Графы знаний

Лекция 2 - Представление знаний в графах

М. Галкин, Д. Муромцев
1. **Introduction**

2. Представление знаний в графах - RDF & RDFS & OWL

3. Хранение знаний в графах - SPARQL & Graph Databases

4. Однородность знаний - Reification & RDF* & SHACL & ShEx

5. Интеграция данных в графы знаний - Semantic Data Integration

6. Векторные представления графов - Knowledge Graph Embeddings

7. Введение в теорию графов - Graph Theory Intro

8. Машинное обучение на графах - Graph Neural Networks & KGs

9. Некоторые применения - Question Answering & Query Embedding
Содержание

● Графы знаний = графы + знания
● Как представлять знания? Логика и семантика
● Модель RDF и RDFS
● Сериализации RDF
● OWL, классы, экземпляры, аксиомы
● Онтологии как схемы графов знаний
Представление знаний - онтологическое

Leonard Nimoy starredIn Star Trek; played Spock.

Spock characterIn Star Trek.

Alec Guinness starredIn Star Wars; played Obi-Wan.

Star Wars genre SciFi.
Представление знаний - статистическое

Leonard Nimoy = [0.1, 0.8, 0.1]
Star Trek = [0.22, 0.34, 0.87]
characterIn = [0.1, 0.1, 0.6]

Spock = [0.1, 0.2, 0.3]
Leonard Nimoy = [0.4, 0.8, 0.1]
Star Trek = [0.22, 0.34, 0.87]

Spock = [0.1, 0.2, 0.3]
Star Trek = [0.22, 0.34, 0.87]

Obi-Wan = [0.05, 0.25, 0.37]
Alec Guinness = [0.33, 0.5, 0.3]
Star Wars = [0.18, 0.4, 0.9]
Граф знаний

$G = (V, E) \mid E \subseteq \mathbb{R}^{\left|V\right| \times \left|V\right|}$
Граф знаний

\( G = (V,E) \mid E \subseteq \mathbb{R}^{\vert V \vert \times \vert V \vert} \)

\( \tau(v) = \gamma, \ v \in V \)

\( \tau(e) = \delta, \ e \in E \)
Граф знаний

Граф знаний — это математическое представление структуры и взаимосвязи знаний. Граф — это пара $G=(V,E)$, где $V$ — множество вершин, а $E$ — множество ребер.

$G = (V,E)$, $E \subseteq \mathbb{R}^{|V|x|V|}$

- Вершины и ребра имеют типы (классы)
- Ребра (отношения) могут иметь разную значимость
- Классы и отношения имеют логический смысл
- Сетевая структура графа — более естественный способ представлять знания о мире

$\tau(v) = \gamma$, $v \in V$

$\tau(e) = \delta$, $e \in E$
Формальная семантика

I(Musician writes songs) • Семантика передает смысл символов некоторого формального или естественного языка
Формальная семантика

\[ I(\text{Musician writes songs}) \quad \Rightarrow \quad \text{Семантика передает смысл символов некоторого формального или естественного языка} \]

\[ \text{Musician} \sqsubseteq \exists \text{writes.Song} \quad \Rightarrow \quad \text{Формальная семантика передает смысл языков в математических терминах} \]
Формальная семантика

Формальная семантика передает смысл символов некоторого формального или естественного языка

Формальная семантика передает смысл языков в математических терминах

Формальная логика предоставляет математический аппарат для интерпретации языков

I(Musician writes songs)

Musician ⊑ ∃ writes.Song

C(x), ∈, ∀, ∃, ∩, ∪, ∃R.C, ∀R.C

Семантика передает смысл символов некоторого формального или естественного языка

Формальная семантика передает смысл языков в математических терминах

Формальная логика предоставляет математический аппарат для интерпретации языков
Формальная семантика

*Семантика передает смысл символов некоторого формального или естественного языка*

Формальная семантика передает смысл языков в математических терминах

Формальная логика предоставляет математический аппарат для интерпретации языков

- Пропозициональная логика
- Логика предикатов
- Модальная логика
- Логика первого порядка
- Дескрипционные логики
Формальная семантика

