Unioeste - Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Docente: Adriana Postal

Atividade – Representação do conhecimento

Luiz Eduardo Garzon de Oliveira

Pedro Henrique de Oliveira Berti

CASCAVEL

2024

1. Representação de conhecimento escolhida:

Tipo de Representação: Rede Semântica

Redes semânticas são estruturas fundamentais na representação e organização do conhecimento dentro de sistemas computacionais. Elas consistem em um conjunto de nós, que representam conceitos ou entidades, conectados por arestas, que representam as relações entre esses conceitos.

No contexto de sistemas de recomendação, por exemplo, as redes semânticas podem ser utilizadas para mapear as interações entre, filmes, gêneros e atores, permitindo que o sistema ofereça sugestões mais precisas e personalizadas.

• 'Surgiram em 1968 quando Ross Quillian as usou para representar modelos psicológicos de memórias associativas'.
• 'É uma forma gráfica de representação de conhecimento, onde os objetos, conceitos ou situações no domínio são representados por nós e arcos'.
• 'Um conjunto de nós é conectado entre si através de um conjunto arcos, que representam as relações entre nós.' (SLIDE 48 do Documento 04_RepAqConhecimento.pdf disponibilizado no Teams)

Aplicação do Formalismo: A Rede Semântica foi aplicada para modelar e representar as relações entre filmes, gêneros, atores, diretores e estúdios, criando uma base de conhecimento para um sistema de recomendação de filmes. Usando RDF (Resource Description Framework) como a base para essa rede, foi possível criar um grafo semântico que captura as interações dos filmes com as demais propriedades. O RDF permite a integração de dados de várias fontes, como a API do TMDb, e facilita a inferência de novos relacionamentos entre as entidades representadas.

Fontes que embasaram o formalismo escolhido: O artigo 'The Semantic Web. Scientific American, 2001', destaca como a utilização de grafos RDF (Resource Description Framework) permite uma representação rica e flexível do conhecimento. Ele argumenta que a estrutura de grafo do RDF facilita a interconexão e o compartilhamento de dados complexos em um formato compreensível tanto por máquinas quanto por humanos.

O Artigo está disponivel em : https://lassila.org/publications/2001/SciAm.html

Após decidir pela utilização de Redes Semânticas com RDF, duas playlists no YouTube foram úteis para aprofundar meu conhecimento e auxiliar na implementação do projeto em Python:

Ontology rdflib python

Esta playlist foi fundamental para aprender a utilizar corretamente as Redes Semânticas com RDF em Python. Os vídeos apresentam tutoriais práticos que mostram como implementar grafos RDF, trabalhar com a biblioteca RDFlib, e integrar dados semânticos de forma eficaz.

Introdução a Ontologias e à Web Semântica (LIG948D)

Esta segunda playlist, por estar em português, foi extremamente útil para entender os conceitos fundamentais da Web Semântica e do RDF desde o início. A abordagem passo a passo, que inclui desde a teoria básica até a prática, ajudou a entender os conceitos e facilitou a aplicação no trabalho.

W3C. RDF - Resource Description Framework.

A leitura do documento do W3C sobre RDF (Resource Description Framework) foi fundamental para um entendimento mais aprofundado do RDF.

2. Descrição do problema:

Problema Escolhido: Sistema de Recomendação de Filmes.

O objetivo de um sistema de recomendação de filmes seria sugerir filmes aos usuários com base em seus interesses. A ideia principal é fornecer uma experiência personalizada para cada usuário de acordo com o seu gosto, interligando gênero, atores, diretor e estúdio.

Contexto: Com a vasta quantidade de conteúdo disponível em plataformas de streaming, os sistemas de recomendação se tornaram essenciais. Esses sistemas utilizam aprendizado de máquina, redes neurais e, no caso deste projeto, uma Rede Semântica, para analisar dados e identificar padrões nas preferências dos usuários. Com isso, é possível fazer sugestões precisas, melhorando a satisfação do usuário.

