжирует найденные. Ответы от каждого базового поиска возвращаются на метапоиск.
5. Подготовка ответа метапоиском. Документы, полученные от базовых поисков, объединяются и ранжируются.
6. Передача сформированного ответа пользователю.

Как следует из определения, *выбор базовых поисков* обеспечивается метапоиском. Это обусловлено тем, что с текстовым запросом поиску также передается вспомогательная информация (например, географическое положение пользователя). Кроме того, сами поисковые запросы обладают различными свойствами (например, могут относиться к определенным группам запросов). Эти данные и свойства могут влиять на множество базовых поисков, которые требуется использовать для поиска документов.

Кэширование интернет-страниц. В процессе передачи информации по сети может использоваться кэширование интернет-страниц – процесс сохранения часто запрашиваемых документов на (промежуточных) прокси-серверах или машине пользователя, с целью предотвращения их постоянной загрузки с сервера-источника и уменьшения трафика. Таким образом, информация перемещается ближе к пользователю. Управление кэшированием осуществляется при помощи HTTP-заголовков.

Как вариант, кэширование веб-страниц может осуществляться с помощью системы управления сайта (CMS) для снижения нагрузки на сервер при большой посещаемости (например, кэширование часто используется в работе соцсети ВКонтакте). Кэширование может производиться как в память, так и в файловый кэш. Недостаток кэширования заключается в том, что изменения, внесенные на одном браузере, могут не сразу отражаться в другом браузере, в котором данные берутся из кэш-памяти.

В метапоиске часть ответов кэшируется и впоследствии возвращается пользователям без обращения к базовым поискам. Таким образом сокращается время ответа и уменьшается нагрузка на базовые поиски.

Ранжирование – процесс упорядочивания результатов поиска, при котором документы, наиболее подходящие запросу, расположены в начале поисковой выдачи. Формула ранжирования – функция от множества факторов. Для ранжирования результатов используется ***релевантность документа*** и ***тематический индекс цитирования*** (или ссылочная метрика, т. е. число и качество внешних ссылок).

Документ, центральный предмет или тема которого в целом соответствует смысловому содержанию информационного запроса, называется ***релевантным.*** Позиция документа тем выше, чем больше его релевантность. Критерием оценки релевантности является положение слова (в начале и в заголовке документа более значимые слова), частота упоминания его в документе, расстояние между словами и другие характеристики.

Тематический индекс цитирования (ТИЦ) показывает авторитетность ресурса относительно других, тематически близких ресурсов. Учитывается только при ранжировании и никак не влияет на процесс поиска документа. ТИЦ рассчитывается для сайта в целом, а не для каждой конкретной страницы.

Вообще, индекс цитирования – принятая в научном мире мера «значимости» трудов какого-либо ученого. Величина индекса определяется количеством ссылок на определенную публикацию автора в других источниках. Однако для действительно точного определения значимости научных трудов важно не только количество ссылок на них, но и качество этих ссылок. Так, на работу может ссылаться авторитетное академическое издание, популярная брошюра или развлекательный журнал. Значимость у таких ссылок разная.

ТИЦ определяет «авторитетность» интернет-ресурсов с учетом качественной характеристики ссылок на них с других сайтов. Эту качественную характеристику условно называют «весом» ссылки. Рассчитывается она по определенному алгоритму. Большую роль играет тематическая близость ресурса и ссылающихся на него сайтов. Само по себе количество ссылок на ресурс также влияет на значение его ТИЦ, но ТИЦ определяется не количеством ссылок, а суммой их весов.

Аналог ТИЦ в ***поисковой системе Google – это величина PageRank***, позволяющая тоже определить порядок выдачи страниц пользователю в результатах поиска.

Основной принцип такой же: «важность» веб-страницы определяется не только большим количеством ссылок на страницу, но и весом каждой из этих ссылок, т. е. вершина, ссылающаяся на другую вершину с большим весом, сама получает большой вес. В настоящее время используемый вариант PageRank учитывает только ссылки с тематически связанных страниц.

Вариантов алгоритма PageRank довольно много. Классический алгоритм ссылочного ранжирования PageRank может быть описан следующим образом:

$$x_i = \alpha \sum_{j=1}^{N} \frac{x_j}{L(j)} + \frac{1-\alpha}{N}, \text{ где}$$

$L(j)$ – количество исходящих ссылок на странице j;

x_j – PageRank j-й страницы, ссылающейся на рассматриваемую страницу i;

N – общее число ссылок на i-й странице;

α – коэффициент затухания (означает вероятность того, что пользователь, зашедший на страницу, перейдет по одной из ссылок, содержащейся на этой странице, а не остановится на текущей); в классической формуле обычно он равен 0,85.

Для удобной оценки качества поиска пользователем используются сниппеты. ***Сниппет*** – информация о документе, который отображается в результатах поиска. Сниппет состоит из заголовка и аннотации документа, поэтому требуется дополнительно хранить тексты страниц для построения аннотаций в процессе поиска. Сниппет позволяет пользователю получить представление о документе или даже искомую информацию, не открывая сам документ.

Разбор поисковых запросов. В поисковой системе Яндекс морфологический анализатор применяется для индексации всех слов текста ресурса и слов запроса пользователя. Синтаксический анализ используется для внутреннего преобразования запроса, дальнейшего отбора документов. Например, из запроса пользователя исключаются некоторые заведомо неинформативные слова, такие как некоторые частицы, предлоги, местоимения, союзы.

Операцию, выполняемую модулем морфологического анализа, можно представить как отображение $W \to L$, где

W – множество всех терминов;

L – множество всех лемм.

При этом количество лемм меньше мощности множества всех терминов $|L| < |W|$. Реализуя данное преобразование, разработчики поисковых систем пытаются

- увеличить полноту поиска,
- улучшить точность поиска.

Увеличение полноты поиска. Так как отбираются документы, которые содержат все формы слова, то в результат поиска попадают не только документы со словом в совпадающей с запросом форме, но и другие документы, содержащие различные формы данного слова. Использование модуля морфологического анализа позволяет увеличить не только полноту, но и адекватность результата информационного поиска запросу. Данный результат можно объяснить тем, что в случае отсутствия морфологической обработки часто встречается ситуация, когда в выборку попадают документы, не релевантные запросу, но содержащие совпадающие формы, в то время как в релевантных документах данные слова употребляются в другой форме.

Улучшение точности поиска. Если использовать статистические алгоритмы поиска и отбора в результат поиска нескольких документов, которые получили наибольший вес, становится важным получение частотных характеристик документов. При этом использование частот лемм вместо частот слов может позволить получить больший вес для релевантных документов и тем самым поместить их во множество отобранных.

Сложность естественного языка приводит к тому, что ни один из описанных подходов не может «охватить» его целиком. Для русского языка известно более 1000 правил словообразования с множеством исключений, что делает создание полного набора правил крайне сложным. Постоянное развитие языков и большой размер словарей делает невозможным использование только декларативного подхода. Поэтому в большинстве систем используют словарь и набор правил для обработки слов, не содержащихся в словаре.

Следует заметить, что на сегодняшний день нет систем, реализовавших морфологию полностью на все 100 процентов. В основном при реализации используется урезанная версия морфологического анализа – ограниченный словарь и упрощенный набор правил. Разумеется, такие системы используют морфологию с определенной погрешностью, которой, в целом, можно пренебречь.

Использование морфологического анализа в автоматических системах имеет свои особенности. Во-первых, снижается трудоемкость размещения информации. Одному запросу или ключевой фразе пользователя соответствует целый набор слов и словосочетаний. Фактически упрощается процесс размещения информации для всех пользователей. Тем самым снижаются затраты. Однако среди пользователей автоматической системы могут быть как надежные, так и неблагонадежные, использующих любые средства и методы для продвижения своей информации. Как правило, активность второй группы пользователей на порядок выше первой. Поэтому снижение себестоимости размещения повышает риск увеличения ложной и неадекватной информации.

Во-вторых, возникают проблемы с точностью обработки запросов. Поскольку реализация полной версии морфологического анализа невозможна, обязательно возникнут несоответствия и «непопадания» в имеющиеся в системе правила.

В-третьих, усложняется формирование релевантного ответа. При формировании неточного совпадения с учетом морфологии построение релевантного списка результатов – очень трудная задача. Как оценить вес неточно совпадающих фраз и слов, определить, какие более значимы, какие менее? Все это требует применения новых правил и методов анализа.

Чаще всего пользователя интересует нахождение тех документов, в которых описываются конкретные свойства запрашиваемого предмета или явления, то есть так называемый поиск по смыслу. Запрос при этом формулируется в виде сложного словосочетания или целого предложения. Становится ясно, что здесь обычный поиск по ключевым словам не будет удовлетворительным. Рассмотрим пример.

Пример

Имеется запрос: *К чему приводит гипертония?*

Документ 1: *Гипертония приводит к нарушению кровоснабжения тканей.*

Документ 2: *Хроническое недосыпание приводит к гипертонии.*

Если учесть общепринятые в лингвистике соглашения и считать, что «результатив» – это результат воздействия чего-либо, а «каузатив» – источник воздействия, то получим таблицу 3.

Таблица 3

Определение релевантности документа

	Результатив	**Каузатив**
Запрос	к чему	гипертония
Документ 1	к нарушению	гипертония
Документ 2	к гипертонии	недосыпание

Проблема в том, что Документ 2 совсем не близок по смыслу запросу, хотя по словам достигнуто полное соответствие. Поисковая система, снабженная только модулем морфологического анализа, посчитает релевантными оба документа, а также много других документов, содержащих слова *гипертония* и *приводит*, и, возможно, не отвечающих смыслу запроса. Естественно, такая ситуация не является удовлетворительной, поэтому при создании поисковых систем не получается обойтись одним лишь модулем морфологического анализа. В структуру поисковой системы требуется внедрение алгоритмов синтаксического анализа запросов и предложений из найденных текстов.

Список литературы

1. Автоматическая обработка текста. URL: http://www.aot.ru.
2. Link Grammar Parser. URL: http://www.abisource.com/projects/link-grammar/.
3. Технологии Яндекса. URL: https://tech.yandex.ru.

ГЛАВА 4. **ФОРМАЛЬНЫЕ ГРАММАТИКИ**

4.1. Способы задания формальных языков

1. Перечисляющие грамматики. Рассмотрим задачу задания языка. Если определяемый язык L состоит из небольшого числа цепочек, то самый очевидный способ описания языка – составить список всех цепочек из L.

Однако многие языки, например, языки программирования невозможно или нежелательно задавать исчерпывающим перечислением входящих в них цепочек. Поэтому, как правило, используются другие способы определения языка, причем такие, которые позволяют описанию языка быть обозримым (заведомо конечным), хотя описываемый язык может быть и бесконечным.

2. Порождающие грамматики (генеративные грамматики) задают правила, с помощью которых можно построить любую цепочку («предложение») языка. Генеративная грамматика – лингвистическая теория, рассматривающая грамматику как систему правил, которая генерируется в точности из тех комбинаций слов, которые составляют (формируют) грамматические предложения на данном языке. Одно из преимуществ определения языка с помощью порождающих грамматик состоит в том, что грамматика придает цепочкам языка определенную структуру, которая в большинстве случаев может отражать смысл предложения. Примером такой порождающей грамматики является рассматриваемая в второй главе грамматика составляющих, которая используется для представления синтаксической структуры предложения на естественном языке.

3. Распознающие грамматики (аналитические грамматики) позволяют по данному слову определить, входит оно в язык или нет. Этот метод определения языка связан со способом задания множества с помощью характеристического свойства (предиката) и состоит в использовании частичного алгоритма (предписания производить некоторые действия), который для произвольной входной цепочки остановится и ответит «да» после конечного числа шагов, если эта цепочка принадлежит языку. Схематизированные устройства, используемые для представления таких алгоритмов, называются распознавателями. Примерами распознавателей являются конечные автоматы, автоматы с магазинной памятью и машины Тьюринга.

4.2. Порождающие грамматики

В 1956 году Н. Хомский предложил иерархию формальных языков, порождаемых грамматиками, по виду их правил. Он выделил четыре типа

грамматик (рис. 12): регулярные (Р), контекстно-свободные (КС), контекстно-зависимые (КЗ) и неограниченные (Н) (с фразовой структурой).

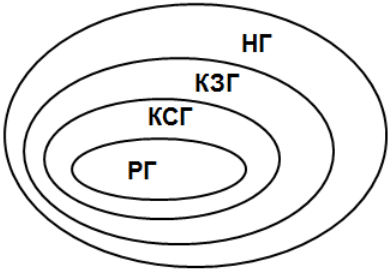


Рис. 12. Иерархия Хомского формальных языков

Каждый следующий тип грамматик формальных языков является расширением предыдущего (под «расширением» здесь понимается увеличение сложности языка).

Регулярные грамматики (автоматные грамматики) – самые простые формальные грамматики. Они являются частным случаем КС-грамматик с определенными ограничениями. В зависимости от ограничивающих условий выделяют праволинейные и леволинейные грамматики. Регулярные грамматики широко применяются в качестве шаблонов для текстового поиска, разбивки и подстановки, в том числе в лексическом анализе.

КС-грамматики содержат так называемые *e*-правила – правила виде $A \to e$, которые означают переход к пустой цепочке, т. е. в некотором смысле «остановку». КС-грамматики иногда называют бесконтекстными, т. к. они не учитывают «контекст». КС-грамматики широко применяются для описания синтаксиса компьютерных языков.

КЗ-грамматики не содержат *e*-правил. Запрещение *e*-правил в КЗ-грамматике вызвано желанием гарантировать рекурсивность порождаемого грамматикой языка, иначе говоря, желанием иметь алгоритм, который для произвольной КЗ-грамматики G и входной цепочки α определял бы, принадлежит ли эта цепочка языку. КЗ-грамматики иногда называют неукорачивающимися из-за требования на внешний вид правила ($\alpha \to \beta$, где длина α не больше длины β), т. к. по ходу повествования «контекст расширяется». КЗ-грамматики могут использоваться при анализе текстов на естественных языках, однако при построении компиляторов практически не используются в силу своей сложности.

Неограниченные грамматики (грамматики с фразовой структурой) не имеют практического применения в силу своей сложности.

В грамматике, порождающей язык L, используются два непересекающихся алфавита:

- **терминальных символов** (из них образуются цепочки порождаемого языка L);
- **нетерминальных символов** (используются при порождении языка L как вспомогательные).

Основу грамматики составляет конечное множество **порождающих правил** (или **правил вывода**), которые могут использоваться в процессе получения цепочек языка. Каждое такое правило – это пара, состоящая из заменяемой и заменяющей цепочек в алфавите терминальных и нетерминальных символов. Причем заменяющей может быть любая цепочка, а заменяемая должна содержать хотя бы один нетерминальный символ. Существование вывода для некоторого слова является критерием его принадлежности к языку, порождаемому данной грамматикой.

Подробное изложение теории формальных языков и грамматик с определениями, утверждениями и примерами можно найти в [1]. Приведем определения лишь некоторых понятий.

Алфавитом называется непустое конечное множество, элементы которого называются **символами** (а также **буквами** и **знаками**).

Если написать последовательность символов из некоторого алфавита Σ, располагая их один за другим, то получается **цепочка** (**слово** или **строка**) символов в алфавите Σ.

Число символов в цепочке α называется ее **длиной** и обозначается через $|\alpha|$.

Пустая цепочка (цепочка нулевой длины) не содержит ни одного символа и обозначается через e.

Если α и β – две цепочки, то цепочка $\omega = \alpha\beta$ называется **конкатенацией** (или **сцеплением**) α и β. Для любого символа a и целого $k(k \geq 0)$ через a^k обозначается k-кратная его конкатенация, т. е. $a^0 = e, a^1 = a$ и $a^{k+1} = a^k a$ для всех $k \geq 1$.

Для любых цепочек α, β, γ цепочка β называется **префиксом** цепочки $\beta\alpha$, **суффиксом** цепочки $\alpha\beta$ и подцепочкой цепочки $\alpha\beta\gamma$. Префикс и суффикс любой цепочки являются ее подцепочками, а пустая цепочка является префиксом, суффиксом и подцепочкой любой цепочки.

Обращением цепочки α (обозначается α^R) называется цепочка α, записанная в обратном порядке, т. е. если $\alpha = a_1 a_2 ... a_n$, где все a_i – символы, то $\alpha^R = a_n a_{n-1} ... a_1$. Кроме того, $e^R = e$.

Пример

Двоичный алфавит – множество $\{0,1\}$. Последовательность $\alpha = 000110$ является цепочкой длины 6.

Она является конкатенацией цепочек $\gamma = 0001$ и $\beta = 10$, т. е. $\alpha = \gamma\beta$.

Цепочки 0011, 000110, 1 – подцепочки цепочки α. 1^0 – пустая цепочка. $\alpha^R = 011000$; $1^3 = 111$.

Пусть Σ – некоторый алфавит. Через Σ^* обозначается множество всех цепочек в алфавите Σ, включая пустую цепочку e, а через Σ^+ – множество $\Sigma^* \setminus e$.

Формально **множество** Σ^* **цепочек** в алфавите Σ определяется по следующим правилам:

1) $e \in \Sigma^*$;

2) если $\alpha \in \Sigma^*$ и $a \in \Sigma$, то $\alpha a \in \Sigma^*$;

3) $\alpha \in \Sigma^*$ тогда и только тогда, когда α является цепочкой в алфавите Σ в силу 1) или 2).

Множество Σ^* иногда называют **замыканием Клини**.

Согласно определению, Σ^* – это минимальное надмножество множества Σ, которое содержит e и замкнуто относительно конкатенации.

Пример

Для множества символов получаем множество цепочек:

$$\{'a','b','c'\}^* = \{e,'a','b','c','aa','ab','ac','ba','bb','bc','ca','cb','cc','aaa',\ldots\}.$$

Произвольное множество L цепочек в Σ, т. е. $L \subseteq \Sigma^*$, называется **языком** в алфавите Σ.

Так как язык – это множество, то операции объединения, пересечения, нахождения разности и дополнения применимы и к языкам. Среди основных операций можно выделить операции объединения, конкатенации (сцепления) и итерации (замыкание Клини).

Пусть L_1 – язык в алфавите Σ_1, а L_2 – язык в алфавите Σ_2. Язык L_1L_2 называется **конкатенацией** (**сцеплением** или **произведением**) языков L_1 и L_2, если он задается множеством $\{\alpha\beta : \alpha \in L_1, \beta \in L_2\}$.

Итерация (замыкание Клини) языка L (обозначается L^*) определяется по следующим правилам:

1) $L^0 = \{e\}$;

2) $L^n = LL^{n-1}$ для $n \geq 1$;

3) $L^* = \bigcup_{n \geq 0} L^n$.

Позитивная итерация языка L (обозначается L^+) – это $L^+ = \bigcup_{n \geq 1} L^n$.

Можно заметить, что $L^+ = LL^* = L^*L$ и $L^* = L^+ \cup \{e\}$. Это следует из определений.

Порождающей грамматикой (или просто **грамматикой**) называется четверка $G = (N, \Sigma, P, S)$, где

1) N – алфавит **нетерминальных символов**;

2) Σ – не пересекающийся с N алфавит **терминальных символов**;

3) P – конечное множество **правил** (или **продукций**) вида $\alpha \to \beta$, где заменяемая цепочка $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$, заменяющая цепочка $\beta \in (N \cup \Sigma)^*$ и «$\to$» – символ, не принадлежащий ни N, ни Σ;

4) S – выделенный символ из L, называемый **начальным** (или **исходным**) символом.

Пример

Грамматикой является четверка $G = (\{A, S\}, \{0, 1\}, P, S)$, где P состоит из правил:

$S \to 0A1,$

$0A \to 00A1,$

$A \to e.$

Нетерминальные символы – A и S; терминальные – 0 и 1; S – выделенный начальный символ.

Определим теперь отношения выводимости, непосредственной выводимости и выводимости нетривиальным образом цепочек в грамматике.

Пусть $G = (N, \Sigma, P, S)$ – грамматика. Отношение **непосредственной выводимости** $\Rightarrow_G$ на множестве $(N \cup \Sigma)^*$ определяется следующим образом: если $\alpha\beta\gamma$ – цепочка из $(N \cup \Sigma)^*$ и $\beta \to \delta$ – правило из P, то $\alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$.

Транзитивное замыкание отношения непосредственной выводимости $\Rightarrow_G$ обозначается через $\varphi \Rightarrow_G^+ \psi$ (читается: ψ **выводима нетривиальным образом** из φ).

Рефлексивное и транзитивное замыкание отношения $\Rightarrow_G$ обозначается через $\varphi \Rightarrow_G^* \psi$ (читается: ψ **выводима** из φ). В данном случае рефлексивность означает выводимость цепочки из самой себя: $\varphi \Rightarrow_G^* \varphi$; условие транзитивности имеет вид: $\varphi \Rightarrow_G^* \psi$, $\psi \Rightarrow_G^* \theta$, тогда $\varphi \Rightarrow_G^* \theta$.

Цепочка α называется **выводимой** цепочкой в грамматике G, если $S \Rightarrow_G^* \alpha$.

Язык, порождаемый грамматикой G (обозначается $L(G)$), – это множество терминальных выводимых цепочек грамматики G, т. е. $L(G) = \{\alpha : \alpha \in \Sigma^*, S \Rightarrow_G^* \alpha\}$.

Другими словами, язык, порождаемый грамматикой – это множество цепочек, которые состоят только из терминальных символов и выводятся, начиная с одной особой цепочки, состоящей из одного выделенного символа, называемого начальным.

Рассмотрим теперь примеры регулярных грамматик, КС-грамматик и КЗ-грамматик.

Пример регулярной грамматики

Рассмотрим грамматику $G = (\{A, S\}, \{0, 1\}, P, S)$ из предыдущего примера и вывод $S \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 0011$. Он получается согласно правилам.

Тогда $S \Rightarrow^3 0011$, $S \Rightarrow^+ 0011$, $S \Rightarrow^* 0011$, и цепочка 0011 принадлежит языку $L(G)$. Можно показать, что грамматика порождает язык $L(G) = \{0^n 1^n : n \geq 1\}$.

Пример КС-грамматики

Пусть задана грамматика $G = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$, где P состоит из правил

$$E \to E + T \mid T,$$
$$T \to T * F \mid F,$$
$$F \to (E) \mid a.$$

Символ «|» означает «или»; используется для сокращенной записи в одну строку двух правил.

В этой грамматике можно получить следующий вывод:

$$\begin{aligned} E &\Rightarrow E+T \\ &\Rightarrow T+T \\ &\Rightarrow F+T \\ &\Rightarrow a+T \\ &\Rightarrow a+T*F \\ &\Rightarrow a+F*F \\ &\Rightarrow a+a*F \\ &\Rightarrow a+a*a. \end{aligned}$$

Пример КЗ-грамматики

Пусть грамматика P определяется правилами

$$\begin{aligned} &S \to aSBC, \\ &S \to abC, \\ &CB \to BC, \\ &bB \to bb, \\ &bC \to bc, \\ &cC \to cc. \end{aligned}$$

Здесь a,b,c – терминалы, A,B,C – нетерминалы, S – начальный символ. Тогда в ней возможен вывод:

$$\begin{aligned} S &\Rightarrow aSBC \\ &\Rightarrow aabCBC \\ &\Rightarrow aabBCC \\ &\Rightarrow aabbCC \\ &\Rightarrow aabbcC \\ &\Rightarrow aabbcc. \end{aligned}$$

Эта грамматика порождает язык $L(G)=\{a^n b^n c^n : n \geq 1\}$.

Рассмотрим специальный класс операций над языками – регулярные операции (объединение, конкатенация, итерация). Регулярные множества получаются из элементарных языков в результате применения конечного числа регулярных операций.

Пусть Σ – некоторый конечный алфавит. **Регулярное множество** в алфавите Σ определяется следующими рекурсивными свойствами:

1) $\varnothing$ – регулярное множество;

2) $\{e\}$ – регулярное множество;

3) $\{a\}$ – регулярное множество $\forall a \in \Sigma$;

4) если P, Q – регулярные множества, то множества $P \cup Q$, PQ, P^* также являются регулярными;

5) других регулярных множеств в алфавите Σ нет.

Регулярные множества – это множества цепочек символов над заданным алфавитом, построенные определенным образом (с использованием операций объединения, конкатенации и итерации).

Все регулярные языки представляют собой регулярные множества. Способ их описания – регулярные выражения и недетерминированные конечные автоматы, допускающие цепочки из этих множеств.

Регулярные выражения в алфавите Σ и регулярные множества, которые они обозначают, определяются рекурсивно следующим образом:

1) $\emptyset$ – регулярное выражение, обозначающее регулярное множество $\emptyset$;

2) e – регулярное выражение, обозначающее регулярное множество $\{e\}$;

3) если $a \in \Sigma$, то a – регулярное выражение, обозначающее регулярное множество $\{a\}$;

4) если p и q – регулярные выражения, обозначающие регулярные множества P и Q соответственно, то

- объединение $(p+q)$ – регулярное выражение, обозначающее регулярное множество $P \cup Q$;
- конкатенация (pq) – регулярное выражение, обозначающее регулярное множество PQ;
- итерация $(p)^*$ – регулярное выражение, обозначающее регулярное множество P^*;

5) других регулярных выражений нет.

Каждое регулярное выражение обозначает одно и только одно регулярное множество, но для одного регулярного множества может существовать сколь угодно много регулярных выражений, обозначающих это множество.

При записи регулярных выражений используют круглые скобки, как и для обычных арифметических выражений. При отсутствии скобок операции выполняются слева направо с учетом приоритета. Приоритет для операций принят следующий: первой выполняется итерация (высший приори-

тет), затем конкатенация, потом – объединение множеств (низший приоритет).

Пример

Множество всех цепочек, составленных из нулей и единиц и оканчивающихся цепочкой 001, может быть описано регулярным выражением $(0+1)^* 011$.

Пример

Регулярное выражение $(a+b)(a+b+0+1)^*$ обозначает множество всех цепочек из $\{0, 1, a, b\}^*$, начинающихся с a или b.

Рассмотрим несколько примеров использования регулярных выражений в задачах обработки текстовой информаци на естественном языке.

Основой синтаксиса регулярных выражений в языках программирования является тот факт, что некоторые символы, встречающиеся в строке, рассматриваются не как обычные символы, а как имеющие специальное значение (так называемые метасимволы). Каждый метасимвол имеет свое значение.

Пример

Ниже приведено описание метода проверки того, что строка заканчивается на «.com», или «.org», или «.ru».

```
public static boolean test(String testString) {
    Pattern p = Pattern.compile(".+\\.(com|org|ru)");
    Matcher m = p.matcher(testString);
    return m.matches();
}
```

В строке «**.+\\.(com|org|ru)**» приняты следующие обозначения:

(…|…) – альтернатива (выбор из нескольких вариантов);

.+ – один или более экземпляров непосредственно предшествующего элемента;

\\. – экранирование точки; таким образом указывается, что идет именно точка, а не любой символ.

Пример

Требуется проверить имена двух пользователей на валидность («@_BEST» и «miha_10»). Имя первого пользователя не соответствует заданным требованиям, имя второго – соответствует.

Первый метод **manualCheck** делает проверку валидности имени пользователя обычной проверкой условий. Второй метод **checkWithRegExp** использует регулярное выражение.