Формальная семантика передает смысл символов некоторого формального или естественного языка

Формальная семантика передает смысл языков в математических терминах

○ Теоретико-модельная семантика (семантика Тарского)

Формальная логика предоставляет математический аппарат для интерпретации языков

○ Пропозициональная логика
○ Логика предикатов
○ Модальная логика
○ Логика первого порядка
○ Дескрипционные логики

I(Musician writes songs)

Musician ∈ ∃writes.Song

C(x), ∈, ∀, ∃, ∨, ⋀, ∃R.C, ∀R.C
Выразительность vs Сложность

Высокая выразительность

✓ Комплексные утверждения
✓ Богатая логика

- Часто алгоритмически неразрешимы
Выразительность vs Сложность

Высокая выразительность

✓ Комплексные утверждения
✓ Богатая логика

- Часто алгоритмически неразрешимы

Низкая выразительность

✓ Для простых задач
✓ Полиномиальная сложность

- Сложные утверждения недоступны

First-order logic (FOL)
Description logics (DL)
Propositional logic (PL)
Идея логического представления знаний в Сети легла в основу концепции Semantic Web.
Semantic Web Layer Cake

The Web Platform

- URI / IRI
- HTTP
- Unicode
- Auth

Базовые средства передачи информации в сети.
Все сущности являются URI

http://example.com/ITMO_University
https://www.wikidata.org/wiki/Q1342013
Semantic Web Layer Cake

Formats

- XML
- Turtle
- JSON-LD
- RDFa

Форматы сериализации моделей высших уровней
Information Exchange

- RDF (Resource Description Framework)

Вводит базовую графовую модель представления знаний в виде триплетов

\langle subject \rangle \ \langle predicate \rangle \ \langle object \rangle
Semantic Web Layer Cake

Models & Rules & Logic

- **RDFS**
- **OWL**
- **SKOS**
- **RIF, SWRL, SPIN**
- **Description Logics**
  - SROIQ(D), SHOIN(D)

Logic extensions to RDF with greater expressiveness

Логические расширения RDF с большей выразительностью
Semantic Web Layer Cake

Querying

- **SPARQL**
- Cypher
- Gremlin

Язык запросов к RDF графам
Semantic Web Layer Cake

Security & Proof & Trust

- SHACL, ShEx
- VoID
- ProvO

Средства обеспечения достоверности и корректности знаний
**Definition 2.1.1** Let \( U, B, L \) be disjoint infinite sets of URIs, blank nodes, and literals, respectively. A tuple \( (s, p, o) \in (U \cup B) \times (U) \times (U \cup B \cup L) \) is denominated as an RDF triple, where \( s \) is called the subject, \( p \) the predicate, and \( o \) the object. \((s,p,o)\) is a generalized RDF triple when \((s,p,o) \in (U \cup B) \times (U) \times (U \cup B \cup L)\). An element \( U \cup B \cup L \) is called an RDF term.
- Модель описания ресурсов
- Ресурсы уникально идентифицируются URI или литералами
RDF - Resource Description Framework

URIs:
- :ITMO_University
- :University
- rdf:type
- :founding_date

"26.03.1900"
RDF - Resource Description Framework

Entities:

URIs:
- :ITMO_University
- :University
- rdf:type
- :founding_date

"26.03.1900"
RDF - Resource Description Framework

URIs:
- :ITMO_University
- :University
- rdf:type
- :founding_date

Entities:

Predicates:
- rdf:type
- :founding_date

"26.03.1900"
RDF - Resource Description Framework

URIs:
- :ITMO_University
- :University
- rdf:type
- :founding_date

Entities:

Predicates:
- rdf:type
- :founding_date

Literals:
- "26.03.1900"
● Uniform Resource Identifier - способ назначения уникальных идентификаторов абстрактным и реальным сущностям и предикатам (RFC 3986)