Relevância do problema: A relevância do problema para a aplicação da representação de conhecimento está na necessidade de capturar e estruturar as relações entre filmes de maneira mais precisa. Ao utilizar uma Rede Semântica, podemos representar não apenas os filmes como entidades isoladas, mas também as conexões entre gêneros, diretores e atores, permitindo uma compreensão mais profunda e uma recomendação mais acertada. Isso torna o sistema capaz de inferir e sugerir filmes de maneira mais 'inteligente'.

Exemplos e fontes da literatura que foram utilizados como base: O Trabalho "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." oferece uma visão abrangente sobre os avanços nos sistemas de recomendação e discute a importância de incorporar informações contextuais e semânticas nas recomendações. Ele serve como base para justificar o uso de Redes Semânticas no aprimoramento da precisão das recomendações de filmes.

3. Metodologia:

3.1. O primeiro passo foi escolher as Entidades Principais. As Entidades escolhidas foram:

- **Filme:** Representa os filmes disponíveis no sistema. Cada filme é tratado como uma instância única com atributos como título, gênero, elenco, diretor e estúdio de produção. No código, os filmes são extraídos da API do TMDb, e cada filme é instanciado na Rede Semântica usando o título como identificador.

- **Gênero:** Representa as categorias de filmes, como "Ação", "Comédia", "Drama", etc. Cada filme pode pertencer a um ou mais gêneros, que são instanciados como entidades separadas na rede.

- **Ator:** Representa os atores que atuam nos filmes. Cada ator é uma entidade única na Rede Semântica, ligada aos filmes nos quais atuou.

- **Diretor:** Representa os diretores responsáveis pela direção dos filmes. Assim como os atores, cada diretor é instanciado como uma entidade separada na Rede Semântica.

- **Estúdio:** Representa as companhias de produção responsáveis por produzir os filmes. Cada estúdio é uma entidade separada e está associado aos filmes que produziu.

3.2. O Segundo passo foi definir a relação entre as entidades

As relações entre as entidades são modeladas como propriedades na Rede Semântica, definindo como essas entidades estão conectadas entre si, essas relações são estabelecidas através de tripletas RDF (sujeito, predicado, objeto):

3.2.1. Filme "pertence ao" Gênero: (ns.pertenceAoGenero)

- **Sujeito**: Filme (URI do filme)
- **Predicado**: pertenceAoGenero (relação que conecta o filme ao gênero)
- **Objeto**: Gênero (URI do gênero)
- **Descrição**: Esta relação define a qual gênero (ou gêneros) um filme pertence.

3.2.2. Filme "tem" Ator: (ns.temAtor)

- **Sujeito**: Filme (URI do filme)
- **Predicado**: temAtor (relação que conecta o filme ao ator)
- **Objeto**: Ator (URI do ator)
- **Descrição**: Esta relação especifica quais atores atuam em um filme específico.

3.2.3. Ator "atuou em" Filme: (ns.atuouEm)

- **Sujeito**: Ator (URI do ator)
- **Predicado**: atuouEm (relação que conecta o ator ao filme)
- **Objeto**: Filme (URI do filme)
- **Descrição**: Esta relação é a inversa da relação "temAtor", indicando em quais filmes o ator atuou.

3.2.4. Filme "dirigido por" Diretor: (ns.dirigidoPor)

- **Sujeito**: Filme (URI do filme)
- **Predicado**: dirigidoPor (relação que conecta o filme ao diretor)
- **Objeto**: Diretor (URI do diretor)
- **Descrição**: Esta relação define qual diretor foi responsável pela direção do filme.

3.2.5. Filme "produzido pelo" Estúdio: (ns.produzidoPeloEstudio)

- **Sujeito**: Filme (URI do filme)
- **Predicado**: produzidoPeloEstudio (relação que conecta o filme ao estúdio)
- **Objeto**: Estúdio (URI do estúdio)
- **Descrição**: Esta relação indica qual estúdio foi o responsável pela produção do filme.

Essas entidades e relações formam a base da Rede Semântica, permitindo que o sistema de recomendação conecte e inferencie informações relevantes para fornecer recomendações de filmes.