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class UserNameCheck {
    public static void main(String[] args){
        System.out.println("Manual check:");
        System.out.println(manualCheck("_@BEST"));
        System.out.println(manualCheck("miha_10"));
        System.out.println("Cool check:");
        System.out.println(checkWithRegExp("_@BEST"));
        System.out.println(checkWithRegExp("miha_10"));
    }

    public static boolean manualCheck(String userNameString){
        char[] symbols = userNameString.toCharArray();
        if (symbols.length < 3 || symbols.length > 15 ) return false;
        String validationString = "abcdefghijklmnopqrstuvwxyz0123456789_";
        for (char c : symbols){
            if (validationString.indexOf(c)==-1) return false;
        }
        return true;
    }
    public static boolean checkWithRegExp(String userNameString){
        Pattern p = Pattern.compile("^[a-z0-9_]{3,15}$");
        Matcher m = p.matcher(userNameString);
        return m.matches();
    }
}
```

При необходимости изменения условия проверки будет достаточно изменить строку регулярного выражения, если используется вызов метода **checkWithRegExp**. Если использовать метод **manualCheck**, то в большинстве случаев понадобится переписывать весь метод заново, что может оказаться непростой задачей.

В строке «**^[a–z0–9_]{3,15}$**» приняты следующие обозначения:

^ – начало проверяемой строки;

$ – конец проверяемой строки;

{3, 15} – повторение от 3 до 15 раз включительно.

Выражение, записанное в примере в квадратных скобках, – это символьный класс, он участвует в записи регулярного выражения. Символьный класс определяет перечень символов, которые могут (или не могут) находиться на месте данного символа. Символьными классами […] иногда удобно заменять конструкцию выбора (…|…). Правила, определяющие состав поддерживаемых метасимволов и их интерпретацию, могут сильно отличаться внутри символьных классов и за их пределами. Так, внутри символьных классов дефис означает интервал символов. Поэтому выраже-

ние из рассмотренного примера [a–z0–9_] задает набор символов, который может состоять из маленьких латинских букв, цифр или символов подчеркивания.

При работе с текстами на естественном языке регулярные выражения позволяют удобно заменять все вхождения одного слова на другое. Это справедливо как для части слова, так и с учетом контекста. Например, когда требуется обновить цену конкретного товара в прайс-листе и т. д.

Пример

Пусть имеется текст: «Мне очень нравится Тайланд. Таиланд – это то место, куда бы я поехал. тайланд – мечты сбываются!» Необходимо найти в нем все варианты написания слова «Тайланд» и заменить их словом «Россия». Перечислить возможные варианты написания слова «Тайланд» можно по крайней мере двумя способами: «(Т|т)а(и|й)ланд» и «Таиланд|Тайланд|таиланд|тайланд». Требуемая подстановка запишется с помощью приведенного ниже регулярного выражения:

```
public class Rexep {
    public static final String TEXT = "Мне очень нравится Тайланд.
     Таиланд — это то место, куда бы я поехал. тайланд —
     мечты сбываются!";
   public static void main(String[] args){
   System.out.println(TEXT.replaceAll("[Тт]а[ий]ланд", "Россия"));
   }
  }
```

4.3. Взаимосвязь регулярных множеств, регулярных грамматик и конечных автоматов

Обычно выделяют три основных способа, с помощью которых можно задавать регулярные языки [2]:

- регулярные (праволинейные и леволинейные) грамматики;
- конечные автоматы (КА);
- регулярные множества (обозначающие их регулярные выражения).

Все три способа являются эквивалентными и в равной степени могут быть использованы для определения регулярных языков.

Связь регулярных выражений и регулярных грамматик:

- для любого регулярного языка, заданного регулярным выражением, можно построить регулярную грамматику, определяющую тот же язык;
- для любого регулярного языка, заданного регулярной грамматикой, можно получить регулярное выражение, определяющее тот же язык.

Связь регулярных выражений и конечных автоматов:

- для любого регулярного языка, заданного регулярным выражением, можно построить конечный автомат, определяющий тот же язык;
- для любого регулярного языка, заданного конечным автоматом, можно получить регулярное выражение, определяющее тот же язык.

Связь регулярных грамматик и конечных автоматов:

- на основе регулярной грамматики можно построить эквивалентный ей конечный автомат;
- для заданного конечного автомата можно построить эквивалентную ему регулярную грамматику.

Последнее утверждение является важным, поскольку регулярные грамматики используются для определения лексических конструкций языков программирования. Создав автомат на основе известной грамматики, мы получаем распознаватель для лексических конструкций данного языка. Таким образом, удается решить задачу разбора для лексических конструкций языка, заданных произвольной регулярной грамматикой. Обратное утверждение также полезно, поскольку позволяет узнать грамматику, цепочки языка которой допускает заданный автомат.

Для построения конечного автомата на основании известной грамматики и для построения грамматики на основании данного конечного автомата используются достаточно простые алгоритмы. Рассматривать их здесь мы не будем. Заметим лишь, что все языки программирования определяют нотацию записи «слева направо». В той же нотации работают и компиляторы. Поэтому такие алгоритмы используют леволинейные грамматики.

Пример

Пусть задана праволинейная регулярная грамматика $G = (\{A, S\}, \{a,b,c\}, P, S)$, где P состоит из правил:

$$S \to aS,$$
$$S \to bA,$$
$$A \to e,$$
$$A \to cA.$$

Эта грамматика описывает тот же язык $L(G) = \{a^n bc^n : n \geq 1\}$, что и регулярное выражение $a^* bc^*$.

4.4. Распознающие грамматики

Конечный автомат – это преобразователь, который позволяет сопоставить входу соответствующий выход, причем этот выход может зависеть не только от текущего входа, но и от того, что происходило раньше, от

предыстории работы конечного автомата. Даже поведение человека, а не только искусственных систем можно описать с помощью конечного автомата. Например, ваша реакция на то, что ваш сосед слушает громко музыку по ночам, будет одной после первого такого случая и совершенно другой после нескольких таких случаев. Таких предысторий может быть бесконечное число. Возникает вопрос: какой памятью должен обладать конечный автомат, чтобы вести себя по-разному для каждой предыстроии? Понятно, что хранить бесконечное число предысторий невозможно. Поэтому автомат как бы разбивает все возможные предистории на классы эквивалентности. Две истории являются эквивалентными, если они одинаково влияют на поведение автомата в дальнейшем. Класс эквивалентности, к которому автомат отнес свою текущую предысторию, называют еще внутренним состоянием автомата.

Недетерминированный конечный автомат (НКА) – это пятерка $M = (Q, \Sigma, \delta, q_0, F)$, в которой

1) Q – конечное множество *состояний*;
2) Σ – конечное множество допустимых *входных символов*;
3) δ – отображение множества $Q \times \Sigma$ во множество подмножеств Q, называемое *функцией переходов*;
4) $q_0 \in Q$ – выделенное *начальное состояние*;
5) $F \subseteq Q$ – множество *заключительных состояний*.

Если $M = (Q, \Sigma, \delta, q_0, F)$ – конечный автомат, то пара $(q, \omega) \in Q \times \Sigma^*$ называется **конфигурацией** автомата M. Конфигурация (q_0, ω) называется **начальной**, а пара (q, e), где $q \in F$, называется **заключительной** (или **допускающей**).

Такт работы автомата M – это бинарное отношение $\vdash_M$, определенное на конфигурациях. Если $\delta(q, a)$ содержит p, то $(q, a\omega) \vdash_M (p, \omega)$ для всех $\omega \in \Sigma^*$.

Отношение $\vdash_M^+$ является транзитивным замыканием отношения $\vdash_M$. Отношение $\vdash_M^*$ является рефлексивным и транзитивным замыканием отношения $\vdash_M$.

Автомат M **допускает** цепочку $\omega \in \Sigma^*$, если $(q_0, \omega) \vdash_M^* (q, e)$ для некоторого $q \in F$. **Языком, определяемым (распознаваемым, допускае-**

мым) автоматом M (обозначается $L(M)$), называется множество входных цепочек, допускаемых автоматом M:

$$L(M) = \left\{\omega : \omega \in \Sigma^* \text{ и } (q_0, \omega) \vdash_M^* (q, e) \text{ для некоторого } q \in F\right\}.$$

Конечный автомат начинает работу в состоянии q_0, считывая по одному символу входной цепочки. Считанный символ переводит автомат в новое состояние в соответствии с функцией переходов. Читая входную цепочку x и делая один такт за другим, автомат прочитает последнюю букву цепочки и окажется в каком-то состоянии q'. Если это состояние является заключительным, то говорят, что автомат допустил цепочку x.

Язык, определяемый автоматом, вводится аналогично языку, порождаемому грамматикой. Конечный автомат можно представить в виде устройства (рис. 13), состоящего из входной ленты (представлена входной цепочкой), считывающего устройства, управляющего устройства (блок управления, который содержит список правил переходов).

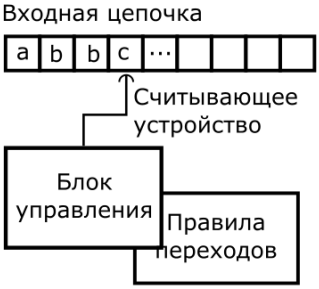


Рис. 13. Устройство конечного автомата

Входная лента – это линейная последовательность клеток, или ячеек, каждая из которых может содержать любой символ исходного алфавита. В каждый данный момент считывающее устройство читает, или, как иногда говорят, обозревает одну входную ячейку, а управляющее устройство находится в одном состоянии из конечного множества Q, т. е. имеет конечную память.

Работа конечного автомата представляет собой некоторую последовательность шагов (или тактов). Каждый такт определяется текущим *состоянием управляющего устройства* и *входным символом*, обозреваемым в данный момент считывающим устройством. Сам шаг состоит из изменения состояния управляющего устройства и сдвига считывающего устройства на одну ячейку вправо.

Детерминированный конечный автомат (ДКА) – это конечный автомат $M = (Q, \Sigma, \delta, q_0, F)$, в котором множество функций перехода $\delta(q, a)$

содержит не более одного состояния для любых состояний $q \in Q$ и входных символов $a \in \Sigma$.

Основное отличие ДКА и НКА состоит в том, что ДКА в процессе работы может находиться только в одном состоянии, а НКА – в нескольких состояниях одновременно.

Пример

Хотим получить детерминированный конечный автомат M, который распознает два стоящих рядом ноля. Такой автомат определяется как пятерка $M = (\{p, q, r\}, \{0, 1\}, \delta, p, \{r\})$, где δ задается таблицей 4.

Таблица 4

Таблица переходов ДКА

Состояние	**Вход**	
	0	1
p	$\{q\}$	$\{p\}$
q	$\{r\}$	$\{p\}$
r	$\{r\}$	$\{r\}$

Пример

Построим недетерминированный конечный автомат, допускающий цепочки в алфавите {1, 2, 3}, у которых последний символ цепочки уже появлялся в ней раньше. Иными словами, цепочка 121 допускается, а 31312 – нет.

Автомат недетерминированный, поэтому состояний может быть несколько в один момент времени.

Введем нейтральное состояние q_0, смысл которого в том, что автомат в этом состоянии не пытается ничего распознать. Введем также состояния q_1, q_2, q_3, находясь в которых автомат «предполагает», что последний символ цепочки совпадает с индексом состояния. Введем также заключительное состояние q_f. Из состояния q_f автомат никуда не переходит.

Формально автомат $M = (\{q_0, q_1, q_2, q_3, q_f\}, \{1, 2, 3\}, \delta, q_0, \{q_f\})$ определяется как пятерка, где δ задается таблицей 5.

Таблица 5

Таблица переходов НКА

Состояние	**Вход**		
	1	2	3
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_3\}$
q_1	$\{q_1, q_f\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2, q_f\}$	$\{q_2\}$

q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3, q_f\}$
q_f	Ø	Ø	Ø

Диаграмма состояний (или **граф переходов**) – размеченный ориентированный граф, вершины которого соответствуют состояниям, а дуги – переходам из одного состояния в другое.

Пусть $M = (Q, \Sigma, \delta, q_0, F)$ – конечный автомат. Если существует такой символ $a \in \Sigma$, что $q \in \delta(p, a)$, то в графе переходов есть дуга (p, q). Сама дуга (p, q) помечается списком, состоящим из таких a, что $q \in \delta(p, a)$.

Теорема (эквивалентность НКА и ДКА). Любой НКА может быть преобразован в ДКА так, чтобы их языки совпадали.

Такие автоматы называются **эквивалентными**.

Построить по НКА эквивалентный ДКА позволяет **алгоритм Томпсона**, который состоит в следующем:

1) взять все параллельные ветки НКА;

2) в них взять все состояния, в которых одновременно может находиться НКА;

3) объединить их в новое состояние ДКА.

Однако на практике подобная детерминизация не всегда возможна, поскольку количество состояний в эквивалентном ДКА в худшем случае растет экспоненциально с ростом количества состояний исходного НКА. Кроме того, конечные автоматы с выходом в общем случае не поддаются детерминизации.

В силу последних двух замечаний, несмотря на бóльшую сложность недетерминированных конечных автоматов, для задач, связанных с обработкой текста, преимущественно применяются именно НКА.

Из определения регулярных выражений следует, что любые операторы регулярных выражений можно выразить с помощью трех операций регулярных языков: объединения, конкатенации и итерации. Значит, построив автомат, на котором эти операции выполняются, получим автомат для регулярных выражений и таким образом свяжем эти два понятия. Ранее уже говорилось о связи между регулярными выражениями и конечными автоматами. Эта связь формулируется в виде теоремы Клини.

Теорема Клини. Классы регулярных множеств и автоматных языков совпадают.

Мы не будем доказывать эту теорему, рассмотрим лишь алгоритм построения НКА по регулярному выражению, т. е. проверим утверждение

теоремы только в одну сторону. Для этого воспользуемся следующим алгоритмом построения НКА по регулярному выражению.

Вход. Регулярное выражение r в алфавите Σ.

Выход. НКА M, такой что $L(M) = L(r)$.

Метод. Автомат для выражения строится композицией из автоматов, соответствующих подвыражениям. На каждом шаге построения строящийся автомат имеет в точности одно заключительное состояние. В начальное состояние нет переходов из других состояний, и нет переходов из заключительного состояния в другие.

Согласно определению регулярных множеств, некоторое множество цепочек в заданном алфавите называется регулярным тогда и только тогда, когда оно либо является одним из множеств $\emptyset$, $\{e\}$, $\{a\}$ $\forall a \in \Sigma$, либо его можно получить из этих множеств применением конечного числа операций объединения, конкатенации и итерации. Поэтому будем строить НКА по определению (рис. 14).

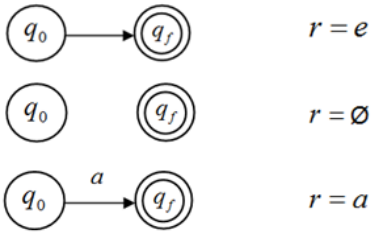


Рис. 14. Построение конечного автомата по регулярному выражению

Далее построим конечный автомат для трех перечисленных операций.

Для выражения $(s+t)$ автомат $M(s+t)$ строится так, как показано на рис. 15. Здесь q_0 вне прямоугольных рамок – новое начальное состояние, q_f вне прямоугольных рамок – новое заключительное состояние. Заметим, что имеет место переход по e из q_0 в начальные состояния автоматов $M(s)$ и $M(t)$ и переход по e из заключительных состояний $M(s)$ и $M(t)$ в q_f. Начальное и заключительное состояния автоматов $M(s)$ и $M(t)$ не являются таковыми для автомата $M(s+t)$.

Начальное состояние автомата $M(s)$ становится начальным для нового автомата $M(st)$. Заключительное состояние автомата $M(t)$ становится заключительным для нового автомата $M(st)$. Начальное состояние $M(t)$ и заключительное состояние $M(s)$ сливаются, т. е. все переходы из начального состояния $M(t)$ становятся переходами из заключительного состоя-

ния $M(s)$. В новом автомате $M(st)$ это объединенное состояние не является ни начальным, ни заключительным (рис. 16).

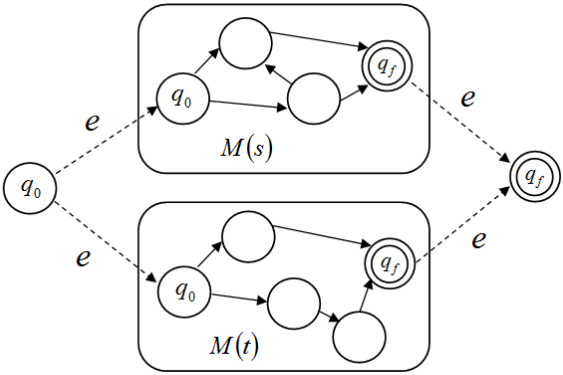


Рис. 15. Объединение двух конечных автоматов

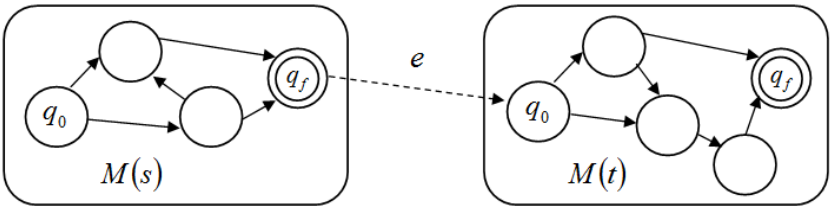


Рис. 16. Последовательное соединение двух конечных автоматов

Для выражения $(s)^*$ автомат $M(s^*)$ строится так, как показано на рис. 17. Здесь q_0 – новое начальное состояние, а q_f – новое заключительное состояние.

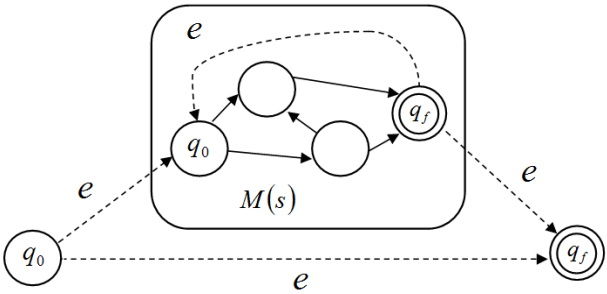


Рис. 17. Замыкание конечного автомата

Лексический и синтаксический анализаторы обычно реализуются в виде конечного автомата, определяемого регулярными выражениями. Одной из важных особенностей такого подхода к разбору строк является то, что анализ выполняется по мере считывания символов (слов) с использованием информации о текущем символе (слове) и символах (словах), прочитанных ранее. Это позволяет вести обработку данных, передающихся по некоторому последовательному каналу, непосредственно в процессе их поступления.

Существует много алгоритмов разбора, использующих генеративные грамматики; известны свойства этих грамматик, т. е. их выразительные способности, сложность обработки и т. п. Порождающие грамматики Н. Хомского успешно применяются при компиляции языков программирования.

Сам Н. Хомский прежде всего лингвист, поэтому в своих работах ориентировался на естественный язык, в первую очередь на английский. Поэтому влияние его трудов достаточно велико в англоязычной компьютерной лингвистике. Порождающие грамматики с их перечислением правил более-менее годятся для английского языка, но не очень подходят для русского языка. Ввиду морфологической развитости происходит быстрое разрастание множества правил. Причем в случае восходящего парсера это действительно неизбежно.

Однако в случае нисходящего парсера это разрастание можно замедлить, приписывая определенные характеристики нетерминальным символам. Нисходящий парсер позволяет обходиться очень небольшим количеством нетерминалов в правилах и учитывать богатую морфологию русского языка, в частности, разнообразные правила согласования по падежам, числам, родам; учитывать одушевленность, перечислимость, падежную валентность и все прочие элементы морфологической модели русского языка.

Нисходящий парсер содержит очень гибкий инструмент для лексикализации нетерминалов и расширения эффективного пространства признаков. Под лексикализацией понимается превращение отдельного элемента языка (слова или сочетания слов) в отдельное знаменательное слово либо устойчивую фразеологическую единицу, функционирующую в качестве эквивалента отдельного слова. Говоря упрощенно, каждый нетерминал после своего сопоставления кроме собственно своего имени динамически расширяется целым набором дополнительных признаков, например, включая главное слово во фразе, второе существительное в предложной фразе, инфинитив в глагольных временах, а также морфологические признаки – род, число, падеж и так далее. Весь этот набор становится доступен в правилах наряду с именем самого нетерминала.

Лексикализация может выполняться одним из двух способов. Во-первых, каждое обращение к нетерминальному символу может экспортировать определенный, заранее объявленный набор лексем и морфологических признаков. Именно этот набор расширяет нетерминальный символ. Во-вторых, в месте обращения к нетерминальному символу можно задать проверки, например, на вхождение/невхождение главного слова в именованные множества, заданные значения морфологических признаков или их согласование.

Одно из возможных решений – ввести параметры для нетерминальных символов. В таком случае грамматику можно написать вручную, а можно вывести при помощи банка деревьев. **Трибанк (treebank)** – это коллекция синтаксически размеченных предложений (т. е. графов разбора), подготовленных вручную. Такие коллекции удобно использовать для автоматического извлечения правил генерации деревьев. Наиболее распространенные проекты: Penn Treebank (https://www.cis.upenn.edu/~treebank/) и Национальный корпус русского языка (www.ruscorpora.ru).

Пример работы нисходящего синтаксического анализатора был рассмотрен в главе 2. В целом парсер последовательно осуществляет следующие операции:

- поиск замен для нетерминальных правил;
- сопоставление полученных терминалов со словами в предложении;
- отсечение некорректных гипотез.

Когда нетерминалов больше не осталось, и текущие терминальные символы сопоставлены со словами в исходном предложении, то парсер прекращает анализ, сообщая о найденной структуре.

Пример работы восходящего анализатора представлен в главе 2. Первый этап, называемый chunking, – это преобразование слов и устойчивых цепочек в новые сущности – нетерминальные символы. Каждый тип нетерминального символа кодирует в своем названии часть речи и ряд ключевых морфологических признаков. Затем алгоритм начинает сворачивать пары нетерминалов в новые нетерминалы, постепенно переходя на более высокие уровни абстракции. Работа заканчивается, когда подлежащее и сказуемое образуют особый нетерминал S, обозначающий целое предложение.

Список литературы

1. Касьянов В. Н. Лекции по теории формальных языков, автоматов и сложности вычислений : учеб. пособ. Новосибирск : НГУ, 1995. 112 с.
2. Молчанов А. Ю. Системное программное обеспечение : учебник для вузов. СПб. : Питер, 2010. 400 с.
3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М. : Мир, 1979. 536 с.
4. Турдаков Д. Ю. Основы обработки текстов. МГУ и ВШЭ, 2015. URL: http://tpc.at.ispras.ru/.

ГЛАВА 5. **МЕТОДЫ ТЕОРЕТИЧЕСКОГО ИССЛЕДОВАНИЯ СЕМАНТИКИ ТЕКСТОВ**

5.1. Исследование семантики в рамках теории «Смысл ⇔ Текст»

В прошлом разделе были рассмотрены формальные грамматики Н. Хомского и их классификация. Хомский – основатель порождающих грамматик – считал необходимым построение универсальной грамматики для всех языков мира. Семантику Хомский не рассматривает. Его языковая модель не преобразует смыслы в тексты, а порождает тексты по определенным правилам; интерпретация же приписывается этим текстам впоследствии.

И. А. Мельчук считал, что нужно построить формальную модель, наиболее полно охватывающую все множество грамматических явлений конкретного языка, в частности, русского. При этом считается, что описание процесса интерпретации текста (определение смысла) может быть получено на основе описания процесса построения текста. В этом заключается основная идея модели «Смысл ⇔ Текст» [1]. При создании этой модели И. А. Мельчук ввел понятие лексической функции и толково-комбинаторного словаря, определив через них понятие языка.

Лексическая функция – функция, аргументами которой являются некоторые слова или словосочетания данного языка, а значениями – множества слов и словосочетаний этого же языка. При этом представляют содержательный интерес и рассматриваются только такие лексические функции, у которых имеются значения, возможные при одних аргументах и невозможные при других.

Значения одной лексической функции от разных аргументов могут полностью или частично совпадать; могут совпадать и значения разных функций от одного аргумента:

$\exists f_1 \neq f_2 \; \exists x : f_1(x) = f_2(x);$

$\exists f \; \exists x_1 \neq x_2 : f(x_1) = f(x_2).$

Необходимо подчеркнуть, что лексические функции введены специально для описания лексической сочетаемости, а не для представления смысла, и потому их не следует трактовать как семантические единицы. Лексические функции совсем не однозначно соотносятся со смыслом слов. В общем случае лексические функции определены не для всех слов и словосочетаний. Например, функции Labor и Func определены лишь для конкретной части речи, или синонимы, которые в принципе возможны для любых слов, а имеются только у некоторых.

В некоторых случаях говорить о лексических функциях как о «многозначных» функциях не совсем корректно и удобно. Удобнее говорить о лексических предикатах. Когда лексические функции представляются в виде предикатов, это не вызывает затруднений. Например, если лексическая функция определена не для всех слов, предикат просто будет ложным.

Пример

Значение предиката $Syn(x, y)$ истинно, если x, y – синонимы; ложно – в противном случае.

Значение предиката $Anti\,(x, y)$ истинно, если x, y – антонимы; ложно – в противном случае.

$Gener\,(x, y)$, y – обобщающее понятие по отношению к понятию, обозначенному x ($x = клубника$, $y = ягода$). Этот предикат зависит от лексической сочетаемости слов в данном языке: если x и m – совпадающие по смыслу слова двух разных языков, то для $Gener\,(x, y)$ и $Gener\,(m, n)$, соответственно, y и n могут не совпадать по смыслу.

Ситуация – определенное лексическое отражение (в данном языке) некоторой части действительности. Ситуации обозначаются отдельными словами (лексемами) и имеют, как правило, от одного до четырех «смысловых» компонентов, или **семантических актантов**, обозначаемых заглавными латинскими буквами A, B, C, D. В то же время каждой такой лексеме сопоставляются **глубинно-синтаксические актанты** – ее зависимые, соответствующие подлежащему и дополнениям (в случае, если данная лексема реализуется глаголом-сказуемым). Глубинно-синтаксические актанты нумеруются арабскими цифрами: 1, 2, 3, 4.

Пример

$Loc(x, y)$, y – предлог типовой локализации (пространственной, временной или абстрактной).

$Loc_{in}(x, y)$, y – «статическая» локализация. Предикат истинен, если $x = Москва$, $y = в$ (например, *в Москве*).

$Loc_{ad}(x, y)$, y – предлог направления. Предикат истинен, если $x = Москва$, $y = в$ (например, *в Москву*).

$Loc_{ab}(x, y)$, y – предлог удаления. Предикат истинен, если $x = Москва$, $y = из$ (например, *из Москвы*).

$Func_1(x, y)$, y – глагол, связывающий название ситуации в роли подлежащего с названиями актантов (если они есть) в роли дополнения. Пре-

дикат выдает истинное значение, если $x =$ *в окно*, $y =$ *светит*, например, в предложении «*Солнце светит в окно*».

$Labor_{12}(x, y)$, y – глагол, связывающий название первого актанта в роли подлежащего с названием второго актанта в роли первого дополнения и с названием ситуации в роли второго дополнения. Предикат выдает истинное значение, если $x =$ *орденом*, $y =$ *наградил*, например, в предложении «*Президент наградил его орденом*»; подлежащее = *президент*, второй актант в роли первого дополнения = *его*.

$Perf(x, y)$, y несет завершенность действия. Предикат выдает истинное значение, если y имеет форму совершенного вида ($x =$ *читать*, $y =$ *прочитать*).

$Destr(x, y)$, y – типовое название «агрессивного» действия ($x =$ *оса*, $y =$ *жалит*).

$Cap(x, y)$, y – «начальник» ($x =$ *факультет*, $y =$ *декан*).

Как уже отмечалось ранее, в общем случае лексическая функция определяется не для всех слов и словосочетаний. Во-первых, некоторые лексические функции определены лишь для той или иной части речи. Так, $Func$ и $Labor$ мыслимы лишь для существительных, а $Perf$ – только для глаголов. Во-вторых, та или иная функция может определяться только для слов определенной семантики. Например: Cap – для слов, смысл которых предполагает наличие «начальника» и «персонала», т. е. для названий учреждений и организаций в самом широком понимании; $Func$ и $Labor$ определены только для названий ситуаций. Если лексические функции представлять в виде предикатов, не возникает никаких затруднений. В рассмотренных выше случаях, когда лексические функции не определены, соответствующие им предикаты будут ложными.

Особую роль при исследовании семантики в подходе И. А. Мельчука играют **валентности слов**, т. е. способности слов вступать в связи с другими словами. Валентностями обладают слова, которые задают ситуацию. Это все глаголы, некоторые существительные (отглагольные), прилагательные (обозначающие сравнение: больше, меньше, выше, ниже), некоторые предлоги и наречия.

Различают два вида валентностей слова: синтаксические и семантические. Хотя это разделение иногда бывает довольно условным. **Семантические валентности** определяются лексическим анализом ситуации, задаваемой конкретным словом. Приведем пример со словом *аренда* или *арендовать*. А *арендует* С – значит, что за какое-то вознаграждение D лицо А приобретает у другого лица В право на эксплуатацию недвижимой

собственности С в течение времени Т. Следовательно, существенными для ситуации *аренды* являются следующие «участники», или семантические актанты: субъект аренды (тот, кто арендует), первый объект аренды (то, что арендуют), контрагент (тот, у кого арендуют), второй объект (плата) и срок.

Эти актанты необходимы, так как устранение какого-либо из них изменяет смысл ситуации. Например, если убрать срок, то ситуация аренды трансформируется в ситуацию купли-продажи. С другой стороны, эти актанты достаточны, поскольку в ситуации аренды не требуется указание того, по какой причине, где, когда и с какой целью она осуществлялась. Хотя соответствующие словоформы грамматически присоединимы к глаголу *арендовать*.

Другими словами, семантическая валентность определяется числом семантических актантов. Таким образом, глагол *арендовать* имеет семантическую валентность 5, т. к. у него пять семантических актантов. Формально эту ситуацию можно записать в виде предиката $P(x_1, x_2, x_3, x_4, x_5)$, где x_1 – «кто», x_2 – «что», x_3 – «у кого», x_4 – «плата», x_5 – «срок».

В предложении могут быть определены не все семантические актанты, некоторые могут просто не упоминаться или вообще не иметь синтаксического выражения. **Синтаксические валентности** определяются количеством синтаксических актантов, которые представлены непосредственно в тексте (т. е. присоединяемыми к глаголу подлежащими и дополнениями) и зависят от контекста.

Например, семантическая валентность глагола *промахнуться* равна 4, т. к. он имеет четыре актанта: кто (деятель), во что/по чему (мишень), из чего (оружие – факультативно) и чем (орган, средство). Но в большинстве контекстов синтаксически выражается лишь одна валентность, например, в предложении *«Он долго целился, но промахнулся»*. Тем не менее, не совсем корректной является фраза *«Он промахнулся в окно бутылкой»*.

Толково-комбинаторный словарь – одно из главных теоретических изобретений И. А. Мельчука. В каком-то смысле можно сказать, что языковая модель, предложенная И. А. Мельчуком, представляет язык как совокупность словарных статей с огромным количеством разнообразной информации; грамматические правила при таком словаре играют скорее второстепенную роль. Толково-комбинаторный словарь отражает прежде всего нетривиальную сочетаемость лексем.

Теория «Смысл ⇔ Текст» с самого начала создавалась для применения в прикладной проблематике автоматического перевода. По замыслу

И. А. Мельчука, с ее помощью, в отличие от традиционных нестрогих теорий, следовало обеспечить построение «действующей» модели языка. Теория «Смысл ⇔ Текст» действительно была использована в некоторых системах машинного перевода, разработанных в России, – прежде всего, в системе англо-русского автоматического перевода ЭТАП, созданной группой под руководством Ю. Д. Апресяна. Все эти системы относятся к числу экспериментальных, то есть их промышленное использование не представляется возможным. Несмотря на то, что они включают много лингвистически полезной информации, в целом ни одна из них пока не обеспечила прорыва в качестве перевода.

На наш взгляд, основная верная идея этой теории состоит в том, что значение слов (языковых единиц) соотносится не непосредственно с окружающей действительностью, а с представлениями носителя языка об этой действительности (иногда называемыми концептами). Природа концептов зависит от конкретной культуры (культурно-специфична); система концептов каждого языка образует так называемую «наивную картину мира», которая во многих деталях может отличаться от «научной» картины мира, являющейся универсальной. Задача семантического анализа лексики в теории «Смысл ⇔ Текст» как раз состоит в том, чтобы обнаружить «наивную картину мира» и описать ее основные категории. Другими словами, важная роль этой теории состоит в описании не только объективной, но и субъективной картины мира.

Несмотря на то, что интерес к теории И. А. Мельчука угасает, разметка синтаксического корпуса проекта «Национальный корпус русского языка» [2] выполняется лингвистическим процессором ЭТАП-3, основанном на принципах теории «Смысл ⇔ Текст».

Как уже упоминалось выше, в разработке процессора ЭТАП участвовал Ю. Д. Апресян. Его идеи несколько отличаются от идей И. А. Мельчука. Центральное место в исследованиях Ю. Д. Апресяна занимает синонимический словарь нового типа. Для этого словаря была разработана подробная схема описания **синонимических рядов**, где каждый элемент ряда характеризовался с точки зрения семантики, синтаксиса, сочетаемости и других свойств. В словаре собрано и обобщено максимальное количество информации о языковом поведении русских синонимов.

Таким образом, согласно теории «Смысл ⇔ Текст»**, язык** – это очень большая модель, в которой определены лексические предикаты, действующие описанным выше образом.

С точки зрения И. А. Мельчука, **статья толково-комбинаторного словаря** несет информацию о валентностях конкретного слова, верную не только в ее рамках, но и в рамках всего языка в целом, т. е. содержит

- основное слово;
- лексические предикаты (характеризующие сочетаемость слов);

– валентность и для каждого актанта указание, в каких падежах и с какими предлогами используются слова, соответствующие данному актанту.

5.2. Теоретико-модельный подход

Теория моделей (раздел математической логики) изучает взаимосвязи между синтаксисом и семантикой (рис. 18). При этом первому в ней отвечает формальный язык, а второму – модель – математическая структура, допускающая некоторое описание этим языком. В качестве формального языка возьмем язык логики предикатов первого порядка.

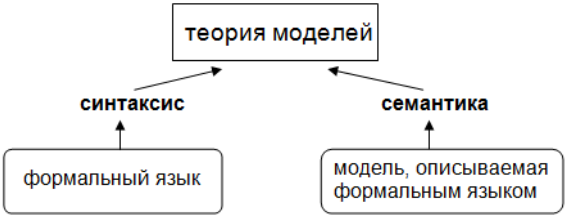


Рис. 18. Связь между синтаксисом и семантикой в теоретико-модельном подходе

Основная идея подхода при построении логических моделей представления знаний состоит в том, что вся информация, необходимая для решения прикладных задач, рассматривается как совокупность фактов и утверждений, которые представляются как формулы в некоторой логике [3]. Знания отображаются совокупностью таких формул, а получение новых знаний сводится к реализации процедур логического вывода. В основе логических моделей представления знаний лежит понятие формальной теории, задаваемое кортежем

$$S =< B, F, A, R >, \text{ где}$$

B – счетное множество базовых символов (алфавит);

F – множество формул (синтаксических правил);

A – выделенное подмножество априори истинных формул (аксиом);

R – конечное множество отношений между формулами, называемое правилами вывода.

Вспомним некоторые определения из математической логики [4].

Сингатурой называется набор $\Sigma = \langle \Sigma_R, \Sigma_F, \Sigma_C, \rho \rangle$, состоящий из множеств

1) Σ_R – символов (имен) для основных **отношений** (предикатов);

2) Σ_F – символов (имен) для основных **операций** (функций);

3) Σ_C – символов (имен) для основных **констант** (выделенных элементов);