● URL-подобные идентификаторы для Content Negotiation

● Префиксы для сокращения строк (http://prefix.cc)

https://en.ifmo.ru/en ifmo:ITMO
http://example.org/ITMO_University ex:ITMO_University
http://dbpedia.org/resource/ITMO_University dbr:ITMO_University
https://www.wikidata.org/wiki/Q1342013 wd:Q1342013

● Одна и та же сущность может быть по-разному представлена в разных графах (interlinking & alignment problem)
● Литералы - строки, числа и XSD-определенные типы

“текст” | 100 | True | “5.5”^^xsd:float

● Хороший тон и рекомендация - назначать строкам тэг языка

“строка”@ru | “string”@en | “Die Zeichenfolge”@de | “chaîne”@fr

● В расширениях RDF можно создавать собственные литеральные типы (“числа, делящиеся на 2”)

● Литералы не могут быть субъектом RDF-триплета
РДФ: Blank Nodes

- Неименованные вершины - анонимно заданные сущности без URI или литерала
- Используются в т.ч. для логических аксиом и реификации

```
:ITMO_University rdf:type :founding_date "date"
```
RDF Graph

```
<http://example.org/ITMO_University> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://example.org/University> .
<http://example.org/ITMO_University> <http://example.org/founding_date> "26.03.1900" .
<http://example.org/ITMO_University> <http://example.org/locatedIn> <http://example.org/Saint_Petersburg> .
```
● RDF позволяет:
  ○ Делать утверждения о принадлежности ресурса к множеству
  ○ Назначать ресурсам атрибуты-литералы

● Набор ключевых слов - предикатов (RDF vocabulary)
  ○ Формальная семантика определена в стандарте

● Логическая модель, не зависящая от синтаксиса
  ○ Но есть популярные сериализации
    ■ Turtle (.ttl)
    ■ JSON-LD (.jsonld)
    ■ XML/RDF (.rdf)
    ■ N-Triples (.nt)
    ■ N3 (.n3)
    ■ TriG (.trig), TriX (.trix)
Сериализация RDF: Turtle

@prefix ifmo: <http://en.ifmo.ru/> .
@prefix ex: <http://example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ifmo:ITMO_University rdf:type ifmo:University ;
ifmo:locatedIn ex:Saint_Petersburg ;
ifmo:foundedIn "26.03.1900"^^xsd:date .

https://www.w3.org/TR/turtle/
Сериализации RDF: JSON-LD

```
{  
  "@graph" : [ {  
    "@id" : "ifmo:ITMO_University",
    "@type" : "ifmo:University",
    "foundedIn" : "26.03.1900",
    "locatedIn" : "ex:Saint_Petersburg"
  } ],
  "@id" : "urn:x-arq:DefaultGraphNode",
  "@context" : {  
    "foundedIn" : {  
      "@id" : "http://en.ifmo.ru/foundedIn",
      "@type" : "http://www.w3.org/2001/XMLSchema#date"
    },
    "locatedIn" : {
      "@id" : "http://en.ifmo.ru/locatedIn",
      "@type" : "@id"
    },
    "ex" : "http://example.org/",
    "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "xsd" : "http://www.w3.org/2001/XMLSchema#",
    "ifmo" : "http://en.ifmo.ru/"
  }  
}
```

https://json-ld.org/spec/latest/json-ld/
RDFS Vocabulary

- **RDFS позволяет:**
  - Объявлять классы с помощью `rdfs:Class`
  - Создавать экземпляры классов с помощью `rdf:type`

```xml
:University rdf:type rdfs:Class .
:ITMO rdf:type :University .
```
RDF Schema (RDFS)

RDFS Vocabulary
- rdfs:Class
- rdf:Property
- rdfs:range
- rdfs:domain

RDFS позволяет:
- Объявлять классы с помощью `rdfs:Class`
- Создавать экземпляры классов с помощью `rdf:type`

```
:University rdf:type rdfs:Class .
:ITMO rdf:type :University .
```

- Объявлять предикаты, их область определения и область значений с помощью `rdf:Property, rdfs:domain, rdfs:range`

```
:locatedIn rdf:type rdf:Property .
:locatedIn rdfs:range :Place .
:locatedIn rdfs:domain :University .
```
RDF Schema (RDFS)