3.3. O terceiro passo foi Escolher a Linguagem/Ferramenta para Implementação

A linguagem escolhida foi python, a escolha do Python se deu muito em conta pela biblioteca RDFlib, devido a uma série de vantagens que essa combinação oferece para o desenvolvimento da Rede Semântica.

A RDFlib é uma biblioteca eficiente e intuitiva para trabalhar com RDF em Python. Ela facilita a criação, manipulação e consulta de grafos semânticos, permitindo que entidades e relações sejam definidas de forma estruturada e interconectada.

Utilizando RDF (Resource Description Framework), é possível representar o conhecimento através de tripletas que seguem a estrutura de sujeito-predicado-objeto. As entidades como filmes, atores, gêneros, diretores e estúdios são representadas como URIs (Uniform Resource Identifiers), e as relações entre essas entidades são modeladas como propriedades RDF, como “pertence ao gênero” ou “atuou em”.

A metodologia adotada permite que novas entidades e relações sejam facilmente adicionadas ao sistema, tornando-o altamente extensível.

3.4. O quarto passo foi preparar o ambiente.

Para instalar a biblioteca RDFLib, foi necessário configurar e ativar um ambiente virtual no sistema Linux Ubuntu. O processo foi realizado com os seguintes comandos:

Criar o Ambiente Virtual: virtualenv venv
Esse comando cria um ambiente virtual isolado dentro do diretório do projeto.

Ativar o Ambiente Virtual: source venv/bin/activate
Após a ativação, o ambiente virtual estará pronto para a instalação de dependências sem interferir nas configurações globais do sistema.

Com o ambiente virtual ativo, a instalação da biblioteca RDFLib foi realizada com o comando: pip install rdflib

Após a configuração do ambiente, foi escolhido o Visual Studio Code (VS Code) como editor de código.

Para a coleta de dados, utilizou-se a API do TMDb (The Movie Database). A TMDb oferece um token de API gratuito para estudantes ou projetos com fins acadêmicos, permitindo o acesso a uma base de dados de filmes, diretores, atores, e outros elementos.

Com o ambiente configurado e a API integrada, o arquivo Python foi desenvolvido e nomeado `redesemantica.py`. Para executar o script e gerar a Rede Semântica, o seguinte comando foi utilizado: python3 redesemantica.py

Ao final da execução, o código sinaliza a criação bem-sucedida do arquivo RDF/XML com a mensagem:"Rede salva com sucesso"

Para visualizar a Rede Semântica em formato de grafo, foi utilizado o RDF Grapher, uma ferramenta online que permite carregar e explorar arquivos RDF/XML. Alternativamente, qualquer outro visualizador compatível pode ser usado para a mesma finalidade.

Outras ferramentas como Webprotegé podem ser úteis, para inspecionar detalhadamente as classes, propriedades e instâncias da Rede Semântica.

3.5. Ferramentas Utilizadas

- Python
- VirtualEnv (Nativo Linux Ubuntu)
- RDFLib
- Visual Studio Code (VS Code)
- TMDb API
- RDF Grapher
- Terminal (Nativo Linux Ubuntu)

4. Representação do conhecimento:

Para fins de exemplificação, a Rede Semântica disponível em PDF foi construída utilizando apenas 2 filmes, com 1 ator e 1 estúdio por filme. Essa abordagem foi adotada para manter a simplicidade e facilitar a compreensão, mas é importante destacar que a Rede Semântica pode ser expandida para incluir quantos filmes, atores, estúdios, e outras entidades forem necessários, dependendo da complexidade desejada.

4.1. Entidades Principais e Suas Classes:

4.1.1. Filmes (Filme):

Instâncias:

- Deadpool & Wolverine
- DivertidaMente 2

Descrição: Cada filme é descrito por suas relações com gêneros, atores, diretores e estúdios.

4.1.2. Gêneros (Genero):

Instâncias:

- Ação
- Comédia
- Ficção científica
- Animação
- Família
- Aventura

Descrição: A classe Genero categoriza os filmes com base em temas ou estilos, como ação, comédia, e animação.