4) функции ρ, сопоставляющей именам предикатов и операций их арность, т. е. местность соответствующих операций и отношений $\rho : \Sigma_R \cup \Sigma_F \to \mathbb{N}$.

Алфавит формального языка состоит из

1) символов сигнатуры: $\Sigma_R, \Sigma_F, \Sigma_C, \rho$;

2) символов переменных ($x, y, z, u, v, w, x_1, y_1, z_1, x_2, y_2, z_2, \ldots \in V$);

3) логических (или пропозициональных) связок:

$\wedge$ – конъюнкции («и»),

$\vee$ – дизъюнкции («или»),

$\to$ – импликации («если ..., то»),

$\neg$ – отрицания («не»);

4) кванторов всеобщности $\forall$ и существования $\exists$;

5) дополнительных символов: «=», «(», «)», «,».

Конечные последовательности символов из этого алфавита являются **словами**, но, как и в естественном языке, не все такие последовательности осмысленны. Выделим среди них сначала термы, потом формулы.

Терм сигнатуры Σ с переменными из V определим по индукции.

1. Базис индукции. Любая переменная $x\,(x \in V)$ является термом c, и любой константный символ $c \in \Sigma_C$ нашей фиксированной сигнатуры Σ_C также является термом c.

2. Шаг индукции. Пусть уже построены термы $t_1, \ldots, t_m$, f – символ операции ($f \in \Sigma_F$), и местность операции равна m (т. е. $\rho(f) = m$), тогда выражение $f(t_1, \ldots, t_m)$ также является термом.

Переменная называется **связанной**, если находится в области действия кванторов $\forall, \exists$. В противном случае она называется **свободной.**

Формулу сигнатуры Σ с переменными из V определим по индукции.

1. Базис индукции. Если p, q – термы, то выражение $p = q$ является формулой, а для любой переменной $x\,(x \in V)$ вхождение x в это выражение $p = q$ является свободным вхождением этой переменной в эту форму-

лу; связанных вхождений переменных в эту формулу нет.

Если P – символ n-местного предиката сигнатуры Σ (т. е. $\rho(P)=n$), а $t_1,\ldots,t_n$ – термы этой сигнатуры, то выражение $P(t_1,\ldots,t_n)$ является формулой, а для любой переменной $x\,(x\in V)$, вхождение переменной x в это выражение $P(t_1,\ldots,t_n)$ является свободным вхождением этой переменной в эту формулу; связанных вхождений переменных в эту формулу нет.

Эти формулы называются **атомарными**, и они не имеют других подформул, кроме самих себя.

2. Шаг индукции. Пусть уже построены формулы φ и ψ, тогда выражения $\varphi\wedge\psi,\varphi\vee\psi,\varphi\to\psi,\neg\varphi$ являются формулами. Подформулами формул $\varphi\wedge\psi,\varphi\vee\psi,\varphi\to\psi$ являются они сами и все подформулы формул φ и ψ. Подформулами формулы $\neg\varphi$ является она сама и все подформулы формулы φ.

Любое вхождение переменной x в таким образом определенные формулы является свободным, если оно входило в соответствующем месте в подформулу свободно, и связанным в противном случае.

Определим теперь конструкцию с кванторами.

Если φ – формула, а x – переменная, то определим две новые формулы $(\forall x)\varphi$ и $(\exists x)\varphi$, полученные навешиванием кванторов $\forall$ и $\exists$ по этой переменной x на формулу φ. Подформулами формул $(\forall x)\varphi$ и $(\exists x)\varphi$ являются сами эти формулы и все подформулы формулы φ.

Заметим, что других типов порождения формул в нашем языке нет.

Любое вхождение переменной y, отличной от переменной x, в таким образом определенные формулы является вхождением и в ее подформулу φ. Это вхождение является свободным, если оно входило в соответствующем месте в подформулу свободно, и связанным в противном случае.

Сама переменная x не имеет свободных вхождений в эти формулы, и все ее свободные вхождения в формулу φ в новых формулах уже связаны соответствующим квантором, который навешивается по этой переменной на формулу.

Предложением (или **замкнутой формулой**) называют формулу без свободных переменных.

Теорией первого порядка называют любое множество предложений.

Аксиомы логики первого порядка

$A_1: A \to (B \to A)$;

$A_2: ((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$;

$A_3: A \wedge B \to A$;

$A_4: A \wedge B \to B$;

$A_5: A \to (B \to (A \wedge B))$;

$A_6: A \to (A \vee B)$;

$A_7: B \to (A \vee B)$;

$A_8: \neg A \to (A \to B)$;

$A_9: A \vee \neg A$;

$A_{10}: (A \to C) \to ((B \to C) \to ((A \vee B) \to C))$;

$A_{11}: (A \to B) \to ((A \to \neg B) \to \neg A)$;

$A_{12}: \forall x A \to A[t/x]$;

$A_{13}: A[t/x] \to \exists x A$,

где $A[t/x]$ – формула, полученная в результате подстановки терма t вместо каждой свободной переменной x, встречающейся в формуле A.

Правила логики первого порядка

Правило отделения (Modus ponens):

$$\frac{A, A \to B}{B}$$

Правило обобщения:

$$\frac{A}{\forall x A}$$

Свойства логики первого порядка

Непротиворечивость – ни одна формула не может быть выведена одновременно со своим отрицанием.

Полнота – для любого предложения теории выводимо либо оно само, либо его отрицание (**Теорема Гёделя о полноте**).

Логика второго порядка в математической логике – формальная система, расширяющая логику первого порядка возможностью квантификации общности и существования не только над переменными, но и над предикатами. Логика второго порядка несводима к логике первого порядка. В свою очередь, она расширяется логикой высших порядков и теорией типов. Логика второго порядка обладает несколько другими свойствами, чем логика первого порядка, например, она не имеет свойства полноты.

Являясь формализованым аналогом обычной логики, логика первого порядка дает возможность строго рассуждать об истинности и ложности утверждений и об их взаимосвязи, в частности, о логическом следовании одного утверждения из другого, или, например, об их эквивалентности. Рассмотрим классические примеры формализации утверждений естественного языка в логике первого порядка.

Для естественного языка переменные и константы будем интерпретировать как слова из словаря, а предикаты могут быть введены различными способами, например, так, как изложено в начале этого раздела. Тогда предложения естественного языка можно записывать в виде формул с помощью логических связок $\vee, \wedge, \neg, \rightarrow$ и кванторов $\forall, \exists$.

Пример

Возьмем рассуждение «*Если я буду интенсивно работать, то получу премию. Если я получу премию, то куплю себе велосипед. Следовательно, если я буду интенсивно работать, то куплю себе велосипед*».

Обозначим

$A =$ «*я буду интенсивно работать*»;

$B =$ «*я получу премию*»;

$C =$ «*куплю себе велосипед*».

Тогда рассуждение можно записать в виде: $\big((A \rightarrow B) \wedge (B \rightarrow C)\big) \rightarrow (A \rightarrow C)$.

Получили свойство транзитивности импликации.

Логические модели представления знаний обладают следующими преимуществами.

1. В качестве основы используется классический аппарат математической логики, методы которой достаточно хорошо изучены и формально обоснованы.

2. Существуют довольно эффективные процедуры логически осмысленного вывода информации (например, реализованные в языке логического программирования Пролог или использующие механизмы автоматического доказательства теорем).

3. В базах знаний достаточно хранить лишь множество аксиом, а все остальные знания получать из них по правилам вывода.

5.3. Теоретико-множественный подход С. Маркуса (S. Marcus)

Соломон Маркус – румынский математик, опубликовал свою теорию в 1970 году. Теоретико-множественный подход относится к любым языкам, но изначально создавался на примере французского как флективного языка (с развитой морфологией).

Минимально используя лексику и опираясь лишь на структурные закономерности, Маркус формально определил для широкого класса естественных языков такие понятия как контекст, часть речи, синтаксический тип (для проверки грамматической правильности предложения), род и падеж для существительных [5]. Здесь ограничимся только некоторыми из них. Сначала рассмотрим наиболее простой случай задания языка в виде пары.

Назовем конечное множество слов **словарем** Γ.

Рассмотрим свободную полугруппу T на Γ, т. е. множество всех конечных последовательностей слов с определенной на нем ассоциативной и некоммутативной бинарной операцией конкатенации (сцепления). Полугруппа T в целом будет называться в этом случае **полным** (или **универсальным языком**) над Γ.

Последовательность слов будем называть **последовательностью** (или **цепочкой**) над Γ.

Нулевая последовательность (обозначим θ) – это такая последовательность, что $\theta x = x\theta = x$ для каждой последовательности x. Если это специально не оговорено, то $\theta \notin \Gamma$.

Языком над Γ назовем пару $\{\Gamma, \Phi\}$, где подмножество $\Phi \subseteq T$.

Наиболее важным разбиением Γ, которое встречается в лингвистике, является так называемое разбиение на **дистрибутивные классы**.

Теперь, после выделения лингвистических единиц (фонем и морфем), объединим их в дистрибутивные классы по следующему принципу: единицы языка, сходные в своей дистрибуции, объединяются в один дистрибутивный класс. Так, английские существительные (*boy, book, dog, street, war, peace* и т. п.) выделяются в особый класс не потому, что они могут обозначать предмет (это был бы семантический принцип, к тому же приложимый лишь к части существительных), а потому, что в тексте им могут предшествовать слова из класса детерминативов (*a, the, my, this, some* и т. п.). Сами же они предшествуют словам из класса глаголов (*is, has, looks, can* и т. д.).

Два слова a и b принадлежат одному **дистрибутивному классу**, если для каждой пары последовательностей x, y выполняется условие $xay \in \Phi \leftrightarrow xby \in \Phi$, т. е. последовательность $xay \in \Phi$ тогда и только тогда, когда $xby \in \Phi$.

Контекст определим как упорядоченную пару последовательностей над Γ (обозначается $\langle x, y \rangle$, где $x, y \in T$).

Слово a **допустимо** в контексте $\langle x, y \rangle$, если последовательность

$xay \in \Phi$.

Обозначим через $\mathbf{J}(a)$ множество всех контекстов, в которых допустимо слово a. Отсюда непосредственно следует, что два слова a и b принадлежат одному дистрибутивному классу тогда и только тогда, когда $\mathbf{J}(a) = \mathbf{J}(b)$, т. е. тогда и только тогда, когда a и b допустимы в одних и тех же контекстах.

Ясно, что два различных дистрибутивных класса не пересекаются. Таким образом, эти классы определяют разбиение словаря Γ, которое мы назовем **дистрибутивным разбиением**. Дистрибутивное разбиение, содержащее слово a, будем обозначать $S(a)$.

Более сложным является понятие языка как тройки $\{\Gamma, P, \Phi\}$, где P – разбиение словаря Γ, отличное от разбиения на дистрибутивные классы. Если P – разбиение Γ, то каждое подмножество P мы будем называть **клеткой** из P (или P **-клеткой**).

Если определено разбиение P, то $\Gamma = \bigcup_{i=1}^{n} P_i$, где P_i – клетка из P, n – число клеток.

Поскольку подмножества P_i не пересекаются, каждое слово принадлежит одной и только одной клетке.

Обозначим через $P(a)$ клетку из P, содержащую слово a. Из сказанного следует, что для двух различных слов a и b выполняется либо $P(a) = P(b)$, либо $P(a) \cap P(b) = \varnothing$.

Множество $P(a)$ обычно интерпретируется в естественных языках как множество флективных форм слова a (т. е. форм слова с разными окончаниями). Такая интерпретация требует введения так называемого **единичного разбиения** Γ, в котором каждая клетка состоит из одного слова.

Рассмотрим два разбиения P и Q словаря Γ. Будем говорить, что разбиение P **мельче** разбиения Q, если $P(a) \subseteq Q(a)$ для каждого $a \in \Gamma$.

Пример

Единичное разбиение E является самым мелким разбиением Γ.

Если мы интерпретируем $P(a)$ как множество всех флективных форм a, то разбиение P окажется мельче, чем разбиение Γ на части речи.

Если $x_1 x_2 \ldots x_n$ – последовательность над Γ, то последовательность $P(x_1)P(x_2)\ldots P(x_n)$ называется P **-структурой** последовательности $x_1 x_2 \ldots x_n$.

Далее будем говорить, что P **регулярно мельче** Q, если P мельче

Q, и для каждой тройки слов x, y, z выполняются следующие условия: $P(x) \subseteq Q(z)$ и $P(y) \subseteq Q(z)$ влечет P-эквивалентность $P(x) \leftrightarrow P(y)$.

Для каждого разбиения P над Γ рассмотрим разбиение P', клетки которого определены следующим образом: $P'(x) = \bigcup_{P(x) \leftrightarrow P(y)} P(y)$ (для каждого $x \in \Gamma$), где объединение берется по всем словам y, для которых $P(x) \leftrightarrow P(y)$. Разбиение P', назовем **производным** от разбиения P.

По своему определению разбиение P', таково, что P регулярно мельче P'.

Пример

Единичное разбиение E регулярно мельче, чем разбиение S на дистрибутивные классы, а значит, разбиение S на дистрибутивные классы является производным от единичного разбиения E, т. е. $S = E'$.

E – единичное разбиение – случай, когда каждая клетка состоит из одного слова.

Языком назовем тройку $\{\Gamma, P, \Phi\}$, где

Γ – конечный словарь;

P – разбиение словаря Γ, отличное дистрибутивного;

Φ – подмножество свободной полугруппы над Γ.

Цепью между словами a и b назовем конечную последовательность $x_1, x_2, ..., x_i, x_{i+1}, ..., x_n$ такую, что $x_1 = a$, $x_n = b$ и $x_i \in S(x_{i+1}) \cup P(x_{i+1})$ для $1 < i \le n-1$. Число n определяет длину цепи. Будем считать, что для каждого слова a существует цепь длиной 1, соединяющая a с самим собой.

Обозначим $R(a)$ множество таких слов b, для каждого из которых существует цепь, соединяющая a и b.

Легко заметить, что на множестве $R(a)$ выполняются свойства рефлексивности, симметричности и транзитивности:

1) $a \in R(a)$;

2) если $b \in R(a)$, то $a \in R(b)$;

3) если $c \in R(b)$ и $b \in R(a)$, то $c \in R(a)$.

Таким образом, множества $R(a)$ при $a \in \Gamma$ определяют разбиение R словаря Γ. Такое разбиение назовем **разбиением из смешанных клеток**.

Для каждого $a \in \Gamma$ выполняется $S(a) \subseteq R(a)$ и $P(a) \subseteq R(a)$.

Отсюда видно, что разбиения P и S мельче R. Объединение P и S

составляет R, т. е. $P \cup S = R$.

Обозначим через $K(a)$ множество слов b таких, что выполняется по крайней мере одно из двух следующих условий:

1) $S(a) \cap P(b) \neq \varnothing$;

2) $S(b) \cap P(a) \neq \varnothing$;

Множество $K(a)$ называется **классом** a.

Поскольку $a \in S(a) \cap P(a)$, то, следовательно, $a \in K(a)$ для каждого $a \in \Gamma$.

Язык $\{\Gamma, P, \Phi\}$ называется **адекватным**, если для каждого $x \in \Gamma$ выполняется включение $S(x) \subseteq P'(x)$.

Пример

Простейший пример адекватного языка – язык, в котором P представляет собой единичное разбиение Γ.

Язык $\{\Gamma, P, \Phi\}$ называется **однородным**, если из того, что $S(x) \cap P(y) \neq \varnothing$ $(x \in \Gamma, y \in \Gamma)$, следует $S(y) \cap P(x) \neq \varnothing$.

Рассмотрим язык $\{\Gamma, P, \Phi\}$, где

Γ – словарь естественного языка L,

$P(x)$ (при $x \in \Gamma$) – множество всех форм слова x,

Φ – множество всех правильно построенных предложений языка L.

Если слово x имеет две различные формы x_1 и x_2, то будем считать, что $P(x_1) \cap P(x_2) = \varnothing$.

Пример

Две омонимичные формы, например, *стих* (существительное) и *стих* (глагол), будут рассматриваться как различные слова, а соответствующие им *P*-клетки – как непересекающиеся.

В любом естественном языке выполняется условие: если два слова a и b принадлежат одной парадигме, т. е. если $b \in P(a)$, то они принадлежат одной части речи. В соответствии с этим правилом можно определить понятие «часть речи» как объединение *P*-клеток.

Пусть даны два слова a и b. Клетки $P(a)$ и $P(b)$ относятся к одной **части речи** тогда и только тогда, когда $P(a)$ и $P(b)$ P-эквивалентны, т. е. тогда и только тогда, когда $b \in P'(a)$.

Таким образом, клетки производного разбиения P' при предлагаемой интерпретации могут рассматриваться как приближенная модель частей речи в языке L.

Пусть $\{\Gamma, P, \Phi\}$ – некоторый язык. Два слова – $a \in \Gamma$ и $b \in \Gamma$ – принадлежат одному и тому же **грамматическому роду** (обозначается $a \gamma b$), если для всякого $a' \in P(a)$ и всякого $b' \in P(b)$ выполняется по крайней мере одно из двух следующих условий: $P(a) \cap S(b') \neq \emptyset$, $P(b) \cap S(a') \neq \emptyset$.

Грамматический род представляет собой одну из наиболее интересных тем для исследований в теории грамматики. Изучались различные ее аспекты, такие как связь между значением существительного и его родом, соотношение между родом существительного и его окончанием.

Ясно, что две словоформы могут иметь один и тот же род, даже если они не принадлежат одной и той же части речи. Из-за этого факта возникает противоречие между грамматическим родом и его математической моделью, потому что имеет смысл говорить о роде только для отдельных частей речи. Например, грамматический род у прилагательных совсем иной, чем у существительных. Введенные модели имеют отношение лишь к существительным. Далее, чтобы измерить различие между двумя данными родами, вводится понятие расстояния между родами.

Справедливы следующие теоремы.

Теорема 1

Существует неадекватный язык.

Теорема 2

Если $\{\Gamma, P, \Phi\}$ – адекватный язык, то $R' = P'$.

Общая гипотеза состоит в утверждении, что каждый естественный язык является адекватным. Теорема 2 дает возможность по-новому определить понятие «часть речи». Для данного слова $a \in \Gamma$ его часть речи совпадает с $R'(a)$.

Теорема 3

Пусть $\{\Gamma, P, \Phi\}$ – адекватный язык. Если классы $P(x)$ определяют разбиение K словаря Γ, то $K' = P'$.

Еще одна возможность определения понятия «часть речи» в случае адекватного языка дается теоремой 3. Если в таком языке классы $K(x)$

определяют разбиение Γ, то $K' = P'$. Отсюда часть речи, которой принадлежит слово a, совпадает с $K'(a)$.

Таким образом, становится ясно, что существуют адекватные неоднородные языки, в которых части речи могут быть построены тремя способами: как множество P'-клеток, R'-клеток и K'-клеток. Этот факт важен, поскольку в естественных языках существует много фрагментов, которые являются адекватными, но неоднородными.

Теорема 4

Если язык $\{\Gamma, P, \Phi\}$ однороден, то $K(x) = R(x)$ для каждого $x \in \Gamma$, т. е. классы совпадают со смешанными клетками.

Наиболее благоприятные условия в отношении анализа понятия «часть речи» предоставляют однородные языки. Действительно, согласно теореме 4 в любом однородном языке классы совпадают со смешанными клетками, т. е. $K(x) = R(x)$ для любого $x \in \Gamma$. Поскольку в любом адекватном языке для каждого $x \in \Gamma$ выполняется $R'(x) = P'(x)$, то, следовательно, в однородном языке $K'(x) = R'(x) = P'(x)$ для любого $x \in \Gamma$.

В соответствии с теоремой 4 можно построить части речи, взяв производное разбиение K'. Если a и b – два существительных различного рода, а c – прилагательное, то K-структуры $K(a)$ и $K(b)$ K-эквивалентны. Таким образом, $K(a)$ содержит все существительные. Если a – существительное, а c – прилагательное, то $K(a)$ и $K(b)$ не являются K-эквивалентными. Значит, части речи в том виде, как они описываются приведенной схемой, совпадают с традиционными.

Список литературы

1. Мельчук И. А. Опыт теории лингвистических моделей «Смысл-Текст». М. : Школа «Языки русской культуры», 1999. 346 с.
2. Национальный корпус русского языка. URL: http://www.ruscorpora.ru/.
3. Jurafsky D., Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. 2008. 1024 p.
4. Гончаров С. С. Лекции по математическо логике. Часть 1. НГУ, 2006. URL: http://mmfd.nsu.ru/mmf/persons/gonchar/logic_gonchar.pdf.
5. Маркус С. Теоретико-множественные модели языков. М. : Наука, 1970. 332 с.

ГЛАВА 6. **ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ДЛЯ КОМПЬЮТЕРНОЙ ОБРАБОТКИ**

Существует несколько способов представления знаний об окружающем нас мире для того, чтобы пользоваться ими при создании интеллектуальных информационных систем:

- логические модели;
- продукционные модели;
- сетевые модели;
- фреймовые модели;
- онтологические модели.

Логические модели базируются на логике исчисления предикатов первого порядка. Это удобный инструмент представления в формальном виде семантики текстов на естественном языке. Речь о нем шла в предыдущей главе. В основе логического способа лежит идея описания знаний о предметной области в виде некоторого множества фактов (утверждений, выраженных в виде логических формул, и правил вывода), позволяющих получать новые утверждения в рамках данной системы.

Продукционные модели являются развитием логических моделей с той разницей, что вместо логического вывода в продукционных моделях появляется вывод на знаниях. Связано это с особенностями описания знаний в виде семантической сети. В этом наблюдается заимствование продукционных моделей из сетевых. Подробную информацию про логические, сетевые и продукционные модели представления знаний можно найти, например, в работе [1].

В связи с вышесказанным наибольший интерес представляют три последних способа представления знаний. Остановимся на них подробнее. Неотъемлемой частью этих моделей являются тезаурусы. Можно сказать, что принципы организации тезаурусов легли в основу сетевых, фреймовых и онтологичесих моделей.

6.1. Тезаурусы

Тезаурус – разновидность словарей общей или специальной лексики, в которых указаны семантические отношения между терминами. Термины могут состоять не только из отдельных слов, но и из нескольких лексических единиц.

В отличие от толкового словаря, тезаурус позволяет выявить смысл не только с помощью определения, но и посредством соотнесения слова с другими понятиями и их группами, благодаря чему может использоваться для наполнения баз знаний систем искусственного интеллекта. В тезауру-

сах обычно используются следующие основные семантические отношения: синонимия, антонимия, гипонимия, гиперонимия, тропонимия, меронимия, холонимия и паронимия.

Синонимы – слова одной части речи, различные по звучанию и написанию, но имеющие похожее лексическое значение (*смелый* – ***храбрый, бесстрашный****; азбука* – ***букварь***) .

Антонимы – это слова одной части речи, различные по звучанию и написанию, имеющие прямо противоположные лексические значения (*добрый* – ***злой****; холодный* – ***горячий****; день* – ***ночь***).

Гипоним – понятие, выражающее частную сущность по отношению к другому, более общему понятию (*животное – собака* – ***бульдог****; спорт* – ***хоккей***).

Гипероним – слово с более широким значением, выражающее общее, родовое понятие, название класса предметов, свойств или признаков (*бульдог – собака* – ***животное****; хоккей* – ***спорт***).

Гипероним является результатом логической операции обобщения, тогда как гипоним – ограничения.

Тропоним – слово, обозначающее способ осуществления действия для конкретного действия (*ходить* – ***маршировать****; двигаться* – ***бежать****; говорить* – ***шептать***). Отношение тропонимии для глаголов аналогично отношению гипонимии для существительных.

Мероним – понятие, которое является составной частью другого (*автомобиль* – ***двигатель, колесо, капот****; стадион* – ***трибуна***).

Холоним – понятие, которое является целым над другими понятиями (*двигатель, колесо, капот* – ***автомобиль****; трибуна* – ***стадион***).

Меронимия и холонимия как семантические отношения являются взаимно обратными друг другу, также как гипоним и гипероним.

Паронимы – слова, сходные по форме, но различающиеся по смыслу (*индеец* – ***индиец****; поступок* – ***проступок***).

В настоящее время существует много тезаурусов специального назначения, содержащих термины из конкретно выбранных областей знаний. Тем не менее, есть примеры тезаурусов, состоящих из общеупотребительной лексики (WordNet, Викисловарь и др.).

WordNet – тезаурус английского языка [2]. Базовой словарной единицей WordNet является синонимический ряд (**синсет**), объединяющий слова со схожим значением. Синсеты состоят из слов, принадлежащих той же самой части речи, что и исходное слово. Каждый синсет сопровождается небольшим определением, разъясняющим его значение. Синсеты связаны между собой различными семантическими отношениями, например, отношениями гипонимии, гиперонимии и др. WordNet содержит приблизительно 155 тысяч различных лексем и словосочетаний, организованных в

117 тысяч синсетов. Вся база данных разбита на три части: существительные, глаголы и прилагательные/наречия. Слово или словосочетание может находиться более чем в одном синсете и принадлежать более чем одной категории части речи. Более подробная информация о количестве уникальных слов, синсетов и пар «слово-синсет» в базе WordNet приведена в таблице 6.

Таблица 6

Информация о количестве уникальных слов, синсетов и пар «слово-синсет» в WordNet

Части речи	Уникальные слова	Синсеты	Общее количество пар «слово-синсет»
Существительные	117798	82115	146312
Глаголы	11529	13767	25047
Прилагательные	21479	18156	30002
Наречия	4481	3621	5580
Всего	**155287**	**117659**	**146312**

Преимуществами WordNet перед остальными похожими ресурсами являются его открытость, доступность, наличие большого количества различных семантических связей между синсетами. Доступ к базе WordNet осуществляется непосредственно с помощью браузера (локально или через Интернет) или при помощи библиотек на C. Недостатком является наличие большого количества устаревших и редких слов, отсутствие новой лексики.

Основу описания иерархии синсетов существительных составляют отношения гипонимии и гиперонимии. Другими важными отношениями, связывающими синсеты существительных, являются отношения меронимии и холонимии. Описание синсетов качественных прилагательных базируется на отношении *антонимии*. Для синсетов относительных прилагательных указываются соответствующие синсеты существительных (например, прилагательному *промышленный* соответствует существительное *промышленность*). Основой описания иерархии синсетов глаголов является отношение тропонимии. Кроме тропонимии, синсеты глаголов описываются различными отношениями следствия.

Существуют реализации WordNet для других языков (около 16). Например, для европейских языков создан EuroWordNet, связь между различными языковыми версиями в котором осуществляется через специальный межъязыковой индекс. Ведутся разработки WordNet и для русского языка. Необходимо отметить, что существуют методы предметной классификации синсетов WordNet, т. е. определение областей знаний, в которых они употребляются. Подобная информация может служить в последствии для сокращения количества возможных значений слов, если известна тематика обрабатываемого документа.

Например, если последовательно просматривать определения некоторых понятий в WordNet, можно получить следующую иерархию.

fork – cutlery used for serving and eating food;

cutlery – tableware implements for cutting and eating food;

tableware – articles for use at the table;

article – one of a class of artifacts;

artifact – a man-made object taken as a whole;

object – a tangible and visible entity.

На рис. 19 приведен другой пример иерархии понятий в WordNet (для слова *car*).

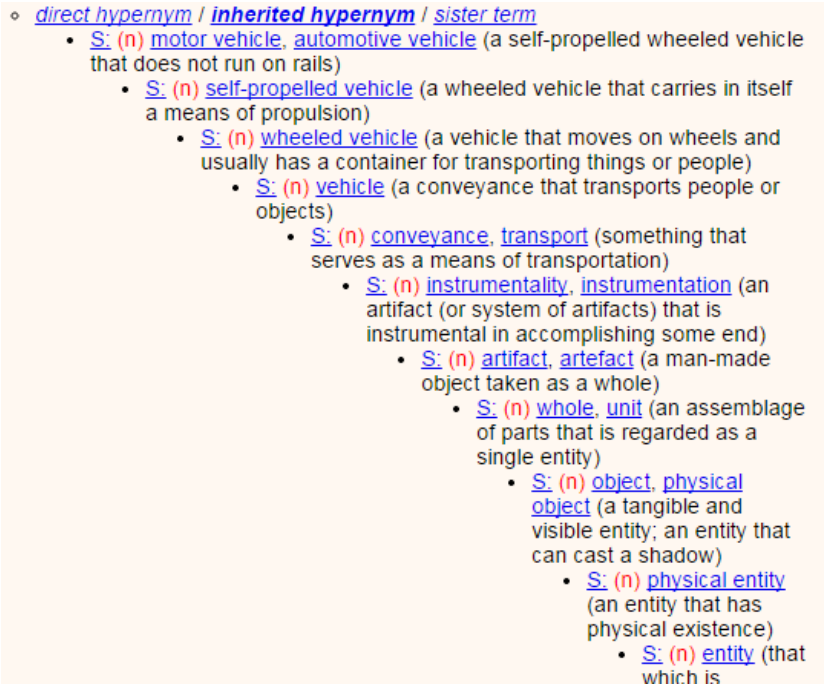


Рис. 19. Иерархия понятий в WordNet

На рисунке видно, что для слова *автомобиль* получается следующая иерархия гиперонимов:

- транспортное средство с мотором;
- транспортное средство на колесах;
- транспортное средство;
- транспортировка;
- играет важную роль в достижении цели;

– артефакт (предмет материальной культуры);
– объект;
– сущность.

Иерархию понятий удобно изображать в виде дерева (рис. 20), в котором наглядно можно увидеть местоположение выбранных слов, например, *car* и *fork*.

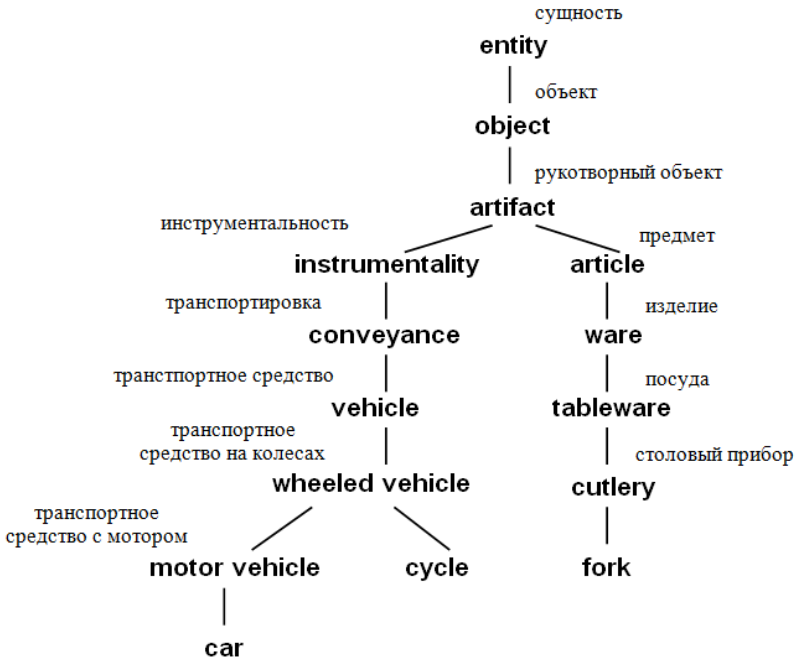


Рис. 20. Представление иерархии понятий в виде дерева

Перейдем теперь к рассмотрению непосредственно сетевых, фреймовых и онтологических моделей.

6.2. Сетевые модели

Семантическая сеть – модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги (ребра) задают отношения между ними. Объектами могут быть понятия, события, свойства, процессы. Таким образом, семантическая сеть отражает семантику предметной области в виде понятий и отношений. Причем в качестве понятий могут выступать как экземпляры объектов, так и их множества.

Семантические сети возникли как попытка визуализации математических формул. За визуальным представлением семантической сети в виде графа стоит математическая модель, в которой каждая вершина соответствует элементу предметного множества, а дуга – предикату.

На рис. 21 приведен пример семантической сети, полученной на основе высказывания *У Иванова есть чёрный BMW*.

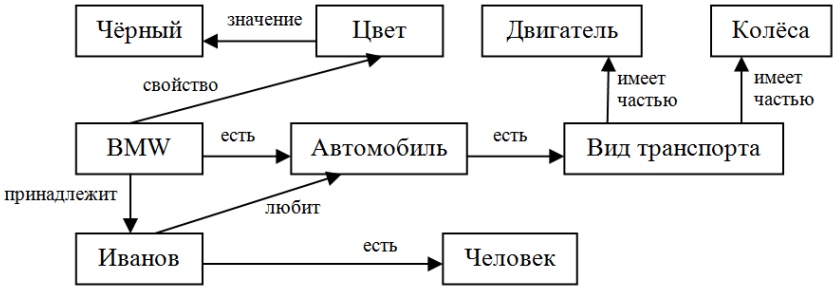


Рис. 21. Пример семантической сети

Терминология, использующаяся в этой области, различна. Чтобы добиться некоторой однородности, узлы, соединенные дугами, принято называть графами, а структуру, где имеется целое гнездо из узлов или где существуют отношения различного порядка между графами, называть сетью. Помимо терминологии, использующейся для пояснения, также различаются способы изображения. Иногда используют кружки вместо прямоугольников; иногда пишут типы отношений над или под дугами, заключая или же не заключая их в овалы; иногда используют аббревиатуры вида O или A для обозначения агента или объекта; иногда используют различные типы стрелок.

Самые первые семантические сети были разработаны в качестве языка-посредника для систем машинного перевода. Последние версии семантических сетей становятся более мощными и гибкими и составляют конкуренцию фреймовым системам, логическому программированию и другим языкам представления знаний.

Несмотря на различную терминологию, разнообразие методов представления кванторов общности и существования и логических операторов, разные способы манипулирования сетями и правила вывода, можно выделить существенные сходства, присущие почти всем семантическим сетям:

1. Узлы семантических сетей представляют собой концепты предметов, событий, состояний.
2. Различные узлы одного концепта относятся к различным значениям, если для них не помечено, что они относятся к одному концепту.

3. Дуги семантических сетей создают отношения между узлами-концептами, пометки над дугами указывают на тип отношения.

4. Некоторые отношения между концептами представляют собой семантические роли, такие как «агент», «объект», «реципиент» и «инструмент». Другие означают временные, пространственные, логические отношения и отношения между отдельными предложениями.

5. Концепты организованы по уровням в соответствии со степенью обобщенности, наподобие иерархии гиперонимов в WordNet. Например, *сущность → живое существо → животное → плотоядное*.

При описании сетевых моделей основными семантическими отношениями являются отношения синонимии и антонимии, а также иерархические отношения между элементами, множествами и частями объектов.

1. Отношения между подмножеством и надмножеством (A Kind Of, ISA). Ясно, что гипоним (более узкое понятие) связан отношением ISA с гиперонимом (более широким понятием). Например: *Canary is a bird*.

2. Отношения между объектом в целом и его частью (HasA) (холонимом и меронимом). Например: *Canary has wings*.

3. Отношения между множеством и объектом, в нем содержащемся (MemberOf, InstanceOf, Example – это отношение, обратное к отношению ISA). Например: *An example of a bird is a canary*.

Заметим, что среди семантических отношений, используемых для описания сетей, могут быть не только семантические отношения, используемые в тезаурусах, но и другие виды связей: функциональные (определяемые обычно глаголами *производит*, *влияет*, …); количественные (*больше*, *меньше*, *равно*, …); пространственные (*далеко от*, *близко от*, *под*, *над*, …); временные (*раньше*, *позже*, *в течение*, …); атрибутивные (*иметь свойство*, *иметь значение*); логические (*И*, *ИЛИ*, *НЕ*) и другие.

На рис. 22 представлена реализация семантической сети на языке LISP при помощи ассоциативных списков. В качестве примера проиллюстрированы основные отношения: ISA и HAS.

```
(defun *database* ()
  ((canary  (is-a bird)
            (color yellow)
            (size small))
   (penguin (is-a bird)
            (movement swim))
   (bird    (is-a vertebrate)
            (has-part wings)
            (reproduction egg-laying))))
```

Рис. 22. Описание семантической сети на языке LISP

Как видно, при помощи функции *assoc* с ключом *canary* можем извлечь из семантической сети всю информацию о типе *canary*.

Следует отметить особенности использования некоторых типов отношений в семантических сетях. В семантической сети в качестве понятий могут быть как экземпляры объектов, так и их множества. Использование одних и тех же отношений и для элементов, и для коллекций может привести к недоразумениям. Рассмотрим в качестве примера четыре предложения.

1. У Пети есть отец по имени Иван.
2a. Для Пети найдется отец из множества мужчин.
2b. Найдется человек, для которого Иван – отец.
3. У каждого человека есть отец из множества мужчин.

Для человека ясен смысл этих фраз, и многие не задумываясь поставили бы во всех трех случаях отношение «есть отец». Однако это является ошибкой: в первом случае действительно описывается отношение между двумя экземплярами, но во втором и третьем – между экземпляром и множеством, а в четвертом – отношение между представителями из двух множеств. В математической записи это выглядит следующим образом для каждого из предложений соответственно:

1. $\exists$ петя & $\exists$ иван : отец(иван, петя);
2a. $\exists$ петя $\rightarrow \exists$ x $\in$ мужчины : отец(x, петя);
2b. $\exists$ иван $\rightarrow \exists$ y $\in$ люди : отец(иван, y);
3. $\forall$ y $\in$ люди $\rightarrow \exists$ x $\in$ мужчины : отец(x, y).

Несмотря не некоторые различия, сетевые модели удобны для чтения и обработки компьютером, являются наглядным и достаточно универсальным средством представления семантики естественного языка. Однако их формализация в конкретных моделях представления, использования и модификации знаний оказывается достаточно трудоемкой, особенно при наличии множественных отношений между ее элементами. Итак, перечислим недостатки сетевых моделей.

1. Сетевая модель не содержит ясного представления о структуре предметной области, поэтому формирование и модификация такой модели затруднительны.
2. Сетевые модели представляют собой пассивные структуры, для обработки которых необходим специальный аппарат формального вывода.
3. Проблема поиска решения в семантической сети сводится к задаче поиска фрагмента сети, который соответствует подсети, отражающей поставленный запрос. Это обуславливает сложность поиска вывода решения на семантических сетях.
4. Представление, использование и модификация знаний при описании систем реального уровня сложности оказывается трудоемкой про-

цедурой, особенно при наличии множественных отношений между ее понятиями.

Рассмотрим, например, некоторую сеть, описывающую высказывание *Настя попросила книгу у Даши*. Допустим, можно приписать приведенным объектам свойства: *Настя* – «*старательная*», *Даша* – «*любознательная*». Между этими объектами есть связь (через книгу). Но, кроме нее, есть много других связей, которые есть в реальном мире: социальный статус (студентки, подруги – не обязательно между собой), родственные отношения (у каждой есть родители и/или другие родственники) и т. д. Получается, что даже для такого простого примера сеть способна разрастись до большого размера, и как следствие, поиск вывода в ней будет слишком сложным.

В сложных семантических сетях, включающих множество понятий, процесс обновления узлов и контроль связей между ними, как видим, усложняют процедуру обработки информации. Стремление устранить эти недостатки послужило причиной появления других способов представления знаний.

6.3. Фреймовые модели

Фреймовые модели представления знаний были предложены М. Минским в 1975 году.

Фреймом называется структура для описания понятия или ситуации, состоящая из характеристик этой ситуации и их значений [3]. Фрейм можно рассматривать как фрагмент семантической сети, предназначенный для описания понятий со всей совокупностью присущих им свойств. Большинство систем искусственного интеллекта используют набор фреймов, которые соединены друг с другом и образуют определенную иерархию. Одним из наиболее важных свойств фреймов в таких иерархиях является наследование свойств. Фрейм-потомок содержит фактические значения атрибутов-слотов (такие же, как в родительском фрейме, который представляет более общее описание сущности). На рис. 23 приведен пример сети фреймов.

Особенность фреймовых моделей представления знаний состоит в том, что все понятия, описываемые в каждом из узлов модели, определяются набором атрибутов и их значениями, которые содержатся в слотах фрейма.

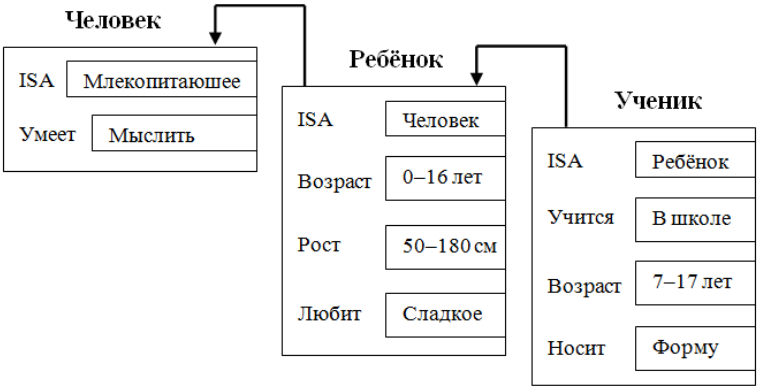


Рис. 23. Пример сети фреймов

Графически фреймовая модель выглядит аналогично семантической сети, но принципиальное отличие состоит в том, что каждый узел во фреймовой системе имеет обобщенную структуру в виде таблицы 7.

Таблица 7

Обобщенная структура фрейма

ПОНЯТИЕ (имя фрейма)			
Имя слота	**Указатель наследования**	**Указатель типа**	**Значение слота**
слот 1 слот 2 ... слот N			

Слот – это атрибут, связанный с узлом в системе, основанной на фреймах. Он является составляющей фрейма.

Указатель наследования показывает, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с теми же именами во фрейме нижнего уровня. Типичные указатели наследования:

- S (тот же) – слот наследуется с теми же значениями данных;
- U (уникальный) – слот наследуется, но данные в каждом фрейме могут принимать любое значение;
- I (независимый) – слот не наследуется.

Указатель типа данных содержит информацию о типе данных, включаемых в слот. Обычно используются следующие типы данных: FRAME (указатель на имя фрейма верхнего уровня); ATOM (переменная); TEXT (текстовая информация); LIST (список); LISP (присоединенная процедура).

Значением слота может быть экземпляр атрибута, другой фрейм или фасет. Фасеты описывают: тип значений, разрешенные значения, число значений и другие свойства значений, которые может приобретать слот. С каждым слотом может быть связана одна или несколько процедур, которые выполняются, когда изменяются значения слотов. Чаще всего со слотами связываются следующие процедуры:

- ЕСЛИ-ДОБАВЛЕНО (IF-ADDED) – выполняется в том случае, когда новая информация помещается в слот;
- ЕСЛИ-УДАЛЕНО (IF-REMOVED) – выполняется при удалении информации из слота;
- ЕСЛИ-НУЖНО (IF-NEEDED) – выполняется при запросе информации из слота в том случае, когда слот пуст.

Эти процедуры могут следить за приписыванием информации к данному узлу и проверять, что при изменении значения производятся соответствующие действия.

Различают фреймы-образцы (прототипы), хранящиеся в базе знаний, и фреймы-экземпляры, которые создаются для отображения реальных ситуаций на основе поступающих данных.

Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире через фреймы-структуры (для обозначения объектов и понятий: *заем, залог, вексель*); фреймы-роли (*менеджер, кассир, клиент*); фреймы-сценарии (*банкротство, собрание акционеров, празднование именин*); фреймы-ситуации (*тревога, авария, рабочий режим устройства*) и др.

Важнейшим свойством фреймовых моделей является заимствование свойств наследования из моделей семантических сетей. И во фреймах, и в семантических сетях наследование происходит по *ISA*. Слот *ISA* указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т. е. переносятся, значения аналогичных слотов.

Специальные языки представления знаний в сетях фреймов позволяют эффективно строить промышленные системы. К числу таких языков следует отнести: FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймовая оболочка Kappa, PILOT/2 и другие программные средства.

Основным преимуществом фреймовой модели представления знаний является соответствие современным представлениям об организации долговременной памяти человека, а также ее гибкость и наглядность. Наиболее ярко достоинства фреймовых систем представления знаний проявляются в том случае, если родовидовые связи изменяются нечасто, и предметная область насчитывает немного исключений.

6.4. Онтологические модели

Онтология – это точная спецификация некоторой области, которая включает в себя словарь терминов предметной области и множество логических связей, описывающих соотношения этих терминов между собой.

Обычно выделяют [4] следующие основные элементы онтологий:

- экземпляры;
- классы объектов (понятий);
- атрибуты (описывают свойства классов и экземпляров);
- функции (описывают зависимости между классами и экземплярами);
- аксиомы (дополнительные ограничения).

Специализированные (предметно-ориентированные) онтологии – это представление какой-либо области знаний или части реального мира. В такой онтологии содержатся специальные для этой области значения терминов. К примеру, слово *поле* в сельском хозяйстве означает участок земли, в физике – один из видов материи, в математике – класс алгебраических систем.

Общие онтологии используются для представления понятий, общих для большого числа областей. Такие онтологии содержат базовый набор терминов, глоссарий или тезаурус, используемый для описания терминов предметных областей. Обычно онтологии создаются для решения конкретных задач.

Как правило, при построении онтологий придерживаются следующих принципов.

1. Формализация, т. е. описание объективных элементов действительности в единых, строго определенных образцах (терминах, моделях).
2. Использование ограниченного количества базовых терминов (сущностей), на основе которых конструируются все остальные понятия.
3. Внутренняя полнота и логическая непротиворечивость.

В центре большинства онтологий находятся классы, которые описывают понятия предметной области. Атрибуты описывают свойства классов и экземпляров. Этим онтологические модели сходны с фреймовыми. Многие понятия и принципы реализации, а также графическая форма представления на начальном этапе структуризации, являются в онтологиях сходными с сетевыми моделями. Основным отличием онотологических моделей от сетевых и фреймовых является необходимость выполнения требований внутренней полноты и логической непротиворечивости используемых понятий. Во фреймовых и сетевых моделях эти требования могут не выполняться.

Следует отметить сходство онтологий с тезаурусами. Однако онтологические модели ориентированы на использование непосредственно компьютером, поэтому структуры данных в онтологиях описаны не на естественном, как это принято в тезаурусах, а на специальном формальном языке. Для описания онтологий используются такие формальные языки, как RDF, OWL, KIF, CycL, OCML и другие.

Сравнение тезаурусов, семантических сетей и онтологий по некоторым признакам приведено в таблице 8.

Таблица 8

Сравнение тезаурусов, семантических сетей и онтологий

	Тезаурусы	**Сем. сети**	**Онтологии**
Форма представления в виде графа	+	+	+
Структуры данных описаны на естественном языке (низкий уровень формализации)	+	+	–
Структуры данных описаны на формальном языке (высокий уровень формализации)	–	–	+
Понятия могут объединяться в классы; атрибуты описывают свойства классов и экземпляров; отношения описывают зависимости между классами и экземплярами	–	+	+
Необходимость выполнения требований внутренней полноты и логической непротиворечивости используемых понятий	–	–	+

Современные онтологические модели являются модульными, то есть состоят из множества связанных между собой онтологий, каждая из которых описывает отдельную предметную область или задачу. Онтологические модели не являются статичными, они постоянно меняются. Если использующая специализированные онтологии система развивается, то может потребоваться объединение онтологий. Существенным недостатком онтологических моделей является сложность их объединения. Онтологии даже близких областей могут быть несовместимы друг с другом. Разница может появляться из-за особенностей местной культуры, идеологии или вследствие использования другого языка описания. Объединение онтологий выполняют как вручную, так и в полуавтоматическом режиме. В целом это трудоёмкий, медленный и дорогостоящий процесс. Онтологические модели широко применяются в системах, основанных на знаниях: экспертных системах и системах поддержки принятия решений.

6.5. Методы измерения семантического расстояния

В данном разделе приведены методы измерения семантического расстояния. Выбор меры расстояния и весов для классифицирующих свойств – очень важный этап, так как от этих процедур зависят состав и количество формируемых классов, а также степень сходства объектов внутри классов. Сама тема классификации и кластеризации текстов подробно рассмотрена в следующем разделе.

Семантическим расстоянием называется количественная величина, показывающая насколько различны значения слов между собой. **Семантической близостью** называется количественная величина, показывающая насколько похожи значения слов между собой.

Другими словами, семантическое расстояние отражает, насколько удалены друг от друга понятия; семантическая близость является величиной, обратной семантическому расстоянию, и показывает, насколько близки друг к другу понятия.

Существуют три основных типа методов измерения семантического расстояния:

- тезаурусные;
- статистические;
- гибридные.

Подробно о методах семантического расстояния можно почитать, например, в работах [5, 6].

1. Тезаурусные методы основаны на использовании сторонних источников знаний, таких как специализированные словари, онтологии, тезаурусы. Эти методы позволяют напрямую вычислить показатель близости двух слов или фраз. В основе определения семантической меры тезаурусными методами лежит предположение о том, что используемая иерархия (таксономия) понятий содержит всю необходимую для вычисления семантической близости информацию.

Считается, что два понятия семантически близки, если они находятся рядом в иерархии. Наглядный пример вычисления семантического расстояния между терминами изображен на рис. 24. Числами обозначено количество ребер в кратчайшем пути, соединяющем два термина.

На практике удобнее оперировать числами из одного и того же промежутка, например, из отрезка [0, 1]. Тогда для примера на рисунке получаем, что $sim(car, cycle) = \frac{1}{3} = 0{,}333$; $sim(car, fork) = \frac{1}{11} = 0{,}091$.

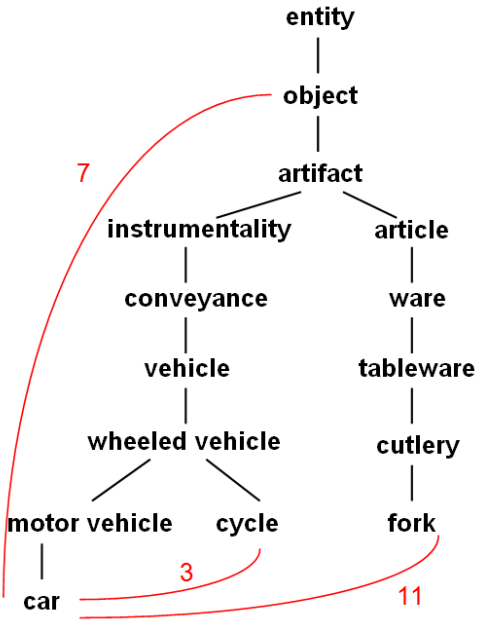


Рис. 24. Вычисление семантического расстояния между понятиями

Простейшая мера вычисляется по формуле:

$$sim(w, w') = -\log\left(\frac{pathlen(w, w')}{2 \cdot D}\right), \text{ где}$$

$pathlen(w, w')$ – число ребер в кратчайшем пути, соединяющем термины w и w' в иерархии тезауруса;

D – максимальная глубина тезауруса.

Мера Ву-Палмера (Wu & Palmer, 1994) вычисляется по формуле:

$$sim_{WP}(w, w') = \frac{2 \cdot depth(LCS)}{depth(w) + depth(w')}, \text{ где}$$

$LCS(w, w')$ – такой общий предшественник для w и w', который находится на кратчайшем расстоянии от них (*Lowest Common Subsumer*);

$depth(w)$, $depth(w')$ – глубина расположения терминов w и w' соответственно в иерархии тезауруса.

Согласно мере Ву-Палмера чем ближе общий «предок» к терминам, тем больше их сходство.

Самый очевидный недостаток тезаурусных методов состоит в том, что необходимо существование тезауруса с заданной иерархией на подходящем естественном языке.

Еще одной проблемой тезаурусных методов является то, что одни части таксономии могут быть весьма развиты и содержать значительное число терминов, тогда как в других частях иерархии плотность терминов может быть меньше. Такая разница в «плотностях» терминов делает эту и другие меры сходства, основанные на простом учете количества ребер в графе, недостаточно точными. Тезаурус и используемая в нем иерархия могут оказаться неполными, т. к. могут отсутствовать слова и словосочетания или могут быть упущены связи между некоторыми понятиями. По этой причине тезаурусные методы, как правило, хорошо работают применительно к существительным и намного хуже для глаголов и прилагательных.

Третий недостаток тезаурусных методов заключается в том, что расстояние между понятиями зависит от выбранного значения многозначного слова. Для слов, имеющих несколько значений, должны рассматриваться разные иерархии. Тогда встает вопрос, какую из иерархий следует выбрать.

2. Статистические методы опираются на частоту упоминаний терминов, являющихся общими в описаниях сравниваемых объектов. Эти методы используют неразмеченные корпуса текстов. Каждое слово рассматривается как точка N-мерного пространства, задаваемого контекстом. Далее для вычисления семантической близости между понятиями выполняется следующая последовательность действий.

1. Задать два понятия через векторы признаков.
2. Применить меру близости векторов.
3. Близкие семантически понятия имеют похожий контекст, поэтому если векторы, ассоциированные с контекстом, близки, то и понятия семантически близки.

Согласно **гипотезе разреженности** обычно документ относится к небольшому количеству тем, т. е. тема определяется относительно небольшим числом терминов. Поэтому правильнее оценивать близость не произвольных слов, а тех, которые по тексту находятся близко к изучаемому слову или синтаксически с ним связаны.

Удаленность изучаемого слова от слов в тексте, задающих контекст (т. е. тех, которые являются признаками), можно задавать заранее, если это требуется. Например, размер «окна» может быть от –10 до +10 слов от изучаемого. Так, в работе [7] было показано, что по силе влияния контекста на изучаемое слово для 10 ближайших соседей (по 5 слева и справа) порядок будет следующий: –1, +1, –2, –3, +2. У оставшихся элементов –4,

+3, –5, +4, +5 происходит резкое уменьшение «силы» влияния. Поэтому часто их можно отбрасывать без ущерба для точности подсчетов. Это сокращает размер словаря на 35–40 % и увеличивает скорость обработки на 10 %. В качестве тестового материала в этих экспериментах был взят «Национальный корпус РЯ».

В тезаурусных методах, рассмотренных ранее, в зависимости от принципов построения иерархии отношение близости между понятиями не всегда является транзитивным, т. е. для таких мер не выполняется неравенство треугольника. В статистических же методах всегда требуется выполнение условий метрики, т. е. в качестве меры близости рассматривается числовая функция ρ , удовлетворяющая условиям:

$\rho(x,y)=0 \Leftrightarrow x=y$ (*аксиома тождества*);

$\rho(x,y)=\rho(y,x)$ (*аксиома симметрии*);

$\rho(x,y)\le \rho(x,z)+\rho(z,y)$ (*аксиома треугольника* или *неравенство треугольника*).

Перечислим наиболее распространенные метрики, которые могут использоваться для вычисления семантического расстояния между терминами.

Евклидова метрика

$$\rho(x,y)=\sqrt{\sum_{i=1}^{N}(x_i-y_i)^2}\ .$$

Применение евклидова расстояния оправдано в следующих случаях:

- свойства (признаки) объекта однородны по физическому смыслу и одинаково важны для классификации;
- признаковое пространство совпадает с геометрическим пространством.

Квадрат евклидова расстояния

$$\rho(x,y)=\sum_{i=1}^{N}(x_i-y_i)^2\ .$$

Придает большие веса более отдаленным друг от друга объектам.

Взвешенное евклидово расстояние

$$\rho(x,y)=\sqrt{\sum_{i=1}^{N}w_i(x_i-y_i)^2}\ .$$

Применяется, когда каждому свойству удается приписать «вес». Определение весов, как правило, связано с дополнительными исследованиями, например, организацией опроса экспертов и обработкой их мнений.

Хеммингово расстояние (Манхэттенская метрика, расстояние городских кварталов)

$$\rho(x, y) = \sum_{i=1}^{N} |x_i - y_i| .$$

Для этой меры влияние отдельных больших разностей (выбросов) уменьшается (так как они не возводятся в квадрат). Это расстояние также называется манхэттенским, сити-блок расстоянием или расстоянием городских кварталов. Хеммингово расстояние является разностью по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида.

Расстояние Чебышева

$$\rho(x, y) = \max_i |x_i - y_i| .$$

Применяется, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате.

Процент несогласия

$$\rho(x, y) = value\, |x_i \neq y_i| .$$

Эта мера используется в тех случаях, когда признаки объекта являются категориальными. Например, первый признак объекта – пол, второй – возраст, третий – место работы. Представим значения свойств (признаков) объекта в виде вектора значений. Первый вектор = (муж, 30 лет, программист), второй вектор = (муж, 28 лет, менеджер). Процент несогласия равен 2/3. Эти вектора различаются на 66,6 %.

Метрика Минковского

$$\rho(x, y) = \left(\sum_{i=1}^{N} (x_i - y_i)^r \right)^{1/r} .$$

При r = 2 получаем евклидову метрику. Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются.

Степенное расстояние (обобщенное расстояние Минковского или пользовательская метрика)

$$\rho(x,y)=\left(\sum_{i=1}^{N}(x_i-y_i)^p\right)^{1/r}.$$

Параметры r и p определяются пользователем.

Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам; параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами.

При $r = p = 2$ получаем евклидову метрику.

Косинус угла между векторами

$$\rho(x,y)=\frac{x\cdot y}{\|x\|_2\|y\|_2}=\frac{\sum_{i=1}^{N}x_i\cdot y_i}{\sqrt{\left(\sum_{i=1}^{N}x_i^{\,2}\right)\left(\sum_{i=1}^{N}y_i^{\,2}\right)}}.$$

Французская железнодорожная метрика

$$\rho(x,y)=\begin{cases}\|x-y\|, & x-p=\lambda(y-p)\\ \|x-p\|+\|y-p\|, & x-p\neq\lambda(y-p)\end{cases},\ \text{где}$$

λ – заданный коэффициент;

p – фиксированная выбранная точка, через которую обязательно должен проходить путь между x и y.

Дивергенция Йенсена–Шеннона (Jensen & Shannon)

$$\rho(x,y)=\sqrt{\sum_i\left(x_i\log\frac{2x_i}{x_i+y_i}+y_i\log\frac{2y_i}{x_i+y_i}\right)}.$$

Основным свойством этой функции можно считать то, что чем дальше x и y друг от друга (чем слабее выражены предпочтения), тем меньше значение функции (меньше сила влияния признака), и наоборот. Существует формула дивергенции Йенсена–Шеннона не только для двух, но и для наборов векторов. В общем виде она выводится из формулы для расстояния Кульбака–Лейблера (Kullback S., Leibler R. A.), иначе называемого относительной энтропией для вероятностных распределений.

Для вычисления веса ассоциации может применяться, например, мера с поточечной взаимной информацией или T-тест ассоциативная мера.

Ассоциативная мера с поточечной взаимной информацией (Pointwise Mutual Information)

$$sim_{PMI}(x, y) = \log \frac{p(x, y)}{p(x)p(y)}, \text{ где}$$

$p(x)$ – частота слова x в корпусе;

$p(x, y)$ – частота совместной встречаемости слов x и y.

T-тест ассоциативная мера

$$sim_{T-test}(x, y) = \frac{p(x, y) - p(x)\,p(y)}{\sqrt{p(x)\,p(y)}}, \text{ где}$$

$p(x)$ – частота слова x в корпусе;

$p(x, y)$ – частота совместной встречаемости слов x и y.

У статистических методов можно выделить следующие недостатки.

К недостаткам статистических методов следует отнести отсутствие наглядности представления при большом числе терминов. Оценка сходства терминов при помощи статистических методов не всегда соответствуют содержательному представлению о семантической близости.

Еще один недостаток состоит в следующем. По идее, если добавлять общие свойства к объектам, то можно было бы точнее оценить семантическое расстояние между ними. Однако статистические методы не чувствительны к такому преобразованию.

3. Гибридные методы сочетают в себе тезаурусные и статистические методы, т. е. предполагают использование тезаурусов и корпусов текстов для измерения семантического расстояния между терминами. Примерами таких мер являются мера Резника, Линя и Цзяна–Конрата.

Мера Резника (Resnik, 1995) вычисляется по формуле:

$$sim_{\mathrm{Resnik}}(w, w') = -\log P(LCS(w, w')), \text{ где}$$

$LCS(w, w')$ – такой общий предшественник для w и w', который находится на кратчайшем расстоянии от них (*Lowest Common Subsumer*);

$P(w)$ – вероятность встретить произвольное слово w в корпусе (вычисляется как отношение частоты данного слова в корпусе к количеству всех слов в корпусе);

$P(LCS(w, w'))$ – означает, что вместо вероятности самого w подставляем в формулу вероятность ближайшего общего предшественника.

Таким образом, если $LCS(w, w')$ – корень иерархии, то $P(LCS(w, w')) = 1$.

Величину $IC(w) = -\log P(w)$ называют **информационным содержанием** (*Information Content*) термина w.

Используя тезаурус WorldNet и оценки частотности слов, полученные из большого массива английских текстов, Резник вычислил семантическое сходство пар слов путем выделения общего предка с наибольшим информационным контентом (содержанием).

Для слов с несколькими значениями в этой мере выбирается тот смысл, который дает максимальную меру сходства. Поэтому мера Резника не может быть использована в случае, когда объект обозначен не одним, а несколькими терминами. Кроме того, диапазон ее значений не нормализован.

Мера Линя (Lin, 1998) следует из теории подобия между любыми объектами:

$$sim_{Lin}(w, w') = \frac{2 \cdot \log P(LCS(w, w'))}{\log P(w) + \log P(w')}.$$

Линь предложил аксиоматическое определение сходства и показал, как подход Резника может быть адаптирован для использования в указанной ситуации. Мера сходства Резника была основана только на общности значений слов, в то время как подход Линя принимает и учитывает значение разницы смыслов слов для определения нормализованного коэффициента сходства. Однако мера Линя не может быть применена в случае, когда в описании объекта используются различные термины.

Мера Цзяна–Конрата (Jiang & Conrath, 1997)

$dist_{JC}(w, w') = 2 \times \log P(LCS(w, w')) - (\log P(w) + \log P(w'))$, где

$sim_{JC}(w, w') = \frac{1}{dist_{JC}(w, w')}$ – семантическая близость – величина, обратная семантическому расстоянию.

Список литературы

1. Загорулько Ю. А. Инженерия знаний : учеб. пособие. Новосибирск : НГУ, 2011. URL: http://193.124.209.204/default.aspx?db=book_zagorulko&int=VIEW&el=1684&templ=I206.
2. WordNet. A lexical database for English. URL: wordnet.princeton.edu.
3. Хабаров С. П. Представление знаний в информационных системах. конспекты лекций. URL: http://www.habarov.spb.ru/bz/bz07.htm.
4. Константинова И. С., Митрофанова О. А. Онтологии как системы хранения знаний // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. 54 с.
5. Крюков К. В., Панкова Л. А., Пронина В. А., Суховеров В. С., Шипилина Л. Б. Меры семантической близости в онтологии // Проблемы управления. 2010. № 5. С. 2–14.
6. Панченко А. Метрики семантической близости с приложениями к задачам в АОТ. URL: http://digsolab.ru/uploads/files/presentation.pdf.
7. Зеленков Ю. Г., Сегалович И. В., Титов В. А. Вероятностная модель снятия морфологической омонимии на основе нормализующих подстановок и позиций соседних слов // Компьютерная лингвистика и интеллектуальные технологии. Труды международного семинара Диалог-2005. 2005. С. 188–197.
8. Соловьев В. Д., Добров Б. В., Иванов В. В., Лукашевич Н. В. Онтологии и тезаурусы : учеб. пособие. Казань ; Москва, 2006. 157 с.

ГЛАВА 7. **МЕТОДЫ КЛАССИФИКАЦИИ И КЛАСТЕРИЗАЦИИ**

Классификация (категоризация) – отнесение документа к одной из нескольких заранее определенных категорий на основании содержания документа.

Кластеризация – выявление групп (их количество заранее не определено) семантически похожих документов среди заданного фиксированного множества документов.

Иногда в литературе можно встретить термин **рубрикация**, в некотором смысле обобщающий понятия классификации и кластеризации.

Можно выделить три основных группы методов классификации и кластеризации текстов.

Во-первых, иногда классификация или кластеризация осуществляется полностью вручную. Например, в библиотеке, когда библиотекарь присваивает книгам тематические рубрики. Подобное ручное распределение дорого и неприменимо в случаях, когда необходимо разделить на группы большое количество документов с высокой скоростью.

Другой подход заключается в написании правил, по которым можно отнести текст к той или иной группе. Например, пусть одно из правил имеет следующий вид: «если текст содержит слова “производная” и “уравнение”, то отнести его к категории *математика*». Специалист, знакомый с предметной областью и обладающий навыком написания регулярных выражений, может составить ряд правил, которые затем автоматически применяются к поступающим документам для их классификации. Этот подход лучше предыдущего, поскольку процесс рубрикации автоматизируется и, следовательно, количество обрабатываемых документов практически не ограничено. Более того, построение правил вручную может дать наилучшую точность. Однако создание и поддержание правил в актуальном состоянии требует постоянных усилий специалиста.

Третий подход основывается на машинном обучении. Для решения задачи классификации документов применяются методы машинного обучения с учителем (обучения по прецедентам). Согласно им набор правил или критерий принятия решения текстового классификатора, вычисляется автоматически из обучающих данных, т. е. осуществляется обучение классификатора. Обучающие данные – это некоторое количество образцов документов из каждого класса. Приписывание класса образцу-документу производится вручную. Этот процесс называют **разметкой.** Разметка является более простой задачей, чем написание правил. Кроме того, разметка может быть произведена в обычном режиме использования системы.

Например, в программе электронной почты может существовать возможность помечать письма как спам, тем самым формируя обучающее множество для классификатора – фильтра нежелательных сообщений. Таким образом, классификация текстов, основанная на машинном обучении, является примером обучения с учителем, где в роли учителя выступает человек, задающий фиксированный набор классов и размечающий обучающее множество.

Задачу кластеризации документов решают при помощи методов машинного обучения без учителя. В этом случае не требуется предварительная разметка части документов, и количество классов тоже никак не фиксируется.

Автоматические методы классификации и кластеризации позволяют ограничить области поиска в поисковых системах, а значит, повысить скорость и точности поиска, когда выбирается заведомо более релевантное подмножество документов и исключается заведомо менее релевантное. Также методы автоматической рубрикации применяются для составления интернет-каталогов, подбора контекстной рекламы, фильтрации спама, определения кодировки и языка текста и пр.

7.1. Классификация текстов, машинное обучение с учителем

Формальную постановку задачи классификации можно записать следующим образом [1]. Имеется множество документов $D = \{d_1,\ldots,d_{|D|}\}$. Имеется множество возможных категорий (классов) $C = \{c_1,\ldots,c_{|C|}\}$. Неизвестная целевая функция $\Phi : D \times C \to \{0, 1\}$ задается формулой:

$$\Phi(d_j, c_i) = \begin{cases} 0, \textit{если } d_j \notin c_i \\ 1, \textit{если } d_j \in c_i. \end{cases}$$

Требуется построить классификатор Φ', максимально близкий к Φ. Если классификатор выдает точный ответ $\Phi' : D \times C \to \{0, 1\}$, то классификация называется **точной.** Если классификатор определяет степень подобия (Categorization Status Value) документа $CSV_i : D \to [0, 1]$, то классификация называется **пороговой**.

В общем случае процесс обучения с учителем (обучения по прецедентам, Supervised learning) заключается в следующем. Системе предъявляется набор примеров, связанных с какой-либо заранее неизвестной закономерностью. Этот набор примеров иногда называют **обучающей выборкой** L. Ее используют для обучения классификатора и определения значения его параметров, при которых классификатор выдает лучший результат.

Далее в системе вырабатываются решающие правила, с помощью которых происходит разделение множества примеров на заданные классы. Качество разделения проверяется **тестовой выборкой** примеров T. При этом необходимо, чтобы выполнялись условия $L \cap T = \emptyset$ и $L \cup T \subset C \times D$, для которых известны значения Ф.

Для пороговой классификации необходимо определить множество пороговых значений, которые вычисляются экспериментально на обучающем наборе. Документ относится в ту или иную категорию в зависимости от того, больше или меньше порогового значения принимает функция. Множество пороговых значений позволяет рассматривать вещественные значения CSV как двоичные, т. е. пороговую классификацию можно свести к точной.

Если в задаче каждому документу $d \in D$ может соответствовать только одна категория $c \in C$, то имеет место **однозначная классификация**, а если произвольное количество категорий, то **многозначная классификация**.

Частным случаем однозначной классификации является **бинарная классификация**, когда нужно разбить коллекцию документов на две непересекающиеся категории. Например, задача определения тональности высказываний (положительная или отрицательная окраска) или задача обнаружения спама (является сообщение спамом или нет) решается при помощи бинарного классификатора.

Решение задачи классификации состоит из трех последовательных этапов.

1. Предобработка и индексация документов.
2. Уменьшение размерности пространства признаков.
3. Построение и обучение классификатора с помощью методов машинного обучения (метод наименьших квадратов; метод *k-NN* (метод ближайших соседей); метод опорных векторов; наивный Байесовский классификатор и др.).
4. Оценка качества классификации.

1. Предобработка и индексация документов.

Предварительная обработка текста включает в себя токенизацию, удаление функциональных слов (семантически нейтральных слов, таких как союзы, предлоги, артикли и пр.). Далее осуществляется морфологический анализ (производится разметка по частям речи и стемматизация). Это позволяет значительно сократить размерность пространства. В результате в качестве **признаков документа** выступают все значимые слова, встречающиеся в документе. В зависимости от приложения может индексироваться как весь текст, так и отдельные его части: заголовок, аннотация, ключевые слова или фразы, первые несколько строк раздела и др. Этот подход

возможен при наличии знаний о структуре классифицируемого документа. В общем случае, когда о структуре документа ничего не известно, определение наиболее важных разделов – довольно неочевидная задача.

Индексация документов – построение некоторой числовой модели текста. Она переводит текст в удобное для дальнейшей обработки представление.

Выбор представления текста зависит от того, что считать значимыми текстовыми единицами (проблема лексической семантики) и значимыми правилами естественного языка для комбинации этих единиц (проблема композиционной семантики).

Например, модель «мешка слов» (bag-of-words) позволяет представить документ в виде многомерного вектора слов и их весов в документе. Другими словами, каждый документ – это вектор в многомерном пространстве, координаты соответствуют номерам слов, значения координат – значения весов $d = (t_1,...,t_n)$. Другая распространенная модель индексации Word2vec преобразует каждое слово к вектору, который содержит информацию о контекстных (сопутствующих) словах. Еще одна модель индексации основана на учете *n*-грамм, т. е. последовательностей из соседних символов.

Очевидно, что для обучающих и тестовых документов должен применяться один и тот же метод индексации.

2. Уменьшение размерности пространства признаков.

Функция взвешивания позволяет оценить вес признака, определить, насколько термин является значимым, или, что то же самое, обнаружить ключевые слова в рассматриваемом документе. Веса терминов обычно нормализуют, чтобы они варьировались от 0 до 1. Частным случаем функции взвешивания является бинарная функция, принимающая значение 1 при наличии термина в документе, и 0 – при его отсутствии.

Следует отметить, что в общем случае для некоторых методов классификации выбор функции взвешивания не является обязательным. Например, Байесовский классификатор сам вычисляет вероятности для признаков.

Существует много методов выбора оптимальных признаков. Наиболее распространенным среди них является вычисление стандартной функции TF-IDF. Строится она следующим образом.

1. Вычисляется ***TF*** (*term frequency*) **частота термина** – оценка важности слова t в пределах одного документа d.

$$TF = \frac{C_{t,d}}{C_d}, \text{ где}$$

$C_{t,d}$ – сколько раз слово t встречается в документе d;

C_d – общее число слов в документе d.

2. Вычисляется ***IDF*** (*inverse document frequency*) **обратная частота документа** – инверсия частоты, с которой слово t встречается в документах коллекции. IDF уменьшает вес общеупотребительных слов.

$$ITF = \log \frac{|D|}{D_t}, \text{ где}$$

$|D|$ – общее количество документов в коллекции;

D_t – количество всех документов, в которых встречается слово t.

3. Итоговый вес термина t в документе d относительно всей коллекции документов вычисляется по формуле:

$$V_{t,d} = TF \cdot IDF .$$

Таким образом, большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

Пример

Документ содержит 100 слов ($C_d = 100$). Слово «музыка» встречается в нем три раза ($C_{t,d} = 3$). Частота слова «музыка» в документе $TF = 3/100 = 0{,}03$. В 1000 документов встречается слово «музыка» ($D_t = 1000$). Всего в коллекции 100000 документов ($|D| = 100000$). Обратная частота документа $IDF = \log 100 = 2$. Вес слова «музыка» в выбранном документе $V_{t,d} = TF \cdot IDF = 0{,}03 \cdot 2 = 0{,}06$.

Функция TF-IDF заключает в себе, с одной стороны, идею того, что чем чаще термин встречается в документе, тем лучше он отражает его содержание, с другой стороны, чем в большем количестве документов встречается термин, тем менее значимым он является для классификации. Отметим, что эта формула оценивает значимость термина только с точки зрения частоты вхождения в документ, тем самым не учитывая порядок следования терминов в документе и их синтаксическую роль. Другими словами, семантика документа сводится к лексической семантике входящих в него терминов, а композиционная семантика не рассматривается.

Вычислительная сложность различных методов классификации (категоризации) напрямую зависит от размерности пространства признаков. Поэтому для эффективной работы классификатора часто прибегают к сокращению числа используемых признаков (терминов). Ведь количество терминов в корпусе может достигать десятков тысяч. Если типичный алгоритм в поисковых системах (такой, как совпадение по косинусу) легко

справляется с большой размерностью, то для более сложных обучающихся алгоритмов индуктивного построения классификатора это большая проблема. В этих случаях часто применяются методики сокращения размерности, сужающие векторное пространство до размерности $|V'| \ll |V|$. В основу этой процедуры положен закон Хипса, который связывает рост количества обрабатываемых документов (или объем текста) с ростом количества терминов V в документе.

За счет уменьшения размерности пространства терминов можно снизить эффект переобучения – явление, при котором классификатор ориентируется на случайные или ошибочные характеристики обучающих данных, а не на важные и значимые. Переобученный классификатор хорошо работает на тех экземплярах, на которых он обучался, и значительно хуже на тестовых данных. Для того чтобы избежать переобучения, количество обучающих примеров должно быть соразмерно числу используемых терминов. В некоторых случаях сокращение размерности пространства признаков в 10 раз (и даже в 100) может приводить лишь к незначительному ухудшению работы классификатора.

3. Построение и обучение классификатора с помощью методов машинного обучения

Существует много литературы, например [2], с подробным описанием методов классификации. Здесь мы ограничимся кратким изложением пяти наиболее распространенных методов классификации, перечислим преимущества и недостатки этих методов. Рассмотрим:

- метод наименьших квадратов (метод регрессионного анализа);
- метод ближайших соседей (метрический метод классификации);
- метод опорных векторов (линейный метод классификации);
- метод Байеса (вероятностный метод классификации);
- метод деревьев решений (логический метод классификации).

Метод наименьших квадратов (Least square method)

Метод наименьших квадратов относится к методам регрессионного анализа, а точнее – восстановительной регрессии. С каждым документом d связывают два вектора: входной вектор весов признаков $I(d)$ размерности $|V'| = n$ и выходной вектор весов категорий $O(d)$ размерности $|C|$.

Задача классификации сводится к определению вектора $O(d)$ по вектору $I(d)$:

$$MI(d) = O(d), \text{ где}$$

M – матрица размера $|C|\times|V'|$.

Другими словами, задача регрессионного анализа состоит в восстановлении функциональной зависимости O по результатам измерений I, т. е. M – искомая функция. Но поскольку точно найти мы ее не можем, мы ищем приближенное решение $\hat{M}$. Параметры (элементы матрицы $\hat{M}$) следует выбирать таким образом, чтобы отклонение значений предложенной функции от результатов эксперимента было минимальным. Поэтому построение классификатора заключается в нахождении матрицы $\hat{M}$, минимизирующей норму $\left\|\hat{M}I - O\right\|_F$, где

I – матрица размера $|V'|\times|L|$, столбцы которой являются векторами $I(d)$;

O – матрица размера $|C|\times|L|$, столбцы которой являются векторами $O(d)$;

$\|X\|_F$ – норма Фробениуса для матрицы размера $n\times m$, которая вычисляется по формуле: $\|X\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2}$.

В результате матрицу $\hat{M}$ получаем через сингулярное разложение матрицы, построенной на обучающем множестве.

Преимущества метода
- Программная реализация алгоритма относительно проста.
- Результаты работы алгоритма легко поддаются интерпретации.

Недостатки метода
- Неустойчивость алгоритма по отношению к выбросам в исходных данных.
- Требуется большой объем данных для получения точных результатов.

Метод k ближайших соседей (k Nearest Neighbors, k-NN)

Метод k ближайших соседей относится к метрическим методам классификации. Для того чтобы найти категорию, соответствующую документу d, классификатор сравнивает d со всеми документами из обучающей выборки L, т. е. для каждого $d_z \in L$ вычисляется расстояние $\rho(d_z, d)$. Далее из обучающей выборки выбираются k документов, ближайших к d. Согласно методу k ближайших соседей документ d считается принад-

лежащим тому классу, который является наиболее распространенным среди соседей данного документа, т. е. для каждого класса c_i вычисляется функция ранжирования:

$$CSV_i(d) = \sum_{d_z \in L_k(d)} \rho(d_z, d) \cdot \Phi(d_z, c_i), \text{ где}$$

$L_k(d)$ – это ближайшие k документов из L к d;

$\Phi(d_z, c_i)$ – это известные величины, уже расклассифицированные по категориям документы обучающей выборки.

Преимущества метода

- Не нужно строить классифицирующую функцию, а значит, есть возможность обновлять обучающую выборку без переобучения классификатора.
- Алгоритм устойчив к аномальным выбросам в исходных данных.
- Программная реализация алгоритма относительно проста.
- Результаты работы алгоритма легко поддаются интерпретации.
- Хорошо справляется с линейно неразделимыми выборками (о линейной неразделимости см. ниже).

Недостатки метода

- Набор данных, используемый для алгоритма, должен быть репрезентативным (показательным).
- Существует высокая зависимость результатов классификации от выбранной метрики.
- Большая длительность работы, т. к. возникает необходимость полного перебора обучающей выборки.
- Не подходит для решения задач большой размерности по количеству классов и документов.

Метод опорных векторов (Support Vector Machine, SVM)

Метод опорных векторов является линейным методом классификации. В настоящее время этот метод считается одним из лучших методов классификации. Будем считать множество документов, которые необходимо расклассифицировать, множеством точек в пространстве размерностью $|D|$.

Выборку точек называют **линейно разделимой**, если точки, принадлежащие разным классам, можно разделить с помощью гиперплоскости (в двумерном случае гиперплоскостью является прямая линия). Очевидный способ решения задачи в таком случае – провести прямую так, чтобы все

точки одного класса лежали по одну сторону от этой прямой, а все точки другого класса были на противоположной стороне. Тогда чтобы классифицировать неизвестные точки, достаточно будет посмотреть, с какой стороны прямой они окажутся.

В общем случае можно провести бесконечное множество гиперплоскостей (прямых), удовлетворяющих нашему условию. Ясно, что лучше всего выбрать прямую, максимально удаленную от имеющихся точек. В методе опорных векторов расстоянием между прямой и множеством точек считается расстояние между прямой и ближайшей к ней точкой из множества. Именно такое расстояние и максимизируется в данном методе. Гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей, называется **разделяющей** (на рис. 25 обозначена буквой L). Ближайшие к параллельным гиперплоскостям точки называются **опорными векторами** (на рис. 25 через них проходят пунктирные линии). Другими словами, алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора (рис. 25), т. к. максимизация зазора между классами способствует более уверенной классификации.

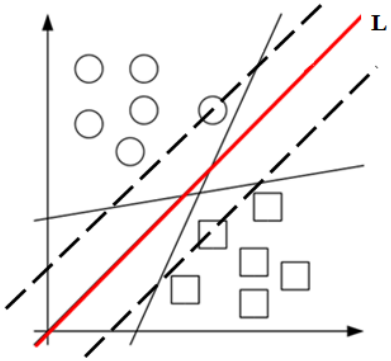


Рис. 25. Разделяющая гиперплоскость в методе опорных векторов

Итак, формальная постановка задачи выглядит следующим образом. Имеется множество документов $D=\{d_1,...,d_{|D|}\}$, где каждый документ представлен в виде вектора признаков $d=(t_1,...,t_n)$. Чтобы разбить множество D на два класса $C=\{-1,1\}$, требуется по обучающей выборке $L=(d_j,c_i)$, $j=1,...,|L|$ найти оптимальные параметры алгоритма классификации:

$$sign\left(\sum_{k=1}^{n} w_k t_k - w_0\right) = sign\left(\langle w, d\rangle - w_0\right), \text{ где}$$

пороговая величина w_0 и вектор $w = (w_1,\ldots,w_n)$ являются искомыми параметрами, т. к. уравнение $\langle w, d\rangle = w_0$ описывает искомую гиперплоскость, разделяющую классы. Напомним еще, что функция сигнум в общем случае определяется следующим образом:

$$sign(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}.$$

На практике зачастую структура данных бывает неизвестна, и очень редко удается построить разделяющую гиперплоскость, а значит, невозможно гарантировать линейную разделимость выборки. Могут существовать такие документы, которые алгоритм отнесет к одному классу, а в действительности они должны относиться к противоположному. Такие данные называются **выбросами**, они создают погрешность метода, поэтому было бы лучше их игнорировать. В этом заключается суть проблемы линейной неразделимости.

Выборку называют **линейно неразделимой**, если точки, принадлежащие разным классам, нельзя разделить с помощью гиперплоскости. Когда такой разделяющей гиперплоскости не существует, осуществляют переход от исходного пространства признаков документов к новому, в котором обучающая выборка окажется линейно разделимой. Для этого каждое скалярное произведение в вышеприведенной формуле заменяют на некоторую функцию, отвечающую определенным требованиям. Например, можно назначать некий штраф за каждый неверно расклассифицированный документ. Эту функцию называют **ядром**. Замена скалярного произведения функцией-ядром позволяет перейти к другому пространству признаков, где данные уже будут разделимы.

В случае линейной неразделимости проблема поиска оптимальной разделяющей гиперплоскости сводится к задаче, эквивалентной поиску седловой точки функции Лагранжа с условиями дополняющей нежесткости. Полученная система уравнений решается методами квадратичного программирования. Это уже чисто вычислительная задача. Этот вариант алгоритма называют **алгоритмом с мягким зазором (soft-margin SVM)**, тогда как в линейно разделимом случае говорят о **жестком зазоре (hard-margin SVM).**

Преимущества метода

– Является одним из наиболее качественных.

– Достаточно небольшого набора данных для обучения.

Недостатки метода
– Скорость обучения одна из самых низких.
– Параметры алгоритма сложно интерпретировать.
– Неустойчивость по отношению к выбросам в исходных данных.

Метод Байеса (Naive Bayes, NB)

Метод Байеса относится к вероятностным методам классификации.

Пусть $P(c_i \mid d)$ – вероятность того, что документ, представленный вектором $d = (t_1,\ldots,t_n)$, соответствует категории c_i для $i = 1,\ldots,|C|$. Задача классификатора заключается в том, чтобы подобрать такие значения c_i и d, при которых значение вероятности $P(c_i \mid d)$ будет максимальным:

$$CSV_i(d) = \arg\max_{c_i \in C} P(c_i \mid d).$$

Для вычиления значений $P(c_i \mid d)$ воспользуемся **теоремой Байеса**:

$$P(c_i \mid d) = \frac{P(c_i)P(d \mid c_i)}{P(d)}, \text{ где}$$

$P(c_i)$ – априорная вероятность того, что документ отнесен к категории c_i;

$P(d \mid c_i)$ – вероятность найти документ, представленный вектором $d = (t_1,\ldots,t_n)$, в категории c_i;

$P(d)$ – вероятность того, что произвольно взятый документ можно представить в виде вектора признаков $d = (t_1,\ldots,t_n)$.

По сути $P(c_i)$ является отношением количества документов из обучающей выборки L, отнесенных в категорию c_i, к количеству всех документов из L.

$P(d)$ не зависит от категории c_i, а значения $t_1,\ldots,t_n$ заданы у нас заранее, поэтому знаменатель – это константа, не влияющая на выбор наибольшего из значений $P(c_i \mid d)$.

Вычисление $P(d \mid c_i)$ затруднительно из-за большого количества признаков $t_1,\ldots,t_n$, поэтому делают «наивное» предположение о том, что любые две координаты, рассматриваемые как случайные величины, статисти-

чески независимы друг от друга. Тогда можно воспользоваться формулой $P(d \mid c_i) = \prod_{k=1}^{n} P(t_k \mid c_i)$.

Далее все вероятности подсчитываются по методу максимального правдоподобия.

Преимущества метода

- Высокая скорость работы (удобен, когда накладываются жесткие ограничения на время выполнения классификации, а возможности воспользоваться более точными методами нет).
- Поддержка инкрементного обучения (классификатор обучается на каждом отдельно взятом образце, нет необходимости предъявлять сразу всю обучающую выборку целиком).
- Программная реализация алгоритма относительно проста.
- Результаты работы алгоритма легко поддаются интерпретации (поскольку вероятности всех признаков сохраняются, можно в любой момент посмотреть, какие признаки документов являются оптимальными для качественной классификации).

Недостатки метода

- Относительно низкое качество классификации.
- Неспособность учитывать зависимость результата классификации от сочетания признаков.

Метод деревьев решений (Decision Trees, DT)

Метод деревьев решений относится к логическим методам классификации.

Деревом решений называют ациклический граф, по которому производится классификация объектов (в нашем случае текстовых документов), описанных набором признаков. Каждый узел дерева содержит условие ветвления по одному из признаков. Каждый узел имеет столько ветвлений, сколько значений имеет выбранный признак. В процессе классификации осуществляются последовательные переходы от одного узла к другому в соответствии со значениями признаков объекта. Классификация считается завершенной, когда достигнут один из листьев (конечных узлов) дерева. Значение этого листа определит класс, которому принадлежит рассматриваемый объект. На практике обычно используют **бинарные деревья решений**, в которых принятие решения перехода по ребрам осуществляется простой проверкой наличия признака в документе. Если значение признака меньше определенного значения, то выбирается одна ветвь, если больше или равно, – другая.

В отличие от остальных подходов, представленных ранее, подход, использующий деревья решений, относится к символьным (т. е. нечисловым) алгоритмам.

Алгоритм построения бинарного дерева решений схематично изображен на рис. 26 и состоит из следующих шагов.

1. Создается первый узел дерева, в который входят все документы, представленные всеми имеющимися признаками. Размер вектора признаков для каждого документа равен n, т. к. $d = (t_1,...,t_n)$.

2. Для текущего узла дерева выбирается наиболее подходящий признак t_k и его наилучшее пограничное значение v_k.

3. На основе пограничного значения выбранного признака производится разделение обучающей выборки на две части. Далее выбранный признак не включается в описание фрагментов в этих частях, т. е. фрагменты в частях представляются вектором с размерностью $n-1$.

4. Образовавшиеся подмножества обрабатываются аналогично до тех пор, пока в каждом из них не останутся только документы одного класса или не останется признаков для различения документов.

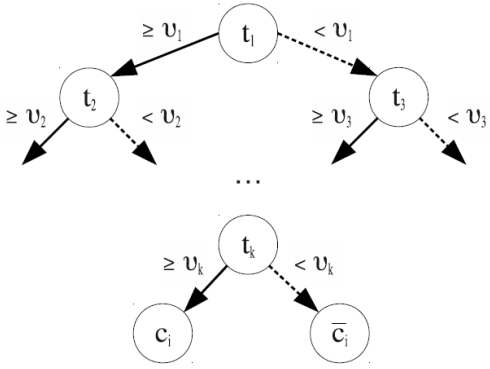


Рис. 26. Бинарное дерево решений

На рисунке приняты следующие условные обозначения: узлы – t_k признаки документа; ребра – условия, накладываемые на значения признаков $\geq v_k$ или $< v_k$; листья (конечные узлы дерева) – категории c_i и $\bar{c}_i$.

Чтобы определить, к какой из категорий – c_i или $\bar{c}_i$ – отнести документ, необходимо пройти по узлам дерева, начиная с корня, и сравнить веса признаков в документе с пороговыми значениями v_k на ребрах.

Когда говорят о выборе наиболее подходящего признака (п. 2 алгоритма), как правило подразумевают частотный признак, т. е. любой признак текста, допускающий возможность нахождения частоты его появления в тексте. Лучшим для разделения является признак, дающий максимальную на данном шаге информацию о классах. Таким признаком для текста может являться, например, ключевое слово. С этой точки зрения любой частотный признак можно считать переменной. Тогда выбор между двумя наиболее подходящими признаками сводится к оценке степени связанности двух переменных. Поэтому для выбора подходящего признака на практике применяют различные критерии проверки гипотез, т. е. критерии количественной оценки степени связанности двух переменных, поставленных во взаимное соответствие: 0 соответствует полной независимости переменных, а 1 – их максимальной зависимости.

Для исследования связи между двумя переменными удобно использовать представление совместного распределения этих переменных в виде **таблицы сопряженности (факторной таблицы**, или **матрицы частот появления признаков**). Таблица сопряженности является наиболее универсальным средством изучения статистических связей, т. к. в ней могут быть представлены переменные с любым уровнем измерения. Таблицы сопряженности часто используются для проверки гипотезы о наличии связи между двумя признаками с использованием различных статистических критериев: критерия Фишера (точного теста Фишера), критерия согласия Пирсона, критерия Крамера, критерия Стьюдента и пр.

Строки таблицы сопряженности соответствуют значениям одной переменной, столбцы – значениям другой переменной, при этом количественные шкалы предварительно должны быть сгруппированы в интервалы. Ниже представлен пример таблицы сопряженности, иллюстрирующей зависимость предпочтений профессиональной принадлежности от места проживания.

Таблица 9

Таблица сопряженности совместного распределения двух переменных

	Учитель	**Врач**	**Программист**	**Всего**
Город	8	12	30	50
Деревня	17	28	5	50
Всего	25	40	35	100

	y_1	y_2	y_3	
x_1	f_{11}	f_{12}	f_{13}	$f_{1.}$
x_2	f_{21}	f_{22}	f_{23}	$f_{2.}$
	$f_{.1}$	$f_{.2}$	$f_{.3}$	n

На пересечении строки и столбца указывается частота совместного появления f_{ij} соответствующих значений двух признаков x_i и y_j. Сумма частот по строке $f_i = \sum_j f_{ij}$ называется **маргинальной частотой строки**;

сумма частот по столбцу $f_j = \sum_i f_{ij}$ называется **маргинальной частотой столбца**. Сумма маргинальных частот равна объему выборки n; их распределение представляет собой одномерное распределение переменной, образующей строки или столбцы таблицы. В таблице сопряженности могут быть представлены как абсолютные, так и относительные частоты (в долях или процентах). Относительные частоты могут рассчитываться по отношению: к маргинальной частоте по строке; к маргинальной частоте по столбцу; к объему выборки.

При построении деревьев решений на этапе выбора наиболее подходящего признака могут использоваться различные методы статистической проверки гипотез: критерий хи-квадрат, критерий Фишера, критерий Пирсона, критерий Крамера, критерий Стьюдента и другие. Приведем наиболее распространенные из них.

Критерий хи-квадрат – метод статистической проверки гипотезы, в которой выборочное распределение критерия имеет распределение хи-квадрат, если нулевая гипотеза верна.

$$\chi^2 = \frac{(f_o - f_e)^2}{f_e}, \text{ где}$$

f_0 – фактические (наблюдаемые) частоты событий;

f_e – ожидаемые частоты событий.

Некоторые примеры критериев хи-квадрат имеют распределение хи-квадрат только в приближении (например, критерий Фишера, критерий Пирсона).

Критерий Фишера (точный тест Фишера)

$$\varphi = \sqrt{\frac{\chi^2}{n}}, \text{ где}$$

n – общая сумма частот в таблице сопряженности.

Как следует из названия, тест Фишера является точным, и поэтому может использоваться независимо от особенностей выборки. Тест становится трудновычислимым для больших выборок или хорошо уравновешенных таблиц сопряженности. Для таких условий хорошо подходит критерий Пирсона.

Критерий Пирсона (критерий согласия Пирсона)

$$c = \sqrt{\frac{\chi^2}{\chi^2 + n}}\ .$$

Критерий Крамера

$$V = \sqrt{\frac{\chi^2}{n\,(k-1)}}\ , \text{ где}$$

k – наименьшее из количества строк и столбцов матрицы сопряженности.

Критерий Стьюдента (*t*-критерий Стьюдента) – метод статистической проверки гипотез, основанный на распределении Стьюдента. Этот критерий часто применяется для проверки равенства средних значений в двух выборках. Для применения данного критерия необходимо, чтобы исходные данные имели нормальное распределение.

Для построения *t*-статистики обычно пользуются следующим принципом: в числителе должна стоять случайная величина с нулевым математическим ожиданием (при выполнении нулевой гипотезы), а в знаменателе – выборочное стандартное отклонение этой случайной величины, получаемое как квадратный корень из несмещенной оценки дисперсии. Для случая независимых выборок статистика критерия равна:

$$t = \frac{\bar{x} - \bar{y}}{\sigma_{x-y}}\ , \text{ где}$$

$\bar{x} = \sum_{i=1}^{n_1} x_i$ – среднее арифметическое в экспериментальной выборке;

n_1 – размер первой выборки;

$\bar{y} = \sum_{i=1}^{n_2} y_i$ – среднее арифметическое в контрольной выборке;

n_2 – размер второй выборки;

σ_{x-y} – стандартное отклонение разностей средних арифметических вычисляется по формуле:

$$\sigma_{x-y} = \sqrt{\frac{\sum (x_i - \bar{x})^2 + \sum (y_i - \bar{y})^2}{n_1 + n_2 - 2} \cdot \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}\ .$$

Пример

Экспериментальная выборка ($n_1 = 11$)	**Контрольная выборка ($n_2 = 9$)**
12 14 13 16 11 9 13 15 15 18 14	13 9 11 10 7 6 8 10 11

Общее количество членов выборки $n_1 = 11$, $n_2 = 9$.

Средние арифметические: $\bar{x} = 13{,}636$; $\bar{y} = 9{,}444$.

Стандартное отклонение: $\sigma_{x-y} = \sqrt{\frac{60{,}545 + 38{,}222}{11+9-2} \cdot \left(\frac{1}{11} + \frac{1}{9}\right)} = 1{,}053$.

Статистика критерия Стьюдента $t = \frac{13{,}636 - 9{,}444}{1{,}053} = 3{,}981$.

Если полученное в эксперименте эмпирическое значение *t* превышает ранее установленное критическое значение, тогда есть основания принять альтернативную гипотезу о том, что на экспериментальной выборке результат лучше.

Преимущества метода
- Программная реализация алгоритма относительно проста.
- Результаты работы алгоритма легко поддаются интерпретации.

Недостатки метода
- Неустойчивость алгоритма по отношению к выбросам в исходных данных.
- Требуется большой объем данных для получения точных результатов.

4. Оценка качества классификации.

Основным критерием при оценке качества классификации является комбинация точности и полноты.

Точность (precision) классификации в пределах класса – это доля найденных классификатором документов, действительно принадлежащих данному классу, относительно всех документов, которые система отнесла к этому классу.

Полнота (recall) классификации – это доля найденных классификатором документов, действительно принадлежащих классу, относительно всех документов этого класса в тестовой выборке.

Оценка качества работы классификатора производится на тестовой выборке. Вместе с тем работу системы оценивает эксперт (таблица 10).

Таблица 10

Оценка качества классификации

Класс c_i		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

В таблице приняты следующие условные обозначения:
TP – истинно положительное решение;
TN – истинно отрицательное решение;
FP – ложно положительное решение;
FN – ложно отрицательное решение.
Теперь **точность** определяется следующим образом:

$$p = \frac{TP}{TP + FP}.$$

Полнота вычисляется по формуле:

$$r = \frac{TP}{TP + FN}.$$

Пример

Пусть тестовая выборка состоит из 12 сообщений. Из них 5 являются спам-сообщениями. Обработав все сообщения, классификатор пометил 3 как спам, причем 2 действительно являются спамом, а третье было помечено в тестовой выборке как нормальное, т. е. $TP = 2$, $FP = 1$, $FN = 3$.

Тогда точность классификатора для класса «спам» составляет $p = \frac{TP}{TP + FP} = \frac{2}{2+1} = 0{,}67$ (67 % положительных решений правильные); полнота $r = \frac{TP}{TP + FN} = \frac{2}{2+3} = 0{,}4$ (классификатор нашел 40 % всех спам-сообщений).

F-мера – характеристика качества работы алгоритма, которая объединяет в себе информацию о точности и полноте:

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r}, \text{ где } 0 \le \beta < \infty.$$

При $0 \le \beta < 1$ большее значение имеет точность.

При $\beta = 1$ точность и полнота равноправны, тогда

$$F_\beta = \frac{2pr}{p + r}.$$

При $1 < \beta < \infty$ большее значение имеет полнота.

Пример

Подставим в формулу F-меры точность и полноту из предыдущего примера: $p = 0{,}67$ и $r = 0{,}4$ при $\beta = 0{,}2$ (т. к. для обнаружения спама приоритетной является точность). Получим

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r} = \frac{0{,}279}{0{,}429} = 0{,}65 .$$

Сравнение методов построения классификаторов является довольно сложной задачей по причине того, что разные входные данные могут приводить к различным результатам. Чтобы провести сравнение различных классификаторов, необходимо выполнить их построение и вычисление эффективности на одинаковых наборах документов для обучения и тестирования.

Для подобных экспериментов часто используют коллекцию документов «Reuters-21578». Она была собрана и размечена в 2004 году Д. Льюисом (доступна по ссылке http://www.daviddlewis.com/). Коллекция содержит 21578 документов из ленты новостей Reuters. Обучающая выборка состоит из 9603 документов. Тестовая выборка включает в себя 3299 документов.

7.2. Кластеризация текстов, машинное обучение без учителя

Формальная постановка задачи кластеризации записывается следующим образом. Имеется множество документов $D = \{d_1, \ldots, d_{|D|}\}$. Задана функция расстояния между объектами $\rho(d_m, d_n), \quad m \neq n, \quad 1 \le m, n \le |D|$. Имеется конечная обучающая выборка документов $L \subset D$, каждому объекту которой приписывается номер кластера. **Алгоритм кластеризации** позволяет построить функцию $\Psi : D \to C$, которая каждому документу $d_j \in D$ ставит в соответствие кластер $c_i \in C$.

Часто ставится задача определить оптимальное число кластеров, т. е. c_i изначально никак не заданы, и неизвестно само множество C. Поэтому в качестве критерия выделения кластеров принято использовать **гипотезу компактности**, согласно которой расстояние между разными кластерами должно быть существенно больше, чем среднее расстояние между объектами внутри одного и того же кластера. Другими словами, задача кластеризации состоит в том, чтобы разбить предъявленную выборку на непересекающиеся подмножества (**кластеры**) так, чтобы каждый кластер состоял из объектов, близких по некоторой метрике, а объекты разных кластеров существенно отличались.

Задачу кластеризации невозможно решить однозначно по нескольким причинам [2]. Во-первых, не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих четко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты. Следовательно, для определения качества кластеризации требуется эксперт предметной области, который бы мог оценить осмысленность выделения кластеров.

Во-вторых, число кластеров, как правило, неизвестно заранее, поэтому определяют его в соответствии с некоторым субъективным критерием.

В-третьих, результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом. Но стоит отметить, что есть ряд рекомендаций к выбору мер близости для различных задач.

Еще раз повторим, что при кластеризации требуется соблюдать баланс. С одной стороны, необходимо выбрать такое разделение документов по кластерам, при котором элементы каждой группы были бы настолько сходны друг с другом, чтобы в некоторых случаях можно было пренебречь их индивидуальными особенностями. С другой стороны, важно прийти к разумному компромиссу относительно размера групп, избегая как формирования большого числа очень мелких кластеров, так и небольшого количества очень крупных кластеров.

Для решения задачи кластеризации применяют методы обучения без учителя, когда система обучается выполнять поставленную задачу без постороннего вмешательства, и не требуется предоставлять ей образцы предполагаемого разбиения множества документов на группы. Основные из этих методов:

- иерархические методы (метод одиночной связи, метод средней связи и др.);
- центроидные методы (метод *k*-средних, FRiS-метод, алгоритмы семейства FOREL и др.);
- вероятностные методы (EM-алгоритм и др.);
- методы на основе систем искусственного интеллекта (метод нечеткой кластеризации С-средних, нейронные сети, генетические алгоритмы и др.).

Рассмотрим подробнее иерархические и центроидные методы. Детальное изучение остальных опустим из-за ограниченности объема пособия.

Иерархические методы

Результат работы иерархических алгоритмов можно представить в виде деревообразной иерархической структуры, называемой **дендрограммой**. Пример дендрограммы приведен на рис. 27.

Алгоритмы иерархической кластеризации подразделяются на агломеративные (объединительные) и дивизимные (разделяющие). По своему принципу агломеративные и дивизимные алгоритмы отдаленно напоминают процедуру восходящего и нисходящего синтаксического анализа соответственно, рассмотренную в главе 2.

Агломеративная процедура состоит из нескольких шагов.

1. Перед началом работы алгоритма рассчитывается матрица расстояний между объектами.
2. Первоначально каждый объект представляет собой отдельный кластер.
3. На каждом шаге в матрице расстояний ищется минимальное значение, соответствующее расстоянию между двумя наиболее близкими кластерами.
4. Найденные кластеры объединяются, образуя новый кластер.
5. Эта процедура повторяется до тех пор, пока не будут объединены все кластеры.

Заметим, что особое внимание следует уделять выбору меры расстояния между объектами. Как видно, на основе нее формируется начальная матрица расстояний, которая и определяет весь дальнейший процесс кластеризации. Расчет матрицы расстояний между объектами внутри кластера может осуществляться при помощи большинства метрик, приведенных в предыдущей главе (Евклидова метрика, Хеммингово расстояние, метрика Чебышева, Минковского и др.).

Для остановки агломеративной процедуры применяется один из следующих критериев:

- получено определенное заранее количество кластеров;
- все кластеры содержат более определенного числа элементов;
- кластеры обладают требуемым соотношением внутренней однородности и разнородности между собой.

Существуют различные методы построения дендрограмм, различающиеся способом вычисления расстояния между кластерами.

1. Метод одиночной связи (Single Linkage), также известен как «метод ближайшего соседа».
2. Метод полной связи (Complete Linkage), также известен как «метод дальнего соседа».
3. Метод попарного среднего (Pair-Group Method with Arithmetic mean, PGMA): невзвешенный (unweighted) и взвешенный (weighted).

4. Центроидный метод (Pair-Group Method with Centroid, PGMC): невзвешенный (unweighted) и взвешенный (weighted).

5. Метод Варда (Ward's method), также известен как метод Уорда.

У каждого метода есть свои особенности. Например, в **методе одиночной связи (методе ближайшего соседа)** расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Это правило позволяет как бы «нанизывать» объекты для формирования кластеров, и результирующие кластеры имеют тенденцию быть представленными длинными «цепочками». На рис. 27 представлен пример пересчета матрицы расстояний и дендрограмма, полученная методом одиночной связи

	1	2	3	4
1	0	2.06	4.03	6.32
2	2.06	0	2.50	4.12
3	4.03	2.50	0	2.24
4	6.32	4.12	2.24	0

	1, 2	3	4
1, 2	0	2.50	4.12
3	2.50	0	2.24
4	4.12	2.24	0

	1, 2	3, 4
1, 2	0	2.50
3, 4	2.50	0

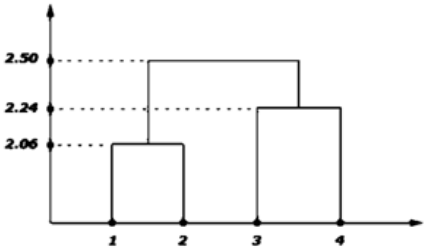


Рис. 27. Последовательный пересчет матрицы расстояний в методе одиночной связи

Метод полной связи (метод дальнего соседа) обычно работает очень хорошо, когда объекты принадлежат сильно различающимся кластерам. Если же кластеры имеют удлиненную форму или их естественный тип является «цепочечным», то этот метод непригоден.

В **методе невзвешенного попарного среднего** расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них. Метод эффективен, когда объекты в действительности сильно отличаются, однако он работает одинаково хорошо и в случаях протяженных («цепочного» типа) кластеров.

Метод взвешенного попарного среднего идентичен методу невзвешенного попарного среднего, за исключением того, что при вычислениях размер соответствующих кластеров (т. е. число объектов, содержащихся в

них) используется в качестве весового коэффициента. Поэтому предлагаемый метод должен быть использован (скорее даже, чем предыдущий), когда предполагаются неравные размеры кластеров.

Метод Варда (метод Уорда) чаще других методов восстанавливает интуитивно наилучшую кластеризацию.

Из вышесказанного видно, что невозможно однозначно ответить, какой из алгоритмов кластеризации лучше, многое зависит от конкретной задачи.

Существует универсальная **формула Ланса-Уильямса (Lance, Williams, 1967)** для вычисления расстояний между кластерами в иерархических методах. На каждый итерации вместо пары самых близких кластеров c_U и c_V образуется новый кластер $c_U \cup c_V$.

Пусть уже известны ρ_{UV}, ρ_{UW}, ρ_{VW} – попарные расстояния между кластерами c_U, c_V и c_W соответственно.

Тогда $\rho_{(UV)W}$ – расстояние от нового кластера $c_U \cup c_V$ до любого другого кластера c_W – вычисляется рекурсивно:

$$\rho_{(UV)W} = \alpha_U \rho_{UW} + \alpha_V \rho_{VW} + \beta \rho_{UV} + \gamma \,|\, \rho_{UW} - \rho_{VW} \,|,$$

где α_U, α_V, β, γ — числовые параметры.

Например, формулу для метода одиночной связи можно получить, взяв $\alpha_U = \alpha_V = 1/2$, $\beta = 0$, $\gamma = -1/2$, а формулу для метода одиночной связи можно получить, взяв $\alpha_U = \alpha_V = 1/2$, $\beta = 0$, $\gamma = 1/2$.

Центроидные методы

Эти методы основаны на идее объединения в один кластер объектов в областях их наибольшего сгущения. В центроидных методах кластер определяется как совокупность элементов, лежащих на расстоянии не больше R от центра (внутри гиперсферы радиуса r или гиперкуба со сторонами $2R$). В качестве центра выбирается один из элементов кластера, называемый **центроидом**.

Разберем по шагам метод K-средних (K-means), алгоритмы семейства Forel и алгоритм с применением функции конкурентного сходства (Function of Rival Similarity, FRiS), а также перечислим преимущества и недостатки каждого из них.

Элементы, не попавшие ни в один кластер, после определенного числа шагов или образования кластеров с требуемыми показателями считаются нераспознанными и могут трактоваться как шум в системе.

Центроидные методы часто применяются в задачах таксономии, т. е. тогда, когда требуется осуществить кластеризацию, отражающую иерар-

хическую организацию систем объектов. Тогда группы объектов называют **таксонами.**

Метод K-средних (K-means). Алгоритм метода состоит из следующих шагов.

1. Выбираются начальные центроиды для множества документов. Существует несколько способов выбора центроидов [3]. Во-первых, можно случайным образом выбрать k документов, где k равно требуемому числу кластеров. На первом шаге эти документы становятся центроидами кластеров. В этом методе необходимо явно указывать требуемое число кластеров. Во-вторых, количество кластеров и их центроидов может определяться на основе Байесовской оценки множества документов. Для Байесовской оценки неизвестных параметров выборки документов необходима априорная информация о природе и распределениях, имеющих место в предметной области. В случае, когда априорной информации нет, можно исходить из того, что распределение нормальное (Гауссовское) или равномерное. В-третьих, можно воспользоваться эмпирическими оценками числа кластеров и их центроидов. Как правило, эмпирические методы оценок дают наилучшие результаты в конкретной области, однако они не универсальны.
2. Все документы множества распределяются среди кластеров. Документ попадает только в один кластер, центроид которого оказался ближе всего к документу.
3. Пересчитываются центроиды кластеров, исходя из нового множества документов в каждом кластере.
4. Если центроид кластера переместился, то цикл вычислений повторяется с пункта 2. Иначе, если центроид стабилизировался в некоторой окрестности или полностью, процесс кластеризации завершается.

В результате алгоритма получаются непересекающиеся кластеры. Это может являться преимуществом, а может быть и недостатком. Все зависит от постановки задачи.

Преимущества метода

- Линейная скорость работы.
- Метод не нуждается в обучении и при необходимости может накапливать сведения для дальнейшего увеличения точности работы (при использовании Байесовских оценок параметров кластеризации).

Недостатки метода

- Требуется задание количества кластеров по крайней мере на начальных этапах.
- Алгоритм не инкрементен.

– Результат зависит от выбора исходных центроидов кластеров, их оптимальный выбор неизвестен. К примеру, в случае, когда центроиды кластеров выбираются случайным образом, результаты, получаемые над одной и той же выборкой документов, будут отличаться. Это может происходить по причине неудовлетворительной работы генератора случайных чисел и вследствие равномерного распределения (без явных областей сгущения) документов в пространстве.

Принцип работы **алгоритмов семейства FOREL (Follow-the-Regularized-Leader)** [4] можно описать следующим образом.

1. Случайным образом выбирается текущий объект из выборки.

2. Помечаются объекты выборки, находящиеся на расстоянии менее чем R от текущего.

3. Производится пересчет центроидов кластеров. Этот центр тяжести помечается как новый текущий объект.

4. Повторяются шаги 2–3, пока новый текущий объект не совпадет с прежним (центр сферы стабилизируется). Таким образом на каждом шаге сфера сдвигается в сторону локального сгущения объектов выборки, т. е. сферой фиксированного радиуса захватывается все большее количество объектов выборки.

5. Все объекты внутри сферы радиуса R помечаются как кластеризованные и удаляются из выборки.

6. Повторяются шаги 1–5, пока не будет кластеризована вся выборка.

Кластеризуемая выборка может быть задана признаковыми описаниями объектов (векторами в линейном пространстве или матрицей попарных расстояний между объектами в метрическом пространстве). Заметим, что в реальных задачах зачастую хранение всех данных невозможно или бессмысленно, поэтому необходимые данные собираются в процессе кластеризации. Параметр R – радиус поиска локальных сгущений – можно задавать как из априорных соображений (знание о диаметре кластеров), так и настраивать скользящим контролем. В модификациях алгоритма возможно введение параметра k – количества кластеров.

Для алгоритмов семейства FOREL было замечено, что чем меньше R, тем больше кластеров. При повторении итераций возможно уменьшение параметра R для скорейшей сходимости. Наилучших результатов алгоритм достигает на выборках с хорошим выполнением условий компактности. В линейном пространстве центром тяжести может выступать произвольная точка пространства, в метрическом – только объект выборки. В линейном пространстве поиск центра происходит за время $O(n)$, в метрическом – $O(n^2)$. Кроме того, рекомендуется повторная прогонка алгорит-

ма для исключения ситуации некачественной кластеризации по причине неудачного выбора начальных объектов.

Преимущества метода

- Сходимость алгоритма за конечное число шагов.
- Наглядность визуализации кластеризации.
- Возможность операций над центроидами кластеров, т. к. они известны в процессе работы алгоритма.
- Возможность подсчета промежуточных функционалов качества, например, длины цепочки локальных сгущений.
- Возможность проверки гипотез схожести и компактности в процессе работы алгоритма.

Недостатки метода

- Неустойчивость алгоритма (зависимость от выбора начального объекта).
- Необходимость задавать радиус кластера R.
- Произвольное по количеству разбиение на кластеры.
- Относительно низкая производительность.
- Плохая применимость алгоритма при плохой разделимости выборки на кластеры.

В **алгоритме с применением функции конкурентного сходства (Function of Rival Similarity, FRiS)** [5] при определении меры похожести между двумя объектами рассматривается конкурентная ситуация. Сходство объекта z с объектом a оценивается относительно сходства этого же объекта z с объектом b. Иными словами, объект z считается более похожими на a не тогда, когда расстояние $R_1 = R(z,a)$ от него до a мало, а тогда, когда оно меньше расстояния $R_2 = R(z,b)$ до конкурирующего объекта b. В таком случае можно делать вывод о принадлежности z и a одному кластеру.

Для вычисления меры конкурентного сходства, измеренной в абсолютной шкале, вводится нормированная величина

$$F(z,a \mid b) = \frac{R_2 - R_1}{R_2 + R_1}.$$

Эта величина называется функцией конкурентного сходства или FRiS-функцией. Таким образом, **функция конкурентного сходства (FRiS-функция)** – мера сходства двух объектов, исчисляемая относительно некоторого другого объекта. FRiS-функция, в отличие от других существующих мер сходства, позволяет не просто оценивать понятия «далеко» или

«близко», «похож» или «не похож», но также давать количественную оценку ответа на вопрос «по сравнению с чем». Такой подход позволяет учитывать большее число факторов при кластеризации. FRiS-функция хорошо имитирует человеческий механизм восприятия сходства и различия.

Алгоритм с применением функции конкурентного сходства состоит из следующих шагов.

1. Все объекты считаются принадлежащими одному кластеру. Вводится конкурирующее множество из виртуальных объектов, находящихся на расстоянии R^* от остальных объектов. Создается модификация (редукция) FRiS-функции, использующая виртуальный кластер-конкурент $F = \frac{R^* - R_i}{R^* + R_i}$. Центроидом текущего кластера назначается произвольный объект.

2. Для выбранного центроида кластера оцениваются расстояния R_i от всех остальных объектов. Происходит пересчет функции F согласно формуле из определения.

3. Вычисляется сумма значений FRiS-функций F_S для всех объектов текущего кластера. В качестве центроида следующего кластера выбирается объект, набравший наибольшее значение F_S по сравнению с центроидом текущего кластера. При этом для сохранения информации о структуре исходных данных центроиды кластеров располагаются в центрах зон локальных сгущений объектов таким образом, чтобы объекты из каждого кластера были больше похожи на свой центроид, чем на центроиды остальных кластеров.

4. Для продолжения кластеризации шаги 2 и 3 повторяются. Процесс увеличения числа кластеров k останавливается, когда достигнут первый локальный максимум функции F либо получено максимальное количество кластеров, которое задается исходным параметром.

Преимущества метода

- Высокая точность.
- Возможность получения кластеров произвольной формы.
- Автоматическое определение наилучшего числа кластеров.

Недостатки метода

- Необходимость создания редуцированной FRiS-функции на начальном этапе кластеризации, что требует дополнительного задания параметра R^*.

Список литературы

1. Епрев А. С. Исследование влияния разрешения лексической многозначности с помощью контекстных векторов на эффективность категоризации текстовых документов // Автореф. дисс. канд. физ.-мат. наук. Омск, 2011. 19 с.
2. Воронцов К. В. Машинное обучение. Курс лекций. URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_%28курс_лекций%2C_К.В.Воронцов%29.
3. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. Новосибирск : ИМ СО РАН, 1999. 270 с.
4. Максимов Н. В., Голицына О. Л., Тихомиров Г. В., Храмцов П. Б. Информационные ресурсы и поисковые системы : учеб. пособие. М. : МИФИ, 2008. 400 с.
5. Загоруйко Н. Г., Борисова И. А., Кутненко О. А., Дюбанов В. В. Построение сжатого описания данных с использованием функции конкурентного сходства // Сибирский журнал индустриальной математики. 2013. Т. 16. № 1 (53). С. 29–41.
6. Барахнин В. Б., Ткачев Д. А. Оценка эффективности метода параллельной реализации процесса кластеризации электронных документов на основе алгоритма FRiS-Cluster // Труды 13-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» (RCDL-2011), Воронеж, Россия, 2011. с. 184–190.
7. Донской В. И. Алгоритмические модели обучения классификации: обоснование, сравнение, выбор. Симферополь : ДИАЙПИ, 2014. 228 с.

ГЛАВА 8. **ПРИКЛАДНЫЕ ЗАДАЧИ АВТОМАТИЧЕСКОЙ ОБРАБОТКИ ТЕКСТОВ**

8.1. Задача определения авторства текстов

Для определения автора текста зачастую приходится обращаться к экспертам. Эксперты могут идентифицировать автора неизвестного текста или определить принадлежность произведения другому автору при помощи характерных языковых особенностей, стилистических приемов.

Атрибуция текста – исследование текста с целью установления авторства или получения каких-либо сведений об авторе и условиях создания текстового документа.

Выделяют диагностические и идентификационные задачи атрибуции. **Диагностические задачи** позволяют определить личностные характеристики автора (образовательный уровень, родной язык, знание иностранных языков, происхождение, место постоянного проживания и др.) или факт сознательного искажения письменной речи. Диагностические задачи решаются, исходя из предположения, что автор текста неизвестен, а значит, нет образца, с которым можно сопоставить исследуемый текст. **Идентификационные задачи позволяют** подтвердить или исключить авторство определенного лица, проверить текст на плагиат. Такие задачи решаются путем сопоставления исследуемого текста с текстами автора.

Методы атрибуции позволяют исследовать текст на пяти уровнях: пунктуационном, орфографическом, синтаксическом, лексико-фразеологическом и стилистическом.

Пунктуационный уровень выявляет особенности употребления автором знаков препинания, характерные ошибки. Орфографический уровень выявляет характерные ошибки в написании слов. Синтаксический уровень позволяет определить особенности построения предложений, предпочтение тех или иных языковых конструкций, употребление времен, активного или пассивного залога, порядок слов, характерные синтаксические ошибки. Лексико-фразеологический уровень определяет словарный запас автора, особенности использования слов и выражений, склонность к употреблению редких и иностранных слов, диалектизмов, архаизмов, неологизмов, фразеологизмов и т. д. Стилистический уровень позволяет определить жанр, общую структуру текста, некоторые характерные речевые приемы. Под «авторским стилем» обычно понимаются последние три уровня. Анализ именно синтаксического, лексико-фразеологического и стилистического уровней представляет наибольший интерес и наибольшую сложность.

Важно отметить, что задача установления авторства текстов (задача атрибуции) встречается в различных областях и представляет интерес для филологов, литературоведов, историков, юристов, криминалистов. Несомненно, экспертный анализ авторского стиля является трудоемким процессом. Поэтому с развитием компьютерных технологий все больше появляется необходимость в создании формальных методов решения задачи атрибуции, создании различных программных приложений для автоматизации деятельности экспертов.

Рассмотрим подробнее формальные методы определения авторства текстов.

Чаще всего решение задачи атрибуции основано на сравнении вычислимых характеристик текстов. Требуется сравнить два текста – текст с заведомо известным автором (**эталонный текст**) и текст, авторство которого требуется установить, подтвердить или опровергнуть (**спорный текст**). В качестве критерия близости двух текстов вводится так или иначе вычисляемое «расстояние» между соответствующими векторами. В простейшем случае можно представить наборы параметров как обычные векторы в *n*-мерном декартовом пространстве, выходящие из начала координат, и считать расстоянием между текстами обычное декартово расстояние между концами соответствующих им векторов. Есть много других вариантов. В итоге именно «расстояние» является интегральной характеристикой различия текстов. Оно определенным образом нормируется, и тексты, для которых расстояние велико, считаются с высокой вероятностью относящимися к разным авторам. Таким образом, чтобы сопоставить авторство двух текстов, достаточно вычислить для них параметры и определить расстояние. Можно также составить векторы формальных параметров, различающие не конкретных авторов (или их группы), а выделяющие определенные характеристики авторов (например, образовательный уровень). В общем случае задачу атрибуции можно представить в виде схемы на рис. 28. При такой постановке задача идентификации автора текста сводится к задаче классификации или кластеризации.

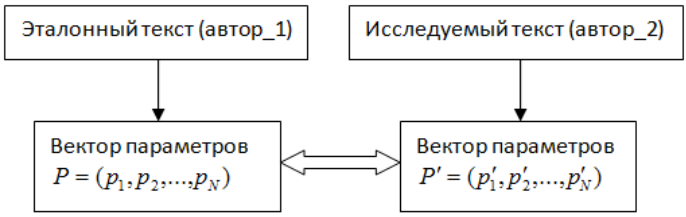


Рис. 28. Формализация задачи атрибуции

В большинстве случаев в качестве характеризующих параметров текста выбираются те или иные его статистические характеристики: частота использования определенных частей речи, некоторых конкретных слов, знаков препинания, фразеологизмов, архаизмов, редких и иностранных слов, количество и длина предложений (измеренная в словах, слогах, знаках), объем словаря, количество полнозначных и служебных слов, средняя длина предложения, отношение числа глаголов к общему количеству словоупотреблений в тексте и т. д.

Основная проблема формальных методов анализа авторства состоит в выборе параметров. Как было отмечено А. А. Марковым [1], существует целый ряд формальных статистических характеристик текстов, непригодных для определения авторства в силу одного из двух недостатков.

Отсутствие устойчивости. Разброс значений параметра для текстов одного и того же автора настолько велик, что диапазоны возможных значений для разных авторов перекрываются. Очевидно, данный параметр не поможет различать авторов, а при использовании в составе группы параметров лишь сыграет роль дополнительного шума.

Отсутствие различающей способности. Параметр может принимать близкие значения для всех или большинства авторов, поскольку его значение определяется свойствами языка, на котором написаны тексты, а не индивидуальными особенностями создателя текста.

Поэтому параметры, используемые в формальных методиках определения авторства, должны предварительно исследоваться на устойчивость и различающую способность, желательно на текстах большого количества различных авторов. В работе [2] выделены три условия применимости формального параметра: условие массовости, устойчивости и различающей способности. Примером формальной характеристики, удовлетворяющей всем трем условиям, является так называемый **авторский инвариант**. Он вычисляется как процент содержания служебных слов (союзов, предлогов, частиц) в тексте. Правда, недостатком такого подхода является низкая разделительная способность в случае большого количества авторов (потенциально метод может разделять только 10 авторских стилей).

Массовость. Параметр должен опираться на те характеристики текста, которые слабо контролируются автором на сознательном уровне. Это условие необходимо, чтобы устранить возможность сознательного искажения автором характерного для него стиля или имитации стиля другого автора.

Устойчивость. Параметр должен сохранять постоянное значение для одного автора. Естественно, в силу случайных причин некоторое отклонение значений от среднего неизбежно, но оно должно быть достаточно мало.

Различающая способность. В наилучшем случае параметр должен принимать существенно различные значения для разных авторов, превышающие колебания, возможные для одного автора. Необходимо отметить, что выбрать параметры, которые гарантированно разделяют двух любых авторов, очень трудно. Какими бы ни были параметры, всегда существует вероятность того, что два или более авторов окажутся по данным параметрам близки. Поэтому на практике считается достаточным, чтобы параметр позволял уверенно различать между собой разные группы авторов, иными словами, чтобы существовало достаточно большое количество групп авторов, для которых средние значения параметра существенно различаются. Параметр, очевидно, не поможет различить тексты авторов из одной группы, но позволит уверенно различать тексты авторов, попавших в разные группы. Различать тексты авторов одной группы можно за счет использования одновременно достаточно большого вектора различных по характеру параметров. В этом случае вероятность случайного совпадения станет заметно меньше. Для уверенного вывода в отношении текстов, для которых формально вычисленное параметрическое расстояние мало, требуется дополнительное исследование экспертными методами.

Еще одной проблемой, возникающей при решении задачи определения авторства, является составление выборки эталонных текстов. С возрастанием объема текста параметры, характеризующие авторский стиль, становятся устойчивыми с вероятностной точки зрения, что позволяет устанавливать авторство по стабильно повторяющимся формальным характеристикам текста. Поэтому более высокое качество атрибуции достигается для текстов большого объема, и менее точный результат получается для текстов маленького объема.

Наиболее полная классификация основных формальных методов атрибуции текстов содержится в работе [3]. Согласно ей все формальные методы можно разделить на статистические и методы машинного обучения. Рассмотрим следующие основные методы атрибуции и программные средства, реализующие их [4].

1. Методы энтропийного кодирования (система «Лингвоанализатор»).
2. Статистические методы (системы «Атрибутор», «Антиплагиат»).
3. Гибридный метод («СМАЛТ»: Статистические Методы Анализа Литературного Текста).
4. Методы классификации и кластеризации на основе машинного обучения (системы «Стилеанализатор», «Авторовед»).

1. Методы энтропийного кодирования (система «Лингвоанализатор»)

Суть метода в том, чтобы добавлять текст, автор которого неизвестен, к тексту, принадлежащему конкретному автору, и смотреть, насколько хорошо сжимается этот текст вместе с «добавкой». Правильный исходный

класс документа – это тот, на котором он сжимается лучше всего. Существенное преимущество метода энтропийной классификации (с помощью сжатия) состоит в отсутствии предварительной обработки текста.

Сжатие данных – алгоритмическое преобразование данных с целью уменьшения занимаемого ими объема. Сокращение объема данных достигается за счет замены часто встречающихся данных короткими кодовыми словами, а редких – длинными (в этом суть **энтропийного кодирования**), либо заменой повторяющейся последовательности ссылкой на уже закодированный фрагмент с указанием его длины.

Несущественная информация – это информация, которой можно пренебречь при сжатии. Ясно, что восстановить первоначальные данные получится только с искажениями. В этом случае говорят о **кодировании с потерями.**

Избыточная информация – неоднократное повторение фрагментов информации. Избыточность может быть устранена без потери информации. Если алгоритм кодирования отбрасывает только избыточную информацию, то говорят о **кодировании без потерь.**

При использовании сжатия без потерь возможно полное восстановление исходных данных, сжатие с потерями позволяет восстановить данные с искажениями, обычно несущественными с точки зрения дальнейшего использования восстановленных данных. Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ, реже – для сокращения объема аудио- и видеоданных, цифровых фотографий и т. п., в случаях, когда искажения недопустимы или нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объема аудио- и видеоданных и цифровых фотографий в тех случаях, когда такое сокращение является приоритетным, а полное соответствие исходных и восстановленных данных не требуется.

Коэффициент сжатия – основная характеристика алгоритма сжатия.

$$k = \frac{S_o}{S_c}, \text{ где}$$

k – коэффициент сжатия;

S_o – объем исходных данных;

S_c – объем сжатых данных.

Чем выше коэффициент сжатия, тем алгоритм эффективнее.

Средняя длина кодового слова вычисляется по формуле

$$\overline{n} = \sum_i n_i P(X_i), \text{ где}$$

n_i – длина кодового слова X_i;

$P(X_i)$ – вероятность кодового слова X_i.

Если вероятность появления элемента X_i равна $P(X_i)$, то наиболее выгодно будет представить этот элемент $-\log_2 P(X_i)$ битами. Если при кодировании удается добиться того, что длина всех элементов будет приведена к $\log_2 P(X_i)$ битам, то и длина всей кодируемой последовательности будет минимальной для всех возможных методов кодирования. При этом, если распределение вероятностей всех элементов $\{P(X_i)\}$ неизменно и вероятности элементов взаимно независимы, то средняя длина кодового слова может быть рассчитана как

$$H(X) = -\sum_i P(X_i) \cdot \log_2(X_i).$$

Это значение называют **энтропией распределения вероятностей** (или **энтропией источника** в заданный момент времени).

Все методы сжатия данных основаны на простом принципе. Если представить, что наиболее часто встречающиеся элементы закодированы более короткими кодами, а реже встречающиеся – более длинными, то для хранения всех данных потребуется меньше места, чем если бы все элементы представлялись кодами одинаковой длины.

Точная взаимосвязь между частотой появления элементов и оптимальной длиной кодового слова описана в теореме Шеннона об источнике шифрования (Shannon's source coding theorem), которая определяет предел максимального сжатия без потерь и энтропию Шеннона.

Теорема Шеннона

$$\frac{H(X)}{\log_2 a} \le ES < \frac{H(X)}{\log_2 a} + 1, \text{ где}$$

$H(X)$ – энтропия источника;

ES – оптимальная длина кодового слова;

a – количество букв в алфавите кодирования.

Ряд исследований был проведен Д. В. Хмелёвым [5, 6], результатом которых явился вывод об эффективности применения алгоритмов сжатия данных для задачи определения авторства. Также был сделан вывод о том, что простейший подход с использованием цепей Маркова первого порядка

показывает хорошие результаты на файлах большого объема и плохие по сравнению с другими методами на отрывках длиной в 2000–5000 символов. Этот метод был реализован в системе «Лингвоанализатор». Система доступна по адресу: http://www.rusf.ru/books/analysis.

Метод, предложенный Д. В. Хмелёвым, основан на применении относительной энтропии. Есть несколько способов вычислить эту характеристику: «шельфовый» (off-the-shelf) алгоритм, метод предсказания по частичному совпадению (PPM – Prediction by Partial Matching) и использование индекса повторяемости. Например, модель PPM использует контекст – множество символов в несжатом потоке, предшествующих данному, чтобы предсказывать значение символа на основе статистических данных. Сама модель PPM лишь предсказывает значение символа, непосредственное сжатие осуществляется алгоритмами сжатия данных.

Среди алгоритмов сжатия данных без потерь наиболее часто встречающимися являются кодирование Хаффмана, арифметическое кодирование, метод Барроуза-Уилера и множество вариаций метода Лемпеля-Зива (LZ). К алгоритмам, специально ориентированным на сжатие текста, относятся следующие: PPM использует Марковскую модель небольшого порядка, DMC (Dynamic Markov compression) использует динамически изменяемую Марковскую модель. В рамках подхода PPM правильный исходный класс документа – это тот, на чьей модели получается наилучшее сжатие. Каждый алгоритм имеет большое число модификаций и параметров (например, существует динамическое кодирование Хаффмана, варьируется объем используемого словаря и пр.). Кроме того, существует множество «смешанных» алгоритмов, где текст, сжатый, например, с помощью алгоритма PPM, дополнительно кодируется с помощью кода Хаффмана.

Здесь мы не будем углубляться в построение алгоритмов сжатия данных. Отметим только, что все эти алгоритмы реализованы в различных программах, которых в настоящий момент существует довольно много. Дополнительное разнообразие возникает из-за того, что у многих программ имеется несколько версий, которые также имеют разные алгоритмы сжатия. В работе [5] приведены некоторые результаты эксперимента по сравнению точности определения авторства текста с использованием разных алгоритмов сжатия данных.

Ряд экспериментов проводился на коллекции Reuters Corpus Volume 1. Было отобрано 50 авторов с наибольшим объемом статей, всего 1813 статей. Выборка случайно была разбита на 10 равных частей, одна из которых использовалась для тестирования. Лучший результат был получен для метода с применением программы **rar** (точность 89,4 %).

Другой ряд экспериментов был проведен на корпусе текстов, состоящем из 385 текстов 82 писателей. Тексты подверглись предварительной обработке. Во-первых, были склеены все слова, разделенные переносом.

Далее были выкинуты все слова, начинавшиеся с прописной буквы. Оставшиеся слова помещены в том порядке, в каком они находились в исходном тексте с разделителем из символа перевода строки. У каждого из писателей было отобрано по контрольному произведению. Остальные тексты были объединены в обучающие тексты. Объем каждого контрольного произведения составлял не менее 50000–100000 символов. Проведенные исследования показали, что программы сжатия угадывают истинных писателей весьма часто на текстах большого объема. Особенно хорошо проявляет себя программа **rarw** с применением модификации алгоритма Лемпеля–Зива (точность 71 %), результаты применения которой превосходят реализацию других подходов в этой области. Тем не менее, остаются и открытые вопросы. Например, почему использование программы **rarw**, применяющей модификацию алгоритма LZ, на файлах большого объема опережает многие другие методы, также применяющие модификацию LZ.

2. Статистические методы (системы «Атрибутор», «Антиплагиат»)

Такие методы учитывают статистику употребления пар элементов любой природы, идущих друг за другом в тексте (биграммы букв, слов).

1. По тем произведениям автора, которые достоверно им созданы, вычисляется матрица переходных частот употребления пар элементов. Она служит оценкой матрицы вероятности перехода из элемента в элемент.
2. Для каждого автора строится матрица переходных частот и оценивается вероятность того, что именно он написал анонимный текст (или фрагмент текста).
3. Автором анонимного текста считается тот, для кого вычисленная оценка вероятности больше.

Примерами использования чисто статистических методов для решения задачи определения авторов текстов являются «Атрибутор» и «Антиплагиат».

Система «Атрибутор»

В качестве стилевых признаков используются триграммы букв. При таком методе анализу поддаются однобуквенные и двухбуквенные служебные слова (предлоги, союзы, частицы и междометия, которые традиционно считаются значимыми стилеметрическими показателями). Двух-, четырехбуквенные и более длинные цепочки оказываются менее показательными.

Можно выделить следующие особенности системы. Во-первых, необходимо исключать те случаи, когда известный писатель в какой-то период своего творчества резко менял стиль изложения. Во-вторых, длина текста должна быть не меньше шести страниц, чтобы избежать ошибок, связан-

ных со сравнением статистически несопоставимых объектов. Кроме того, для более высокого качества атрибуции требуется исключать из текста имена собственные.

Система «Антиплагиат»

Этот интернет-сервис позволяет осуществить проверку текстовых документов на наличие заимствований из общедоступных сетевых источников. Есть возможность проводить атрибуцию текстов на различных языках. Осуществляется сравнение последовательностей символов без учета языковых особенностей и речевых взаимосвязей.

На первом этапе система собирает информацию из различных источников: загружает из Интернета и обрабатывает сайты, находящиеся в открытом доступе, базы научных статей и рефератов. Загруженные документы проходят процедуру фильтрации, в результате которой отбрасывается бесполезная с точки зрения потенциального цитирования информация (например, HTML-страницы с большим количеством рекламы, новостные заголовки и т. д.).

На следующем этапе каждый из полученных таким образом текстов форматируется и заносится в системную базу данных. Кроме того, в общую базу текстов поступают документы, загруженные на проверку пользователем, если такая возможность была разрешена им во время процедуры загрузки. Все пользовательские документы, загружаемые для проверки, ставятся в очередь на обработку.

Поиск совпадений осуществляется методом сравнения последовательностей символов без учета языковых особенностей и речевых взаимосвязей. За счет этого достигается высокая, в несколько секунд, скорость поиска совпадений. Проверка документа, например, реферата среднего размера, занимает несколько секунд. После проверки документа пользователь получает отчет, в котором представляются результаты. Структура отчета позволяет выделять в проверяемом тексте заимствованные части как по всем источникам, так и по их любому подмножеству.

К преимуществам системы следует отнести высокую скорость работы и наглядность получаемых результатов. После проверки документа в проверяемом тексте выделяются заимствованные части как по всем источникам, так и по их любому подмножеству, что очень наглядно для пользователя. Система позволяет проводить атрибуцию текстов на различных языках. К недостаткам можно отнести то, что все программные алгоритмы являются коммерческой тайной, и открытого доступа к ним нет.

3. Гибридный метод (система «СМАЛТ»)

Перед использованием статистических методов осуществляется предварительная обработка исследуемого текста. Обработка производится в несколько этапов.

1. Выполняется автоматизированное разбиение исходного текста на лексические единицы, среди которых выделяется часть (или раздел), абзац, предложение, слово.
2. Осуществляется морфологический разбор.
3. Осуществляется синтаксический анализ.

Базой данных литературных произведений для проведения исследований послужила 81 публицистическая статья 60–70 гг. XIX в. журналов «Время» и «Эпоха». Целью было установить, является ли действительным автором выбранных статей Ф. М. Достоевский. Проводилось исследование с выборками разных объемов: в 200, 300, 400, 500 и 600 слов. В качестве параметров были взяты следующие величины: средняя длина слова в буквах; средняя длина предложения в словах; индекс разнообразия лексики (отношения числа разных словоформ к числу словоупотреблений).

Для выбора параметров используется метод проверки гипотез с помощью критерия Стьюдента (см. главу 7). Далее применяется метод иерархической кластеризации с двумя мерами расстояния между объектами (Евклидова мера и мера Чебышева). Для определения расстояния между кластерами использовались методы ближнего и дальнего соседа.

Исследование проводилось на основе двух наборов признаков: основного, состоящего их частей речи (16 признаков), и расширенного, с подключением дополнительных морфологических параметров, например падежа, рода и т. п. (156 признаков).

Применение методов иерархической кластеризации показало неэффективность использования формально-грамматических параметров для классификации исследуемых статей с целью решения задачи атрибуции. Более того, было доказано, что увеличение числа этих параметров не улучшает результаты исследования.

Главный недостаток системы состоит в том, что задачу определения авторства сводится к задаче построения качественного и быстрого синтаксического анализатора.

4. Методы классификации и кластеризации на основе машинного обучения (системы «Стилеанализатор», «Авторовед»)

Система «Стилеанализатор»

В системе реализованы различные типы алгоритмов классификации: иерархические методы, деревья решений, вариации энтропийного метода и методы на основе нейронных сетей прямого распространения. В программе имеется возможность задания числа слоев и нейронов сети, этапов обу-

чения сети, числа итераций, скорости обуечения и т. п. В качестве меры сравнения матриц частот появления признаков использовалась мера хи-квадрат и дивергенция Йенсена–Шеннона (см. главу 6).

В экспериментах по сравнению методов классификации было взято два набора текстов: художественных произведений (156 текстов, разбиение производилось на три подмножества: 30, 20 и 10 авторов) и газетных статьей (5697 текстов за 2003–2004 гг. из 57 журналистов). Рассмотрены количественные признаки трех уровней: уровня букв, уровня слов и уровня предложений. Всего 14 различных наборов признаков. В результате были сделаны следующие выводы.

1. Было обнаружено, что для разных текстов, с разным числом классов, для разных наборов признаков существует постоянное минимальное значение объема фрагментов для приемлемой классификации. Оно составляет примерно 30000–40000 символов, или 5000–6000 слов, или 400–600 предложений.
2. Методы энтропийного кодирования выигрывают как в скорости обучения, так и в качестве классификации.
3. Нейронные сети дают сопоставимое качество, но сильно проигрывают в скорости.
4. Деревья решений обеспечивают наихудшее качество классификации, но при этом дают наглядный вид решения и по ходу производят отбор самых информативных признаков.

Система «Авторовед»

В этой системе применяются метод опорных векторов и одна из разновидностей нейронных сетей.

Основные результаты проведенных исследований были получены на корпусе, состоящем из 215 прозаических текстов 50 русских писателей. Тексты взяты из электронной библиотеки М. Мошкова. Использовались выборки объемом 1000–100000 символов. Количество обучающих примеров каждого автора бралось равным трем.

Вывод: Автора можно определить с точностью в среднем 0,95–0,98 при объеме текстовой выборки 20000–25000 символов. При этом начиная с 10000 метод SVM показывает лучшие результаты, чем нейронные сети.

Установлено, что использование при идентификации автора комбинации частот букв русского языка, знаков пунктуации, наиболее частых триграмм символов и наиболее частых слов увеличивает точность идентификации в среднем на 6–12 % на объемах текста до 10000 символов.

В таблице 11 приведено сравнение некоторых систем атрибуции текстов, рассмотренных в данном разделе. Видно, что наибольшую точность дают системы «Стилеанализатор» и «Авторовед», использующие методы машинного обучения.

Таблица 11

Сравнение систем атрибуции текстов

Название	Методы	Средства анализа текстов	Необходимый объем текста	Точность, %
Лингво-анализатор	Методы энтропийного кодирования	Графем., стат. анализ	40000–100000 символов	84–89
Атрибутор	Статистические методы	Стат. анализ	>20000 символов	Не изв.
СМАЛТ	Методы из теор. вер. и мат. стат., автоматизация морф. и синт. анализа	Графем., морф., синт., стат. анализ, поддержка дореволюц. орф.	500 слов для определения однород-ности	Не изв.
Стиле-анализатор	Методы из теор. вер. и мат. стат., методы машинного обучения	Графем., стат. анализ	30000–40000 символов	90–98
Авторовед	Методы из теор. вер. и мат. стат., методы машинного обучения	Графем., морф., стат. анализ	20000–25000 символов	95–98
			100 символов	76

8.2. Задача определения тематической принадлежности текстов

Рассмотрим пример из сообщения о ремонте концертного зала: 10 раз упоминается словосочетание «концертный зал» (тема культуры) и по одному разу упоминаются 10 разных терминов, напрямую связанных с техническими аспектами ремонта (тема ЖКХ). Никаких других терминов по теме «культура» в сообщении нет. Очевидно, что тема ЖКХ в этом сообщении представлена более многогранно и полно. Она доминирует над темой культуры, которая представлена лишь косвенно. Как вариант предлагается рассматривать процентную принадлежность к различным темам: «культура» – 20 %, «ремонт» – 50 %, «образование» – 30 %. Для коллекции документов результатом работы системы определения тематической принадлежности текстов может являться, например, таблица 12.

Таблица 12

Результат работы системы определения тематической принадлежности текстов

	Наука	Спорт	Культура	ЖКХ	Транспорт	Здоровье
Текст 1	21 %		53 %			26 %
Текст 2		39 %	52 %	9 %		
Текст 3	19 %		48 %		23 %	10 %

Тематическая принадлежность текстов может быть полезна для

– автоматической рубрикации текстов по темам;
– определения релевантности документов поисковому запросу в поисковых системах;
– игнорирования спамдексинга. **Спамдексинг (поисковый спам)** – злоупотребление частотой ключевых слов с целью манипулирования поисковыми системами; общее название для методов неэтичного продвижения сайтов, так называемая «черная оптимизация», или поисковая оптимизация (Search Engine Optimization, SEO).

Следует отметить, что в последние годы уже начали появляться интернет-сервисы, которые по ключевым словам вычисляют степень присутствия на web-странице некоторых наиболее общераспространенных тематик и жанров. Например, они могут отличать научный текст от остальных жанров. Из русскоязычных интернет-сервисов, способных на такое, можно указать сервис «Семантическое Зеркало» (http://www.ashmanov.com/tech/semantic/demo). Результат работы приведен на рис. 29. Тем не менее, вычислить уровень доминирования более узких тематик, интересующих пользователя, такие сервисы не способны.

Результаты для страницы http://chessok.net/programs/

Результаты «Семантического зеркала»

Категория	Название	Вес
Sports	Спорт	75.0%
Sports/ChessDraughts	Шахматы, шашки	74.2%
Sports/ChessDraughts/Chess	Шахматы	73.6%
SciTech	Техника и наука	64.7%
SciTech/Technics	Техника, электроника	64.7%
SciTech/Technics/CompInt	Компьютеры, Интернет	56.8%
SciTech/Technics/CompInt/Computers	Компьютеры	55.4%
Leisure	Досуг	59.9%
Leisure/Games	Игры	58.8%
Leisure/Games/ComputerGames	Компьютерные игры	58.2%
Leisure/Games/ComputerGames/SimulationGames	Симуляторы	55.1%
Leisure/Games/ComputerGames/SimulationGames/Sports	Спортивные	55.1%

Результаты «Автоконтекста»

Термин	Вес
программы	5.618715
игре	5.618715
Скачать	5.618715
бесплатно	4.398581
мира	3.847276
sony ericsson	1.548325
игры шахматы	1.138962
шахматные программы	0.637123
Скачать шахматы	0.634765
скачать шахматные	0.547528
шахмат мира	0.534575
шахмат Чемпионы мира	0.505401
уровень игры	0.503146
Скачать бесплатно	0.500677
онлайн шахмат	0.469413

Рис. 29. Сервис «Семантическое Зеркало»

Вычислить уровень доминирования тематики в тексте можно с помощью лексических индикаторов конкретных тем (предметных областей), т. е. ключевых слов, которые всегда однозначно определяют контекст. Другой способ – применить методы нечеткой логики Заде.

1. Лексические индикаторы тем (с привлечением тезаурусов)

Абсолютная частота встречаемости одного и того же слова вовсе не связана напрямую с уровнем доминирования темы. Проверка наличия в тексте определенной тематики является для компьютерной программы статистической гипотезой. Как известно, проверить ее в общем случае можно, вычислив отношение произошедших событий, делающих истин-

ность гипотезы более вероятной, ко всем возможным событиям в изучаемой выборке. В такой модели тематической индексации роль выборки выполняет лексический состав анализируемого текста. «Всеми возможными событиями» при этом являются употребления в индексируемом тексте лексических индикаторов всех тем из ограниченного списка, которые нас интересуют. Это исходное предположение человека (аналитика-эксперта), что количество лексических индикаторов именно такое.

Каждое слово или выражение может потенциально выражать множество смыслов относительно разных контекстов и концептов конкретного текста. Необязательно, да и невозможно учитывать все эти смыслы и контексты для решения конкретных поисково-прикладных задач. Поэтому величину представленности или доминирования тем в тексте правильнее подсчитывать только относительно других тем, заранее заданных пользователем в виде ограниченного списка. Это позволит учитывать смысловое взаимовлияние разных тем в рамках одного текста для корректной тематической индексации. Следует подчеркнуть, что в современных поисковых системах такое взаимовлияние не учитывается.

В подобных ситуациях полезным инструментом является нечеткая логика Заде [7].

2. Нечеткая логика Заде (Fuzzy logic Lotfi A. Zadeh)

Пусть E – универсальное множество, x – элемент E, а R – некоторое свойство.

Четкое (обычное) **подмножество** A универсального множества E, элементы которого удовлетворяют свойству R, определяется как множество упорядоченных пар $A=\{\mu_A(x)/x\}$, где $\mu_A(x)$ – **функция принадлежности** (или **характеристическая функция принадлежности**), принимающая значение 1, если x удовлетворяет свойству R, и значение 0 – в противном случае.

Нечеткое подмножество отличается от обычного тем, что для элементов $x \in E$ нет однозначного ответа «да – нет» относительно свойства R.

Нечеткое подмножество A универсального множества E определяется как множество упорядоченных пар $A=\{\mu_A(x)/x\}$, где $\mu_A(x)$ – функция принадлежности, принимающая значения в некотором вполне упорядоченном множестве M (например, $M=[0,1]$).

Функция принадлежности указывает степень (или уровень) принадлежности элемента x подмножеству A. Множество M называют **множеством принадлежностей**.

Если $M=\{0,1\}$, то нечеткое подмножество A может рассматриваться как четкое (обычное) множество.

Нечёткое подмножество может быть задано различными способами: табличным, графическим или аналитическим. Нечеткое множество можно задать при помощи **таблицы значений** характеристической функции.

Пример

Человека, у которого лоб среднего размера, короткий нос, большой разрез глаз и цвет глаз ближе к темному, можно описать следующим нечетким подмножеством с помощью таблицы значений (таблица 13).

Таблица 13

Таблица значений характеристической функции принадлежности

	параметр	**0**	**1**
x_1	высота лба	0,5	0,5
x_2	длина носа	1	0
x_3	разрез глаз	0	1
x_4	цвет глаз	0,3	0,7

Пример

Пусть $E = \{1, 2, 3, \ldots, 100\}$ и соответствует понятию «возраст».

Тогда нечеткое множество «молодой» можно определить при помощи характеристической функции:

$$\mu_{\text{"молодой"}}(x) = \begin{cases} 1, & x \in [1, 25] \\ \dfrac{1}{1 + \left(\dfrac{x-25}{5}\right)^2}, & x \geq 25. \end{cases}$$

Основные операции над нечеткими множествами

Для нечетких множеств, как и для обычных, определены основные логические операции. Самыми основными, необходимыми для расчетов, являются пересечение и объединение.

Дополнение. Пусть A и B – нечеткие множества, заданные на E. A и B дополняют друг друга, если $\forall x \in E \quad \mu_A(x) = 1 - \mu_B(x)$.

Обозначение: $B = \overline{A}$ или $A = \overline{B}$. Очевидно, что $\overline{(\overline{A})} = A$.

Пересечение $A \cap B$ (нечеткое «И») – наибольшее нечеткое подмножество, содержащееся одновременно в A и B, с функцией принадлежности $\forall x \in E \quad \mu_{A \cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right)$.

Объединение $A \cup B$ (нечеткое «ИЛИ») – наименьшее нечеткое подмножество, включающее как A, так и B, с функцией принадлежности $\forall x \in E \quad \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$.

Нечёткой переменной называется тройка $< \alpha, X, A >$, где

α – имя нечёткой переменной;

X – универсальное множество, которое является областью определения нечёткой переменной α;

A – нечёткое подмножество универсального множества X, для каждого элемента которого определена функция принадлежности $\mu_A(x)$ (при $x \in X$), описывающая ограничения на значения нечёткой переменной α.

Лингвистической переменной называется пятерка $< \beta, X, T, G, M >$, где

β – имя лингвистической переменной;

X – универсальное множество, которое является областью определения лингвистической переменной β;

T – некоторое множество значений лингвистической переменной β, каждое из которых является нечеткой переменной на множестве X (для каждого элемента которого определена функция принадлежности $\mu_T(x)$ (при $x \in X$), описывающая ограничения на значения нечеткой переменной β).

Множество T называют **базовым терм-множеством**, поскольку оно задает минимальное количество значений, на основании которых при помощи правил G и M можно сформировать остальные допустимые значения лингвистической переменной.

G – синтаксическое правило, при помощи которого генерируются новые термы из слов естественного языка.

M – семантическое правило, определяющее вид функции принадлежности для каждого значения, образованного при помощи правила G.

Пример

β = «Скорость»

$X = [0, 120]$

T = {«Быстро», «Медленно»}

G = {«Очень»}

M задается таблицей, которая каждому значению лингвистической переменной ставит в соответствие нечёткое подмножество множества X.

Очень медленно	Медленно	Быстро	Очень быстро
0–40	41–70	71–90	91–120

8.3. Задача анализа тональности текстов

Анализ тональности текстов (определение эмоциональной окраски текстов, сентимент-анализ, Sentiment analysis, Opinion mining) – область компьютерной лингвистики, которая занимается изучением мнений и эмоций в текстовых документах.

В изучении этого вопроса следует понимать, что любое высказывание несет оттенок субъективности, является отражением реальности именно с точки зрения говорящего, т. е. присутствует некоторая модальность (модальность – семантическая категория, выражающая отношение говорящего к содержанию его высказывания). Автор всегда вкладывает в высказывание свой жизненный опыт, оперирует своими образами. Здесь можно вспомнить выражение: «Все в мире относительно».

Анализ тональности находит свое применение в

- социологии – сбор разного рода данных из соц. сетей (например, о религиозных взглядах);
- медицине и психологии – выявление психических особенностей и особенностей поведения конкретных пользователей и групп пользователей. Например, также можно определять тенденцию к возникновению депрессии у пользователей соцсетей в связи с политическими, экономическими трудностями;
- маркетинге – например, можно оценить фильм, ресторан, гостиницу и пр.;
- политологии – сбор свежих данных из блогов о политических взглядах населения.

Целью является нахождение мнений в тексте и определение их свойств, при этом каждое высказывание можно представить в виде набора признаков:

- **субъект тональности (автор)** – кому принадлежит высказывание;
- **объект тональности и его свойства (тема)** – о чем говорится в высказывании;
- **тональная оценка** – позиция автора относительно упомянутой темы.

В современных системах автоматического определения эмоциональной оценки текста чаще всего используется одномерное эмотивное пространство: позитив или негатив (хорошо или плохо). Конечно, гораздо более интересны случаи использования многомерных пространств.

Одним из минусов данного подхода является то, что эмоциональную составляющую документа не всегда можно однозначно определить, т. е. документ может содержать как признаки позитивной оценки, так и признаки негативной. Поэтому в некоторых случаях удобнее считать, что необходимо осуществить классификацию полярности выбранного документа (высказывания) на три класса: позитивный, негативный или нейтральный.

Причем классификация, как говорилось раньше, может быть точная (когда результат {0, 1}) или пороговая (когда результат [0, 1]). В последнем случае может подразумеваться оценивание по пяти-, десятибалльной шкале, или, например, по шкале от –10 до 10.

Пример

Сообщение в Твиттере:

Программа «Глобальное образование»: наши студенты получат гранты на обучение в ведущих университетах мира.

Автор: Дмитрий Ливанов @DmitryLivanov

Тема: Выплаты российским студентам

Тональность: положительная

Например, у нас есть подборка рецензий на художественный фильм, и стоит задача определить, какие это рецензии – положительные или отрицательные. Эту задачу можно решить с помощью автоматической системы оценки тональности текста: система определяет характер рецензии, анализируя языковые средства.

Несмотря на то, что тональность является лишь одной из характеристик мнения, именно задача классификации тональности является наиболее часто изучаемой в наши дни. Это можно объяснить несколькими причинами.

1. Определение автора и темы являются гораздо более трудными задачами, чем классификация тональности, поэтому имеет смысл сначала решить более простую задачу, а затем уже переключиться на остальные.

2. Во многих случаях достаточно лишь определить тональность, т. к. другие характеристики уже известны. Например, если мы собираем мнения из блогов, обычно авторами мнений являются авторы постов, следовательно, определять автора нам не требуется. Также зачастую нам уже известна тема: например, если мы производим в Твиттере поиск по ключевому слову «Windows 8», то затем нужно лишь определить тональность найденных твитов.

Существуют различные подходы к определению тональности: основанные на правилах; основанные на словарях; использующие машинное обучение с учителем; использующие машинное обучение без учителя. В

контексте задачи анализа тональности текстов методы машинного обучения без учителя, вообще говоря, показывают очень низкую точность. По этой причине их редко используют в системах сентимент-анализа несмотря на то, что они автоматизированы и не требуют данных для обучения. Рассмотрим оставшиеся три подхода.

1. Системы, состоящие из набора правил

Например, для предложения «*Я люблю шоколадные конфеты*», можно применить следующее правило: если слово («люблю») входит в положительный набор глаголов («люблю», «обожаю», «нравится» и т. д.), и в предложении не имеется отрицаний, то классифицировать тональность как «положительная».

Ввиду того, что этот метод наиболее точный, многие коммерческие системы используют данный подход несмотря на то, что он требует больших затрат, т. к. для хорошей работы системы необходимо составить большое количество правил. Зачастую правила привязаны к определенной теме (например, «ресторанная тематика»), и при смене темы («обзор фотоаппаратов») требуется заново составлять правила. Тем не менее, этот подход является наиболее точным при наличии хорошей базы правил, но совершенно неинтересным для исследования.

Преимущества подхода

– Наиболее точный.

Недостатки подхода

– Требует больших затрат для создания правил и поддержки их в актуальном состоянии.
– Классификатор привязан к определенной теме.
– Неинтересен для исследования (в нашем случае это недостаток).

2. Подходы, основанные на словарях, используют тональные словари, которые представляют собой список слов со значением тональности для каждого слова, как например, в таблице 14.

Таблица 14

База ANEW, переведенная на русский язык

слово	**тональность (1–9)**
счастливый	8,21
хороший	7,47
скучный	2,95
сердитый	2,85
грустный	1,61

Чтобы проанализировать текст, можно воспользоваться следующим алгоритмом: сначала для каждого слова в тексте определяется его тональность из словаря (если оно присутствует в словаре), а затем вычисляется общая тональность всего текста. Вычислять общую тональность можно разными способами. Самый простой из них – среднее арифметическое всех значений.

Основной проблемой словарных методов считается трудоемкость процесса составления словаря: чтобы получить метод, классифицирующий документ с высокой точностью, термины словаря должны иметь вес, адекватный предметной области документа. Например, слово «большой» является положительной характеристикой по отношению к объему памяти жесткого диска, но отрицательной по отношению к размеру мобильного телефона.

Существует ряд тезаурусов, специально размеченных с учетом эмоциональной составляющей. Такие словари, описанные далее, необходимы компьютерным программам при анализе тональности текста.

WordNet-Affect (http://wndomains.fbk.eu/wnaffect.html) – это семантический тезаурус, в котором понятия, связанные с эмоциями («эмоциональные концепты», англ. «affective concepts») представлены с помощью слов, обладающих эмоциональной составляющей («эмоциональные слова», англ. «affective words»). WordNet-Affect состоит из такого подмножества синсетов WordNet, где каждый синсет, соответствующий «эмоциональному концепту», может быть представлен с помощью «эмоциональных слов».

WordNet-Affect является расширением WordNet Domain, где каждому синсету приписано не менее одной пометы предметной области (например, спорт, политика, медицина). Всего в иерархически организованную структуру было включено около двухсот предметных помет. Таким образом, WordNet-Affect был создан на основе WordNet для английского языка путем выбора и отнесения наборов синонимов (синсетов) к различным эмоциональным понятиям. В частности, синсеты глаголов, существительных, прилагательных, наречий, которые представляют собой описание эмоций, были вручную размечены с помощью специальных эмоциональных меток (affective labels, A-labels). Эти эмоциональные метки характеризуют различные состояния, выражающие настроения, эмоциональные отклики или ситуации, которые вызывают эмоции. Примеры таких эмоциональных меток на русском и английском языках приведены в таблице 15. Существуют версии WordNet-Affect для других языков.

Подобный словарь, позволяющий разделить эмоции только на три класса (положительную, отрицательную, нейтральную), SentiWordNet (http://sentiwordnet.isti.cnr.it/) широко используется в англоязычных автоматических системах сентимент-анализа.

Таблица 15

Примеры меток в словаре для описания эмоций

Эмоциональная метка	**Пример**
Эмоция (emotion)	сущ. гнев#1, гл. бояться#1 (fear)
Настроение (mood)	сущ. враждебность#1 (animosisy), прил. любезный#1J (amiable)
Особенность (trait)	сущ. агрессивность#1 (aggressiveness), прил. конкурентный#1 (competitive)
Когнитивное состояние (cognitive state)	сущ. замешательство#2 (confusion), прич. изумленный#2 (dazed)
Физическое состояние (physical state)	сущ. хворь#1 (illness), прич. утомленный#1 (exhausted)
Гедонический сигнал (gedonic signal)	сущ. боль#3(hurt), сущ. страдание#4 (suffering)
Ситуации, вызывающие эмоции (emotion-eliciting situation)	сущ. неловкость#3 (awkwardness), сущ. безопасность#1 (out of danger)
Эмоциональные отклики (emotional responses)	сущ. холодный пот#1 (cold sweat), гл. дрожать#2 (tremble)
Поступки (behaviour)	сущ. преступление#1 (offense), прич. замедленный#1 (delayed)
Отношение, позиция (attitude)	сущ. нетерпимость#1 (intolerance), сущ. оборона#1 (defensive)
Чувство (sensation)	сущ. холод#1 (coldness), гл. чувствовать#3 (feel)

Для русского языка создан словарь эмоциональной лексики (http://www.cir.ru/SentiLexicon/ProductSentiRus.txt), представляющий собой список из 5000 оценочных слов, извлеченных из коллекций отзывов в нескольких предметных областях (фильмы, книги, игры, телефоны, камеры). Словарь был разработан прежде всего для выявления предпочтений в маркетинговых исследованиях, а не для анализа общественного мнения в текстах социально-политической направленности.

Преимущества подхода

– Простой в применении.

Недостатки подхода

– Сильная зависимость терминологии от контекста.
– Трудоемкость составления словарей.

3. Системы, использующие машинное обучение с учителем

Методы классификации, использующие машинное обучение с учителем, рассмотрены в предыдущей главе. Напомним, что процесс классификации состоит из следующих этапов.

1. Индексация документов.
2. Уменьшение размерности пространства признаков.
3. Построение и обучение классификатора.
4. Оценка качества классификации.

Для каждого документа из обучающей выборки нужно указать «правильный» ответ, т. е. тип тональности (например, положительная или отрицательная); по этим ответам и будет обучаться классификатор.

На этапе индексации документов необходимо осуществить выбор признаков. В качестве признаков можно рассматривать кобинации *n*-грамм слов или *n*-грамм символов.

Пример (n-граммы слов)

Предложение «*Мне нравятся шоколадные конфеты*».

Набор униграмм: (*Мне, нравятся, шоколадные, конфеты*)

Набор биграмм: (*Мне нравятся, нравятся шоколадные, шоколадные конфеты*)

Их комбинация: (*Мне, нравятся, шоколадные, конфеты, Мне нравятся, нравятся шоколадные, шоколадные конфеты*)

Обычно униграммы и биграммы слов дают лучшие результаты, чем *n*-граммы более высоких порядков (триграммы и выше), т. к. обучающая выборка в большинстве случаев недостаточно большая для подсчета *n*-грамм высших порядков.

Пример (n-граммы символов)

Предложение «*Мне нравятся шоколадные конфеты*».

Набор четырехсимвольных *n*-грамм: (*«Мне », «не н», «е нр», « нра», «нрав», …*).

Символьные *n*-граммы могут быть полезны:

- при наличии орфографических ошибок в тексте – набор символов у текста с ошибками и набор символов у текста без ошибок будет практически одинаков в отличие от слов;
- для языков с богатой морфологией (например, для русского) – в текстах могут встречаться одинаковые слова, но в разных вариациях (разные род или число), но при этом не изменяется корень слов, а следовательно, и общий набор символов.

Несмотря на то, что такой способ может показаться слишком примитивным, т. к. на первый взгляд набор символов не несет в себе никакой семантики, этот метод иногда дает результаты даже лучшие, чем *n*-граммы

слов. Если присмотреться, то можно увидеть, что *n*-граммы символов соответствуют в какой-то мере морфемам слов, а в частности, корень слова («люб») несет в себе его смысл.

Также можно использовать дополнительные признаки, такие как части речи, пунктуация (наличие в тексте смайлов, восклицательных знаков), наличие в тексте отрицаний («не», «нет», «никогда»), междометий и т. д.

Символьные *n*-граммы применяются гораздо реже, чем *n*-граммы слов, но иногда они могут улучшить результаты. Для коротких сообщений, например, в Твиттере, больше подходят символьные *n*-граммы. Для текстов больше подходят *n*-грамм слов. В некоторых случаях полезна их комбинация.

После того как документы проиндексированы, следует осуществить выбор функции взвешивания. В предыдущей главе говорилось, что в качестве весовой функции часто используют TF-IDF. Для задачи оценки тональности текстов в отличие от задачи поиска не слишком важны слова, которые часто повторяются в документе (т. е. слова с высоким TF). Поэтому для анализа тональности стандартная функция TF-IDF не дает хороших результатов. Гораздо лучше сюда подходит бинарная функция взвешивания или модификация функции TF-IDF – так называемая дельта TF-IDF. Основная идея состоит в том, чтобы придать больший вес словам, которые имеют не нейтральную тональность. Первые два шага остаются прежними (см. главу 7). На третьем шаге итоговый вес слова t в документе d вычисляется по формуле:

$$V_{t,d} = C_{t,d} \cdot \log\left(\frac{|N| \cdot P_t}{|P| \cdot N_t}\right), \text{ где}$$

$C_{t,d}$ – количество раз слово t встречается в документе d;

$|P|$ – количество документов положительной тональности;

$|N|$ – количество документов отрицательной тональности;

P_t – количество документов положительной тональности, в которых встречается слово t;

N_t – количество документов отрицательной тональности, в которых встречается слово t.

Пример системы классификации отзывов о фильмах по тональности

Данные для классификатора тональности отзывов о фильмах были собраны с сайта «Кинопоиск» (http://www.kinopoisk.ru/): было отобрано 500 положительных и 500 отрицательных отзывов. В качестве алгоритма клас-

сификации использовался наивный Байесовский классификатор (NB) и метод опорных векторов (SVM). В качестве признаков рассматривались униграммы, биграммы и их комбинация. В качестве функции взвешивания была выбрана бинарная функция для метода Байеса и SVM, и дельта TF-IDF для SVM.

Для оценки работы классификатора была проведена перекрестная проверка: для каждого набора параметров было запущено подряд пять тестов, в каждом из которых использовалось 800 отзывов для обучения и 200 для тестирования. В таблице 16 представлены результаты (точность в процентах) для всех девяти наборов параметров.

Таблица 16

Оценка точности классификатора тональности отзывов о фильмах

Признаки	**NB**	**SVM**	**SVM+delta**
униграммы	85,5	82,5	86,2
биграммы	84,9	86,5	87,8
комбинация	86,5	88,4	90,8

Выводы о проведенном эксперименте.

1. Для данной коллекции лучшие результаты показывает метод опорных векторов с функцией взвешивания дельта TF-IDF.
2. Если использовать обычную бинарную функцию, то оба классификатора (NB и SVM) показывают примерно одинаковые результаты.
3. Комбинация униграмм и биграмм дает лучшие результаты во всех тестах.

Преимущества подхода

– Простой в применении.
– В случае использования символьных *n*-грамм независимость от языка и терминологии.

Недостатки подхода

– Зависимость результата от выбора параметров метода.
– Трудоемкость подготовки данных для обучения и тестирования.

Список литературы

1. Марков А. А. Об одном применении статистического метода // Текстология.ru. URL: http://www.textology.ru/library/book.aspx?BookId=8&textId=2.
2. Фоменко В. П., Фоменко Т. Г. Авторский инвариант русских литературных текстов // Новая хронология Греции: Античность в Средневековье. М. : МГУ, 1995. 422 с.
3. Романов А. С. Методика и программный комплекс для идентификации автора неизвестного текста : автореф. дис. канд. тех. наук. Томск, 2010. 26 с.
4. Батура Т. В. Формальные методы определения авторства текстов // Вестник НГУ. Серия: Информационные технологии. 2012. Т. 10. № 4. С. 81–94.
5. Хмелёв Д. В. Классификация и разметка текстов с использованием методов сжатия данных // Всё о сжатии данных, изображений и видео. URL: http://compression.ru/download/articles/classif/intro.html.
6. Хмелёв Д. В. Распознавание автора текста с использованием цепей А. А. Маркова // Вестник МГУ. Серия: Филология. М. : МГУ, 2000. № 2. С. 115–126.
7. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. М. : Мир, 1976. 166 с.

Учебное издание

Батура Татьяна Викторовна

МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА И АВТОМАТИЧЕСКАЯ ОБРАБОТКА ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Учебное пособие

Редактор *Д. М. Валова*
Обложка *Е. В. Неклюдовой*

Подписано в печать 30.06.2016 г.
Формат 60 x 84 1/16. Уч.-изд. л. 10,3. Усл. печ. л. 9,6.
Тираж 100 экз. Заказ №
Редакционно-издательский центр НГУ
630090, Новосибирск-90, ул. Пирогова, 2.