RDFS Vocabulary
- rdfs:Class
- rdf:Property
- rdfs:range
- rdfs:domain

● RDFS позволяет:
  ○ Объявлять классы с помощью rdfs:Class
  ○ Создавать экземпляры классов с помощью rdf:type
    :University rdf:type rdfs:Class .
    :ITMO rdf:type :University .
  ○ Объявлять предикаты, их область определения и область значений с помощью rdf:Property, rdfs:domain, rdfs:range
    :locatedIn rdf:type rdf:Property .
    :locatedIn rdfs:range :Place .
    :locatedIn rdfs:domain :University .
RDF Schema (RDFS)

RDFS Vocabulary
rdfs:Class
rdf:Property
rdfs:range
rdfs:domain
rdfs:subClassOf
rdfs:subPropertyOf

● RDFS позволяет:
  ○ Создавать иерархию классов с помощью `rdfs:subClassOf`:
    ```
    :University rdfs:subClassOf :Institution .
    ```

  ○ Создавать иерархию предикатов с помощью `rdfs:subPropertyOf`:
    ```
    :city rdfs:subPropertyOf :locatedIn .
    ```
RDF Schema (RDFS)

RDFS Vocabulary

rdfs:Class
rdf:Property
rdfs:range
rdfs:domain
rdfs:subClassOf
rdfs:subPropertyOf
rdfs:label
rdfs:comment
rdfs:seeAlso
rdfs:isDefinedBy

- RDFS позволяет создавать аннотации (не участвуют в логическом выводе):
  - rdfs:label - человекочитаемое имя ресурса
    :ITMO rdfs:label “ITMO University”@en .
    :ITMO rdfs:label “Университет ИТМО”@ru .
  - rdfs:comment - текстовый комментарий
    :ITMO rdfs:comment “A university in Russia” .
  - rdfs:seeAlso - ссылка на объясняющий ресурс
    :ITMO rdfs:seeAlso :Universities_in_Russia .
OWL - Web Ontology Language

- OWL - еще более выразительное надмножество RDFS
  - Основан на дескрипционных логиках
  - OWL 1 (2004) \( SHOIN(D) \)
  - OWL 2 (2009) \( SROIQ(D) \)
  - Классы, предикаты и экземпляры классов (почти как в RDFS)
  - Гипотеза об открытом мире - отсутствие информации не говорит о ее истинности или ложности
    
    :Alice :knows :Bob .

    Не значит, что только Алиса знает Боба

  - Гипотеза уникальных имен не выполняется - различия нужно указывать отдельно
Логика первого порядка алгоритмически полуразрешима

Существует алгоритм, способный за конечное время подтвердить истинность некоторого высказывания в данной теории, а иначе может работать бесконечно долго.

Экспрессивность $\uparrow$ - разрешимость $\downarrow$

Получение разрешимых логик - хотя бы $O(e^n)$ - требует уменьшать экспрессивность

Дескрипционные логики позволяют получить довольно экспрессивные, но алгоритмически разрешимые теории
Логика первого порядка алгоритмически полуразрешима
○ Существует алгоритм, способный за конечное время подтвердить истинность некоторого высказывания в данной теории, а иначе может работать бесконечно долго.

Экспрессивность $\uparrow$ - разрешимость $\downarrow$

Получение разрешимых логик - хотя бы $O(e^n)$ - требует уменьшать экспрессивность

Дескрипционные логики позволяют получить довольно экспрессивные, но алгоритмически разрешимые теории

$\text{OWL EL, OWL RL, OWL QL} \subseteq \text{OWL 2 DL} \subseteq \text{OWL 2 Full} \subseteq \text{FOL}$
● **Аксиомы**
  ○ **TBox**: подклассы $C \sqsubseteq D$ (H)
  ○ **RBox**: иерархия предикатов $R \sqsubseteq S$ (H), инверсные предикаты $R^{-}$ (I), транзитивность предикатов $R^{+}$ (S)
  ○ **ABox**: факты о классах $C(a)$, предикатах $R(a, b)$, эквивалентность $(a=b)$, различие $(a \neq b)$