4.1.3. Atores (Ator):

Instâncias:

- Ryan Reynolds

- Amy Poehler

Descrição: A classe Ator representa os indivíduos que desempenham papéis em filmes. Eles estão conectados aos filmes nos quais atuaram.

4.1.4. Diretores (Diretor):

Instâncias:

- Shawn Levy
- Kelsey Mann

Descrição: A classe Diretor identifica os indivíduos responsáveis pela direção dos filmes.

4.1.5. Estúdios (Estúdio):

Instâncias:

- Marvel Studios
- Walt Disney Pictures

Descrição: A classe Estúdio representa as companhias que produzem os filmes.

4.2. Propriedades (Relações) Entre as Entidades:

4.2.1. Propriedade pertenceAoGenero:

- **Domínio:** Filme
- **Alcance:** Genero

Descrição: Conecta um filme aos gêneros aos quais pertence.

Exemplo de Uso:

- Deadpool & Wolverine pertence aos gêneros Ação, Comédia, e Ficção científica.
- DivertidaMente 2 pertence aos gêneros Animação, Família, Aventura, e Comédia.

4.2.2. Propriedade temAtor:

- **Domínio:** Filme
- **Alcance:** Ator

Descrição: Conecta um filme aos atores que atuaram nele.

Exemplo de Uso:

- Deadpool & Wolverine tem como ator Ryan Reynolds.
- DivertidaMente 2 tem como atriz Amy Poehler.

4.2.3. Propriedade dirigidoPor:

- **Domínio:** Filme
- **Alcance:** Diretor

Descrição: Conecta um filme ao diretor que o dirigiu.

Exemplo de Uso:

- Deadpool & Wolverine foi dirigido por Shawn Levy.
- DivertidaMente 2 foi dirigido por Kelsey Mann.

4.2.4. Propriedade produzidoPeloEstudio:

- **Domínio:** Filme
- **Alcance:** Estúdio

Descrição: Conecta um filme ao estúdio que o produziu.

Exemplo de Uso:

- Deadpool & Wolverine foi produzido pelo estúdio Marvel Studios.
- DivertidaMente 2 foi produzido pelo estúdio Walt Disney Pictures.

4.2.5. Propriedade atuouEm:

- **Domínio:** Ator
- **Alcance:** Filme
- **Descrição:** Conecta um ator ao filme em que atuou.

Exemplo de Uso:

- Ryan Reynolds atuou em Deadpool & Wolverine.
- Amy Poehler atuou em DivertidaMente 2.

4.3. Estrutura do Grafo:

- **Nodos (Nós):** Representam as entidades principais (filmes, gêneros, atores, diretores, estúdios).
- **Arestas (Conexões):** Representam as propriedades que ligam os nós, descrevendo como essas entidades estão relacionadas.

4.4. Como a representação resolve o problema proposto

A Rede Semântica implementada resolve o problema proposto de construir um sistema de recomendação de filmes ao criar uma estrutura de conhecimento que captura de forma detalhada as relações entre diferentes elementos, como filmes, gêneros, atores, diretores e estúdios. Essa estrutura, baseada em RDF (Resource Description Framework), permite representar e explorar as conexões entre esses elementos, proporcionando uma base sólida para a geração de recomendações.

4.4.1. Flexibilidade e Expansão

A estrutura em grafo da Rede Semântica permite que novos filmes, atores, gêneros e outras entidades sejam facilmente adicionados à rede, mantendo a flexibilidade necessária para acomodar expansões futuras.

4.4.2. Conclusão

Ao representar o conhecimento sobre filmes, gêneros, atores, diretores e estúdios de forma estruturada e interconectada, a Rede Semântica resolve de maneira eficaz o problema de recomendar filmes de forma personalizada e relevante. Ela oferece uma base flexível, expansível e capaz de integrar e inferir dados, tornando o sistema de recomendação não apenas funcional, mas também adaptável às necessidades e preferências ao longo do tempo.

4.5. Código Python

4.5.1 Importações e Configuração

```
import requests
from rdflib import Graph, Namespace, RDF, RDFS, URIRef
```

- **requests**: Biblioteca para fazer requisições HTTP.
- **rdflib**: Biblioteca para trabalhar com grafos RDF (Resource Description Framework).

4.5.2. Configurações da API

```
API_KEY = '5f3d37b7db5f82624c4baeffc563be36'
BASE_URL = 'https://api.themoviedb.org/3'
```

- **API_KEY:** Chave API para acessar o TMDb, (Necessário se cadastrar no TMDb API como estudante para receber uma)
- **BASE_URL:** URL base para as requisições da API TMDb.

4.5.3. Criação do Grafo RDF

```
g = Graph()
ns = Namespace("http://example.org/")
```

- **g**: Criação de um novo grafo RDF.
- **ns**: Definição de um namespace para as suas entidades RDF.

4.5.4. Definição das Classes

```
g.add((ns.Filme, RDF.type, RDFS.Class))
g.add((ns.Genero, RDF.type, RDFS.Class))
g.add((ns.Ator, RDF.type, RDFS.Class))
g.add((ns.Diretor, RDF.type, RDFS.Class))
g.add((ns.Estúdio, RDF.type, RDFS.Class))
```

- Adiciona Classes ao grafo.

4.5.5. Definição das Propriedades

```
g.add((ns.pertenceAoGenero, RDF.type, RDF.Property))
g.add((ns.pertenceAoGenero, RDFS.domain, ns.Filme))
g.add((ns.pertenceAoGenero, RDFS.range, ns.Genero))

g.add((ns.temAtor, RDF.type, RDF.Property))
g.add((ns.temAtor, RDFS.domain, ns.Filme))
g.add((ns.temAtor, RDFS.range, ns.Ator))

g.add((ns.atuouEm, RDF.type, RDF.Property))
g.add((ns.atuouEm, RDFS.domain, ns.Ator))
g.add((ns.atuouEm, RDFS.range, ns.Filme))

g.add((ns.dirigidoPor, RDF.type, RDF.Property))
g.add((ns.dirigidoPor, RDFS.domain, ns.Filme))
g.add((ns.dirigidoPor, RDFS.range, ns.Diretor))

g.add((ns.produzidoPeloEstudio, RDF.type, RDF.Property))
g.add((ns.produzidoPeloEstudio, RDFS.domain, ns.Filme))
g.add((ns.produzidoPeloEstudio, RDFS.range, ns.Estúdio))
```

- Define propriedades que relacionam as classes.

4.5.6. Funções de Busca

```
def buscar_filmes_populares(pagina=1):
    url = f"{BASE_URL}/movie/popular"
    params = {
        'api_key': API_KEY,
        'language': 'pt-BR',
        'page': pagina
    }
    resposta = requests.get(url, params=params)
    return resposta.json()
```

- **buscar_filmes_populares**: Obtém uma lista de filmes da API do TMDb. Sempre pega os filmes em ordem de popularidade.

```python
def buscar_detalhes_filme(filme_id):
    url = f"{BASE_URL}/movie/{filme_id}"
    params = {
        'api_key': API_KEY,
        'language': 'pt-BR'
    }
    resposta = requests.get(url, params=params)
    return resposta.json()
```

- **buscar_detalhes_filme**: Obtém detalhes de um filme específico usando o ID do filme.

```python
def buscar_elenco_filme(filme_id):
    url = f"{BASE_URL}/movie/{filme_id}/credits"
    params = {
        'api_key': API_KEY,
        'language': 'pt-BR'
    }
    resposta = requests.get(url, params=params)
    return resposta.json()
```

- **buscar_elenco_filme**: Obtém informações sobre o elenco e a equipe de um filme.

4.5.7. Coleta de Dados e Construção do Grafo

```
filmes = buscar_filmes_populares(pagina=1)['results']

for filme in filmes[:2]:
    filme_id = filme['id']
    detalhes = buscar_detalhes_filme(filme_id)
    elenco = buscar_elenco_filme(filme_id)

    filme_uri = URIRef(ns[detalhes['title'].replace(" ", "")])
    g.add((filme_uri, RDF.type, ns.Filme))

    for genero in detalhes['genres']:
        genero_uri = URIRef(ns[genero['name'].replace(" ", "")])
        g.add((genero_uri, RDF.type, ns.Genero))
        g.add((filme_uri, ns.pertenceAoGenero, genero_uri))

    for membro in elenco['cast'][:1]:
        ator_uri = URIRef(ns[membro['name'].replace(" ", "")])
        g.add((ator_uri, RDF.type, ns.Ator))
        g.add((filme_uri, ns.temAtor, ator_uri))
        g.add((ator_uri, ns.atuouEm, filme_uri))

    for membro in elenco['crew']:
        if membro['job'] == 'Director':
            diretor_uri = URIRef(ns[membro['name'].replace(" ", "")])
            g.add((diretor_uri, RDF.type, ns.Diretor))
            g.add((filme_uri, ns.dirigidoPor, diretor_uri))
            break

    if detalhes.get('production_companies'):
        estúdio = detalhes['production_companies'][0]
        estúdio_uri = URIRef(ns[estúdio['name'].replace(" ", "")])
        g.add((estúdio_uri, RDF.type, ns.Estúdio))
        g.add((filme_uri, ns.produzidoPeloEstudio, estúdio_uri))
```

- Obtém uma lista de filmes populares e itera sobre eles, Para cada filme, coleta detalhes e informações sobre o elenco, cria instâncias no grafo para cada filme, gênero, ator, diretor e estúdio, e adiciona as propriedades.

4.5.8. Salvamento do Grafo

```
g.serialize(destination="redesemantica.rdf", format="xml")
```

- Salva o grafo.

4.5.9. Feedback

```
print("rede salva com sucesso")
```

- Imprime uma mensagem de feedback ao usuário.

5. Análise e discussão:

5.1. Eficácia da Representação:

A utilização de uma Rede Semântica com RDF foi muito eficaz na captura e organização do conhecimento necessário para um sistema de recomendação de filmes. A estrutura em grafo permitiu modelar com precisão as relações entre filmes, gêneros, atores, diretores e estúdios, oferecendo uma base robusta para a inferência e personalização das recomendações. Além disso, a flexibilidade do RDF possibilitou a integração de dados e fácil expansão da rede, garantindo que o sistema possa evoluir.

5.2. Limitações Encontradas:

A coleta de dados em tempo real pode ser limitada pela disponibilidade e qualidade das APIs utilizadas. A complexidade de inferência também aumenta conforme o grafo cresce em tamanho e detalhes.

5.3. Melhorias Futuras:

Adicionar mais detalhes aos dados, como críticas e avaliações de filmes, e explorar técnicas de inferência semântica para aumentar a precisão das recomendações.

6. Referências:

04_RepAqConhecimento.pdf

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 6, p. 734–749, jun. 2005.

Getting Started. Disponível em: <https://developer.themoviedb.org/docs/getting-started>.

HENDLER, J.; LASSILA PHOTOILLUSTRATIONS, O.; SALMERON, M. **THE WEB SEMANTIC**. [s.l: s.n.].

Introdução a Ontologias e à Web Semântica (LIG948D). Disponível em: <http://www.youtube.com/playlist?list=PLLrlHSmC0Mw6lEj060psXrt3BSATBFVzV>.

MICROSOFT. **Visual Studio Code.** Disponível em: <https://code.visualstudio.com/>.

Ontology rdflib python. Disponível em: <http://www.youtube.com/playlist?list=PLJ-sIDAvcXl7VaU-AR27Kwiw52Q9UTDEn>.

LUGER, G. F. **Artificial Intelligence.** [s.l.] Pearson Higher Ed, 2011.

.PYTHON. **Python**. Disponível em: <https://www.python.org/>.

RDF Grapher. Disponível em: <https://www.ldf.fi/service/rdf-grapher>.

rdflib 7.0.0 — rdflib 7.0.0 documentation. Disponível em: <https://rdflib.readthedocs.io/en/stable/#>.

RDF - Semantic Web Standards. Disponível em: <https://www.w3.org/RDF/>.

The Movie Database. Disponível em: <https://www.themoviedb.org/?language=pt-BR>.

WebProtégé. Disponível em: <https://webprotege.stanford.edu/>