● **Составные классы**
  ○ Конъюнкция $C \sqcap D$, дизъюнкция $C \sqcup D$, отрицание $\neg C$ классов
  ○ Кванторы существования $\exists R.C$ и всеобщности $\forall R.C$
  ○ Ограничения на количество предикатов $\leq n R$, $\geq n R$ (N)
  ○ Номинальные классы $C = \{a\}$ (O)

● **Типы данных (D)**
Составные классы
- Имена классов $A$, $B$
- Конъюнкция $C \cap D$
- Дизъюнкция $C \cup D$
- Отрицание $\neg C$
- Кванторы существования $\exists R.C$
- Кванторы всеобщности $\forall R.C$
- Self $\exists S.Self$
- Количество предикатов $\leq n R.C (Q)$
- Количество предикатов $\geq n R.C (Q)$
- Номинальные классы $C = \{a\}$ (O)

Предикаты
- Имена предикатов $R$, $S$, $T$
- Простые предикаты $S$, $T$
- Инверсные предикаты $R^-$
- Универсальные предикаты $U$

Аксиомы о классах (TBox)
- Подклассы $C \subseteq D$
- Эквивалентность $C \equiv D$

Аксиомы о предикатах (RBox)
- Иерархия предикатов $R_1 \subseteq R_2$
- Цепи $R_-^1 \sqsubseteq R_-^2 \sqsubseteq \ldots \sqsubseteq R_-^n \subseteq R$
- Транзитивность $\sqsubseteq^+ (S)$
- Симметричность (R)
- Рефлексивность (R)
- Иррефлексивность (R)
- Несовместность

Факты (ABox)
- Факты о классах $C(a)$,
- Факты о предикатах $R(a, b)$,
- Негативные факты о предикате $\neg R(a, b)$
- Эквивалентность $(a=b)$,
- Различие $(a \neq b)$
Class: Musician

EquivalentTo:

(writes some Song)

and (writes only Song)
Musician ≡ ∃writes.Song ⊓ ∀writes.Song

:Musician rdf:type owl:Class ;
owl:equivalentClass [ owl:intersectionOf ( [ rdf:type owl:Restriction ;
owl:onProperty :writesSongs ;
owl:someValuesFrom :Song ]
[ rdf:type owl:Restriction ;
owl:onProperty :writesSongs ;
owl:allValuesFrom :Song ] ) ] .
Классы в OWL

● Простые или сложные (задаются аксиомой)
  :University rdf:type owl:Class

● Два уже определенных класса
  ○ owl:Thing - все экземпляры всех классов
  ○ owl:Nothing - пустое множество

● Сложные классы

  Musician ≡ ∃ writes.Song ⊓ ∀ writes.Song
Экземпляры в OWL

- Задаются через `owl:NamedIndividual`

```plaintext
:ITMO_University rdf:type :University .
:ITMO_University rdf:type owl:NamedIndividual .
```
Объектные: значение - ресурс

:hasStudent rdf:type owl:ObjectProperty ;
rdfs:range :Student ;
rdfs:domain :University .
Предикаты в OWL

- Объектные: значение - ресурс

```xml
:hasStudent rdf:type owl:ObjectProperty ;
rdfs:range :Student ;
rdfs:domain :University .
```

- Литеральные: значение - литерал

```xml
:foundedIn rdf:type owl:DatatypeProperty ;
rdfs:range xsd:NonNegativeInteger ;
rdfs:domain :University .
```
Объектные предикаты в OWL

- **Транзитивность**: $\text{TransitiveProperty}(p)$, $p(x,y) \land p(y,z) \models p(x,z)$

- **Симметричность**: $\text{SymmetricProperty}(p)$, $p(x,y) \models p(y,x)$

- **Асимметричность**: $\text{AsymmetricProperty}(p)$, $p(x,y) \not\models p(y,x)$

- **Рефлексивность**: $\text{ReflexiveProperty}(p) \models p(x,x)$
Объектные предикаты в OWL

- **Иррефлексивность**  
  \[ \text{IrreflexiveProperty}(p) \neq p(x,x) \]

- **Функциональность**  
  \[ \text{FunctionalProperty}(p), p(x,y) \sqcap p(x,z) \models y \equiv z \]

- **Инверсная функциональность**  
  \[ \text{InverseFunctionalProperty}(p), p(x,y) \sqcap p(z,y) \models x \equiv z \]

- **Несовместность**  
  \[ \text{propertyDisjointWith}(p,h), p(x,y) \sqcap h(x,z) \equiv \bot \]
OWL Restrictions & Rules

- Квалифицированные ограничения

\[
\text{SuccessfulAuthor} \sqsubseteq \geq 1 \text{notableWork.Bestseller}
\]

\[
:\text{SuccessfulAuthor} \text{ a } \text{owl:Class} ;
\]

\[
\text{rdfs:subClassOf} \ [
\text{a } \text{owl:Restriction;}
\text{owl:onProperty :notableWork;}
\text{owl:minQualifiedCardinality 1;}
\text{owl:onClass :Bestseller} \ ] .
\]
• Квалифицированные ограничения

\[
\text{SuccessfulAuthor} \sqsubseteq \geq 1 \text{notableWork.Bestseller}
\]

:SuccessfulAuthor a owl:Class ;
    rdfs:subClassOf [a owl:Restriction;
    owl:onProperty :notableWork;
    owl:minQualifiedCardinality 1;
    owl:onClass :Bestseller ] .

• Отношения над предикатами

\[
\text{hasMother}(x,y) \sqsubseteq \text{husbandOf}(z,y) \models \text{hasFather}(x,z)
\]

• Правила “если-то” (продукции)
  ○ Стандарты SWRL, SPIN
Вход: RDF-граф
Выход: RDF-граф с новыми триплетами
- Новые атрибуты вершин
- Новые ребра между вершинами

Логическое обоснование каждому введенному факту (чего не может статистическо-вероятностный ML)
Reasoning - логический вывод новых фактов

- Вход: RDF-граф
  Выход: RDF-граф с новыми триплетами
  ○ Новые атрибуты вершин
  ○ Новые ребра между вершинами

- Логическое обоснование каждому выведенному факту (чего не может статистическо-вероятностный ML)

- Время работы быстро растет от размера графа

- Актуальный ресерч:
  Объединить объяснимость символьных вычислений и скорость вероятностных
Граф знаний

ABox (Assertion Box)
Наполнение графа
Факты графа знаний

→ Создание как правило через семантическую интеграцию существующих источников (СУБД, неструктурированные данные) с помощью схемы графа

TBox (Terminology Box)
Схема данных графа
Модель предметной области

Онтология:

→ “формализованная модель некоторой области знаний, согласованная с экспертами этой области”

→ Итеративное (вручную) или автоматизированное создание
Крупнейшие онтологии:
- SNOMED-CT - медицина
- FIBO - финансы
- DBpedia Ontology - из Википедии
- YAGO - из Википедии
- Wikidata Ontology ~ из Википедии

Мета-онтологии:
- Dublin Core Terms (dcterms)
- SKOS
- DCAT
- RDF DataCube

https://lov.linkeddata.es/dataset/lov
Linked Open Data - 2020
Изначально неполная картина мира

Все, что не задано (как истина) - возможно, истина

Онтологии могут быть неполными by design для расширения

Задается все возможное знание

Все, что не задано (как истина) - ложь

Нужен источник истины, например

- Колонка в БД
- Поле объекта
- Слот во фрейме

<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>28</td>
</tr>
<tr>
<td>Bob</td>
<td>N/A</td>
</tr>
</tbody>
</table>
В следующей серии

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - Reification & RDF* & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Векторные представления графов - Knowledge Graph Embeddings
7. Введение в теорию графов - Graph Theory Intro
8. Машиноное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding