UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

JOSE WELLINGTON FRANCO DA SILVA

UM FRAMEWORK BASEADO EM CONHECIMENTO DE SENSO COMUM PARA SISTEMAS DE PERGUNTAS E RESPOSTAS SOBRE GRAFO DE CONHECIMENTO

FORTALEZA

2022

JOSE WELLINGTON FRANCO DA SILVA

UM FRAMEWORK BASEADO EM CONHECIMENTO DE SENSO COMUM PARA SISTEMAS DE PERGUNTAS E RESPOSTAS SOBRE GRAFO DE CONHECIMENTO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Bancos de Dados

Orientadora: Profª. Drª. Vânia Maria Ponte Vidal

Coorientadora: Profª. Drª. Vládia Célia Monteiro Pinheiro

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58f Silva, José Wellington Franco da.
Um Framework baseado em Conhecimento de Senso Comum para Sistemas de Perguntas e Respostas sobre Grafo de Conhecimento / José Wellington Franco da Silva. – 2022.
124 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2022.
Orientação: Profa. Dra. Vânia Maria Ponte Vidal.
Coorientação: Profa. Dra. Vládia Célia Monteiro Pinheiro.

1. Sistemas de Perguntas e Respostas. 2. Templates. 3. Conhecimento de Senso Comum. 4. Grafo de Conhecimento. I. Título.

CDD 005

JOSE WELLINGTON FRANCO DA SILVA

UM FRAMEWORK BASEADO EM CONHECIMENTO DE SENSO COMUM PARA SISTEMAS DE PERGUNTAS E RESPOSTAS SOBRE GRAFO DE CONHECIMENTO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Bancos de Dados

Aprovada em: 17/02/2022

BANCA EXAMINADORA

Profª. Drª. Vânia Maria Ponte Vidal (Orientadora)
Universidade Federal do Ceará (UFC)

Profª. Drª. Vládia Célia Monteiro Pinheiro (Coorientadora)
Universidade de Fortaleza (UNIFOR)

Prof. Dr. José Maria da Silva Monteiro Filho
Universidade Federal do Ceará (UFC)

Prof. Dr. Angelo Roncalli Alencar Brayner
Universidade Federal do Ceará (UFC)

Prof. Dr. Luciano de Andrade Barbosa
Universidade Federal de Pernambuco (UFPE)

À Lucia e Nelito, meus pais. Ao Rafael, meu irmão.

AGRADECIMENTOS

Agradeço em primeiro lugar a DEUS pelo o dom da vida, por todas as oportunidades que já tive em minha vida. Obrigado Senhor, por me guiar e me iluminar em todos os momentos da minha vida.

Aos meus pais, Nelito e Lucia que me educaram para vida, que me ensinaram as primeiras palavras, e que diversas vezes abriram mão de seus sonhos para que não faltassem nada para mim e para meu irmão. Ao meu irmão Rafael, por sempre iluminar a minha vida. Tenho certeza que você é um anjo que Deus enviou para nossa família. Sem você eu não teria chegado até aqui. Agradeço também a todos tios, tias, primos e primas que me incentivaram durante esse período.

À minha orientadora Vânia Vidal, pelo cuidado, atenção e confiança durante o desenvolvimento desse trabalho e por ter aceitado o convite para me orientar.

À minha co-orientadora Vládia Pinheiro, por ter aceitado me co-orientar. Obrigado também por todo o apoio durante o desenvolvimento do trabalho e pelas valiosas contribuições durante todo o doutorado.

Aos meus amigos e parceiros de pesquisa, Professor Gilvan Maia, Artur Franco, Caio Àvila, Lucas Cabral e Matheus Mayron por todo apoio e coloboração no meu trabalho de doutorado. Espero que nossa parceria em pesquisa de prolongue por muito tempo.

Aos amigos do grupo Arida, por sempre estarem prontos para ajudar em qualquer dificuldade. Dentre eles, destaco os amigos: Narciso Arruda, Túlio Vidal, Tiago Vinnuto, Arlino Henrique, Ernando Gomes, Mardson Ferreira, Salomão Santos, Nickson, Gabriel, Bruno pelo o apoio durante o doutorado.

Aos amigos e amigas da Universidade Federal do Ceará em Crateús pela oportunidade do afastamento para se dedicar ao doutorado e todo o apoio durante o período. Em especial destaco os seguites colegas: Amanda Drielly, Lívio Freire, Rennan Dantas, Fabiana Tomaz, Emerson Barros, Marciel Barros, Felipe Brasil, Anderson Almada, Italo Mendes, Simone Santos e tantos outros.

Ao Professor José Maria da Silva Monteiro Filho, por ter aceitado participar da minha banca e por todos os valorosos conselhos durante o doutorado.

Ao Professor Ângelo Roncalli Alencar Brayner, por ter aceitado participar da minha banca e pela presteza e paciência em contribuir com esse trabalho.

Aos Professores Luciano de Andrade Barbosa e Fábio André Machado Porto, por ter

aceitado participar da minha banca e pela prestatividade em contribuir de forma enriquecedora para esse trabalho.

Ao meu amigo, Regis Torquato pelo apoio e ajuda na revisão do texto da tese.

Agradeço também aos diversos amigos e amigas que se preocuparam e me incentivaram durante esse processo. Vale a pena ressaltar os amigos e amigas do Centro Inaciano de Juventude e a Pré-Comunidade de Vida Cristã Nossa Senhora da Estrada. Em especial os amigos e amigas Raquel Vieira, Italo Gois, Marcela Cavalcante, Rafael Batista, Anderson Albuqueque, Denis Valone, Evenice Neta, Juliano Efson, Macedo Maia, Fabiano Tavares, Paulo Alexandre, Douglas Nogueira e todos os outros que não cito aqui.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Por fim, a todos que de alguma forma passaram na minha vida durante esse tempo de crescimento e aprendizado.

“Não é o muito saber que sacia e satisfaz, mas o sentir e saborear internamente as coisas.”

(Santo Inácio de Loyola)

RESUMO

As diferenças entre a linguagem natural não estruturada e os dados estruturados dificultam a construção de um sistema computacional que suporte o uso da Linguagem Natural para consultar Grafos de Conhecimento. Uma alternativa para esse problema é recorrer a sistemas de Question-Answering (QA) na realização de consultas com o intuito de criar pontes entre os dados e as questões em linguagem natural. Esses sistemas são acessíveis a usuários sem conhecimento técnico quanto às informações em Grafos de Conhecimento, possibilitando o seu uso sem a necessidade de experiência em linguagens de consulta ou *schemas* complexos. Contudo, o uso desses sistemas apresenta alguns problemas, principalmente limitações nas variações observadas na linguagem natural, bem como uma forte dependência de interferência humana. Este trabalho propõe um *framework* completo para a consulta em linguagem natural sobre Grafos de Conhecimento baseado em *templates* construídos com o auxílio de conhecimento de senso comum. O *framework* é composto por (i) um sistema de QA baseado em *templates*, (ii) um conjunto de dados de QA over Knowledge Graphs (KGQA) com várias perguntas parafraseadas usando conhecimento de senso comum e (iii) conjunto de algoritmos e técnicas em cada módulo do sistema que podem ser aproveitadas em outras abordagens. São os principais diferenciais do nosso *framework* (a) a forma com que uma base de conhecimento de senso comum é usada para melhorar a variabilidade do conjunto de dados gerados e (b) o uso de um sistema *Open IE* para conferir maior qualidade nos *templates* gerados. Ademais, propomos uma técnica de indexação para recuperação da resposta à pergunta submetida ao sistema de QA e uma metodologia experimental para verificar a qualidade desse tipo de sistema. Os resultados obtidos neste trabalho fornecem contribuições para o estado-da-arte de sistemas de QA sobre QA over Knowledge Bases (KBQA). Por fim, realizamos apontamentos para pesquisas futuras.

Palavras-chave: sistemas de perguntas e respostas baseado em *templates*; conhecimento de senso comum; grafos de conhecimento.

ABSTRACT

The differences between unstructured natural language and structured data make it challenging to build a computational system that supports the use of Natural Language to query Knowledge Graphs. An alternative to this problem is to use QA systems to perform queries to make this connection between data and questions in natural language. Therefore, these systems are accessible to users who do not have technical knowledge in accessing information in Knowledge Graphs, thus eliminating the need to learn complex query languages and *schemas*. However, using these systems presents some problems, mainly limitations in the variations observed in the natural language and the strong dependence on human interference. This work proposes a complete *framework* for querying Knowledge Graphs in natural language based on *templates* built with the help of common sense knowledge. This *framework* is composed of a QA system based on *templates* and a dataset of KGQA, along with a set of algorithms and techniques in each module of the system that can use in other approaches. Finally, ExQuestions, a question and answer dataset with multiple questions paraphrased using common sense knowledge. The main differences of our *framework* are: (a) how a common-sense knowledge base is used to improve the variability of the generated dataset and (b) the use of a *Open IE system* to improve the quality of the generated *templates*. Furthermore, we propose an indexing technique to retrieve the answer to the question submitted to the QA system and an experimental methodology to verify the quality of this type of system. The results obtained in this work provide contributions to the state-of-the-art of QA systems on KBQA and, finally, we make notes for future research.

Keywords: question answering systems; query templates; common dense; knowledge graph.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMT	Amazon Mechanical Turk
ANN	Approximate Nearest Neighbors
ANNOY	Approximate Nearest Neighbors Oh Yeah
API	Applications Protocol Interface
BSP	Binary Space Partition
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HNSW	Hierarchical Navigable Small World Graphs
KB	Knowledge Base
KBQA	QA over Knowledge Bases
KG	Knowledge-Graph
KGQA	QA over Knowledge Graphs
LOD	Linked Open Data
LSTM	Long Short-Term Memory
MRR	Mean Reciprocal Rank
NGT	Neighborhood Graph and Tree for Indexing Highdimensional Data
NLI	Natural Language Interfaces
NLP	Natural Language Processing
NLPG	Natural Language Pattern Generation
NLQ	Natural Language Query
QA	Question-Answering
QAC	Community Question Answering
QACR	QA for Comprehension Reading
QADR	Closed-Domain Question Answering
QALD	Question Answering over Linked Data
QALOD	Linked Open Data Question Answering
QASOBO	QA Over Domain Ontologies, Ontology-Based QA
QF	Query Formulation
QM	Question Matching
RDF	Resource Description Framework

ROC	Receiver Operating Characteristic
SAT	The Satisfiability Problem
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
TBQA	Template-Based Question Answering
TE	Template Expansion

SUMÁRIO

1 INTRODUÇÃO

Com o advento dos Grafos de Conhecimento (*Knowledge Graphs*, KG), surgiram vários trabalhos que procuram desenvolver estudos que visam acessar os dados presentes nos KG de forma rápida e consistente (SILVA *et al.*, 2020; DIEFENBACH *et al.*, 2017; LATIFI, 2018).

De acordo com Bizer *et al.* (2009), os KG apresentam uma grande quantidade de dados e consequentemente uma certa complexidade em acessar os dados. Pois os mesmos, demandam um conhecimento especializado em linguagens de consulta — por exemplo, SPARQL Protocol and RDF Query Language (SPARQL) — quanto uma profunda familiaridade com as ontologias subjacentes.

1.1 Motivação

Diversos trabalhos (KAUFMANN; BERNSTEIN, 2010; ZOU *et al.*, 2014; ZHENG *et al.*, 2015; XU *et al.*, 2016) propuseram sistemas de *Perguntas e Respostas* (*Question and Answer Systems*, QA Systems) que buscam acessar os dados presentes no KG por meio de perguntas expressas em linguagem natural.

Para Dubey *et al.* (2019), os sistemas de QA sobre grafos de conhecimentos — do Inglês, KGQA — têm como principal objetivo recuperar informações factuais[1] contidas nesses grafos de conhecimento.

Para exemplificar como funciona um sistema de QA, considere a pergunta sobre um KG: *Quem é a esposa do Obama?*. O sistema de QA deve buscar as entidades e relações presentes na pergunta para a construção de uma consulta que expresse uma resposta adequada. No caso desse exemplo, encontramos o objeto/entidade *Obama* e a relação/predicado *é esposa de*. Com essas informações extraídas, o sistema de QA constrói e executa a consulta sobre o KG, produzindo como retorno o sujeito procurado: *Michele Obama*. Na Figura 1, construímos uma representação que mostra o funcionamento do QA baseado nesse mesmo exemplo.

Apesar de esse processo inicial se mostrar como algo aparentemente simples, há uma lacuna a ser superada entre a base de conhecimento estruturada e a pergunta em linguagem natural. A maioria dos métodos existentes usa a estrutura morfossintática da pergunta para gerar uma consulta e, em seguida, executar a correspondência de subgrafo sobre o grafo de conhecimento. Se a semântica da consulta gerada estiver incorreta, é muito provável que os

[1] Segundo o Dicionário Priberam, factual é um adjetivo que associa a informação a um fato ou baseado em fatos, ou seja, trata-se de informação cuja veracidade ou existência está comprovada.

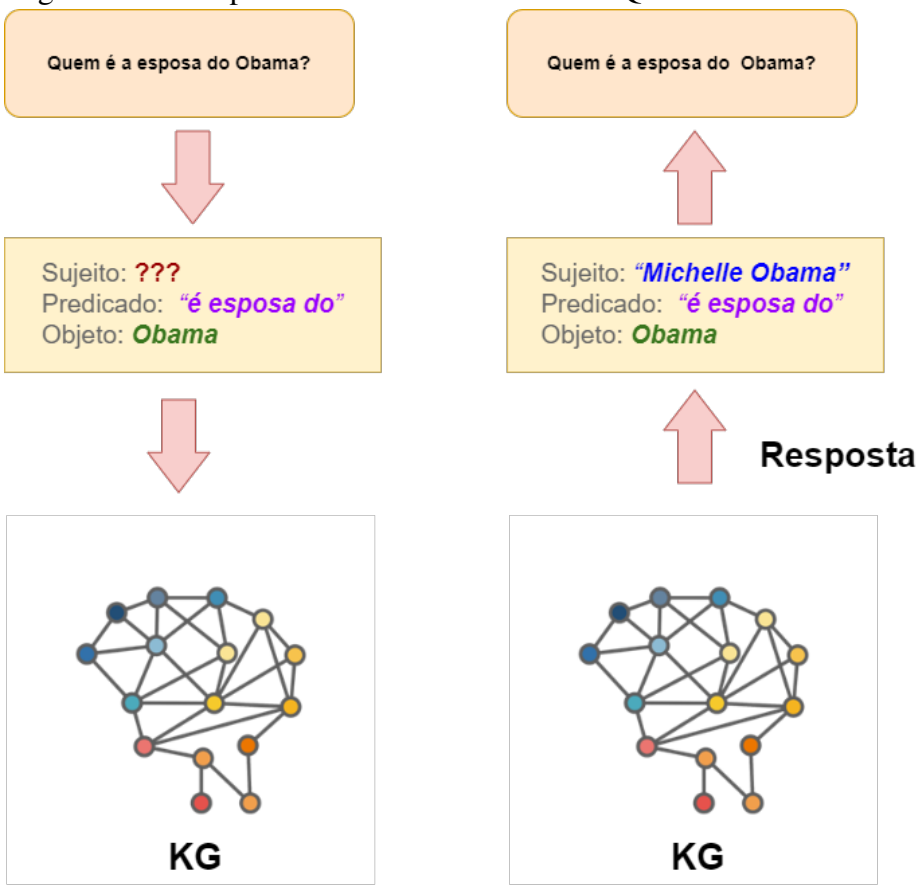
métodos encontrem respostas falsas ou incompletas (SILVA *et al.*, 2020). A seguir, apresentamos os principais desafios existentes para sistemas de QA.

1.2 Principais Desafios

De acordo com Dubey *et al.* (2016), Cai e Yates (2013), as abordagens usadas nos primeiros sistemas de QA utilizavam regras ou modelos de aprendizado de máquina. Observamos essas abordagens na literatura em relação a aspectos ligados à elaboração, construção, validação e evolução dos sistemas de QA. Diversos desafios se apresentam a partir disso, dos quais destacam-se:

– *Lacuna Léxica*: para Diefenbach *et al.* (2018), a diferença de entre os termos adotados em uma pergunta e à rotulagem encontrada na base de conhecimento dificulta a associação pelos sistemas de QA. Por exemplo, as expressões *"X escreveu Y"* e *"Y foi escrito por X"* identificam a mesma relação entre X e Y usando termos diferentes, o que exige maior sofisticação ao lidar com a associação subjacente;

Figura 1 – Exemplo do funcionamento de uma QA sobre um KG



Fonte: Elaborado pelo autor (2021).

– *Lacuna Semântica*: os termos (entidades e predicados) presentes nas consultas estruturadas devem ser os mesmos presentes no KG para que a consulta seja executada de forma correta. Infelizmente, nem sempre isso é possível devido à variação semântica presente na linguagem natural. Por exemplo, na consulta *"Liste todos os cosmonautas"*, a entidade *"cosmonautas"* não seria encontrada em um KG no qual se encontra a entidade *"astronautas"*, pois, nessa situação, essas entidades não possuem vínculo semântico;
– *Baixa qualidade de exemplos nos Golden Standards*: também conhecidos como *Padrão Ouro*, *Golden Standards* são conjuntos de dados que servem como um índice de referência (*benchmark*) para a realização de pesquisas em determinado tema. Isso permite comparar resultados de forma mais justa e objetiva. É importante destacar que alguns *Golden Standards* em QA apresentam inconsistências e erros pontuais. Ao analisar o conjunto de dados Question Answering over Linked Data (QALD) em 2019, encontramos erros de sintaxe como apresentado no exemplo a seguir:

```
1        SELECT (COUNT(DISTINCT ?uri) as ?c)
2        WHERE { <http://dbpedia.org/resource/Facebook>
3        <http://dbpedia.org/ontology/foundedBy> ?uri .
4        ?uri <http://dbpedia.org/ontology/foundedBy> ?
            uri . }
```

Essa limitação pode influenciar nos resultados dos testes realizados, principalmente no caso de uma avaliação qualitativa;
– *Pequena quantidade de conjunto de dados rotulados*: um dos maiores desafios de pesquisa dessa área é criar ou expandir os conjuntos de dados existentes (KACUPAJ *et al.*, 2021). Isso normalmente é realizado em cima de duas perspectivas. A primeira é o tamanho do conjunto de dados. Esse fator é bem relevante devido ao número crescente de sistemas de QA baseados em aprendizado de máquina profundo que necessitam de um volume maior de dados de treinamento (BORDES *et al.*, 2015). Outro aspecto crítico dos conjuntos de dados é a complexidade. Para Diefenbach *et al.* (2018), questões complexas são definidas como questões com necessidades de informação que só podem ser expressas usando agregação, comparação, superlativos, raciocínio temporal ou uma combinação deles. Portanto, as questões factuais precisam ser transformadas em questões complexas. Quando falamos em *Golden Standards*, esse problema cresce de tamanho. Por exemplo,

no QALD, ao longo de dez anos de evolução, há cerca de 2.275 exemplos, o que é pouco recomendável para o treinamento de modelos de aprendizagem profunda (LECUN *et al.*, 2015); e

– *Forte Dependência de Gramáticas*: algumas das principais abordagens de sistemas de QA são construídas a partir de *parsers* (BORDES *et al.*, 2014). A construção desses *parsers* demanda a criação de gramáticas que tentam prever a estrutura presente na pergunta de entrada. Essas gramáticas são desenvolvidas por humanos. Isso pode ser uma fator limitante porque, para cada padrão novo de pergunta, a gramática precisa ser revista. Consequentemente, os sistemas de QA têm uma forte dependência na criação de uma gramática adequada para que consigam atingir bons resultados. Vale a pena ressaltar que essa questão é específica para sistemas de QA que usam gramática.

Todos os desafios apresentados anteriormente ocorrem com diversas aplicações de QA mas um relacionado ao KG de extrema importância, é o pouco uso da semantica presente no KG. Os sistemas de QA recentemente têm focado em abordagens que levam apenas em consideração o uso de conjunto de dados para treinamento de modelos, não levando em consideração a informação presente nos KG. Assim, algumas inferências implícitas que poderiam ser feitas com base na própria estrutura semântica presente no KG não são realizadas (SILVA *et al.*, 2020; HÖFFNER *et al.*, 2017; DIEFENBACH *et al.*, 2018);

Entre as diversas alternativas de sistemas de QA, uma que vem se mostrando promissora para resolver esses desafios são os sistemas *Template-Based Question Answering* (TBQA) (UNGER *et al.*, 2012). Tais sistemas tentam classificar a questão fornecida como entrada em padrões de consultas (*query templates*). Esses padrões são conhecidos de antemão e são construídos em linguagens formais de consulta, a exemplo de SPARQL e Structured Query Language (SQL). Esses *templates* possuem campos especiais “em branco”, denominados *slots*, a serem preenchidos pelos parâmetros informados pelo usuário em sua pergunta. Desse modo, devem existir padrões em uma diversidade suficiente para que o sistema consiga capturar a estrutura arbitrariamente complexa necessária para responder uma questão qualquer (DIEFENBACH *et al.*, 2018). Na Figura 2, temos um exemplo do uso de *template* em sistema de QA.

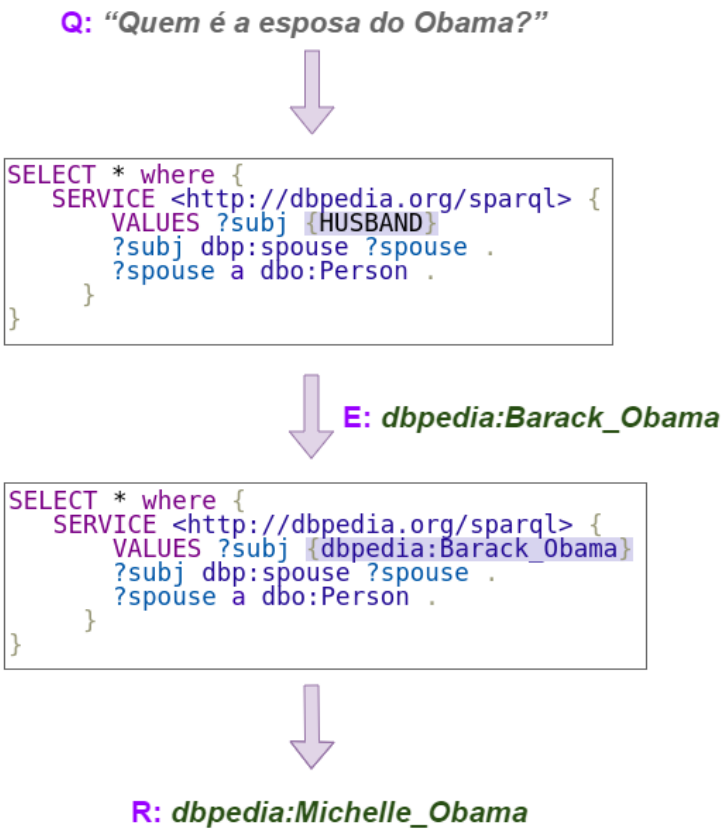
Os sistemas Template-Based Question Answering (TBQA) possuem algumas limitações bem conhecidas na literatura. Por exemplo, uma baixa diversidade nos *templates* gerados e a dificuldade de classificá-los para cada uma das inúmeras perguntas possíveis em linguagem natural. De acordo com Cimiano e Minock (2009), um dos maiores desafios para

qualquer interface de linguagem natural é lidar com a grande variabilidade na maneira como um determinado fato ou relação pode ser expresso. Aplicando esse conceito aos *templates*, uma qualidade-chave de um TBQA é a sua capacidade de gerar um bom número desses padrões, de modo que representem um possível conjunto de entradas em linguagem natural. Por fim, em relação à classificação, deve-se identificar precisamente qual dos *templates* suportados no sistema é o mais adequado para responder a uma determinada questão.

A Tabela 1 enumera alguns dos principais problemas encontrados na abordagem de TBQA.

Apesar de existirem diversos sistemas de QA (UNGER *et al.*, 2012; ZHENG *et al.*, 2015; CUI *et al.*, 2017; ABUJABAL *et al.*, 2017; SILVA *et al.*, 2020), ainda há poucos trabalhos de conjuntos de dados de associados a *templates* na literatura (TRIVEDI *et al.*, 2017; DUBEY *et al.*, 2019). Quando analisamos as bases de KBQA que utilizam paráfrases, esse número é ainda mais reduzido. Nessa última categoria, destacam-se os trabalhos LC-QuAD 2.0 (DUBEY *et al.*, 2019) e ComQA (ABUJABAL *et al.*, 2018), os quais serão detalhados mais adiante nesta tese.

Figura 2 – Exemplo do funcionamento de um sistema de QA que usa *template*.



Fonte: Elaborado pelo autor (2021).

Tabela 1 – Principais limitações identificadas na abordagem proposta por Zheng *et al.* (2018) segundo experimentos realizados usando nossa implementação das técnicas utilizadas pelos autores.

Limitação	Exemplo	Porcentagem de erros
Ligação entre entidade/ Baixa variabilidade dos *templates* gerados	"Give me all cosmonauts"	41%
Similaridade com *template*/ Dificuldade em classificar os *templates* gerados em relação à questão correspondente.	"Show me everyone who was born on Halloween"	33%
Ausência de *template*	"What is the height difference between Mount Everest and K2?"	18%

É importante ressaltar que a maioria dos conjuntos de dados para QA são criados de forma manual, recorrendo a especialistas (TRIVEDI *et al.*, 2017) ou usando serviços de *crowd-sourcing* (CHRISTMANN *et al.*, 2019). O principal problema dessas abordagens é o custo envolvido: seja financeiro, usando uma ferramenta de *crowd-sourcing*; ou intelectual, através de especialistas, pois o processo é bem cansativo.

Por outro lado, existem algumas técnicas que geram conjunto de dados de QA de forma automática (SERBAN *et al.*, 2016). O principal problema dessas técnicas é não criar grupos representativos de perguntas, pois, normalmente, as perguntas geradas têm uma baixa variabilidade por resultarem em muitas perguntas parecidas (DIEFENBACH *et al.*, 2018; HÖFFNER *et al.*, 2017).

Uma possível solução para essa baixa variabilidade é fazer uso de conhecimento de senso comum devido a riqueza semântica presente na sua estrutura. Para Liu e Singh (2004), o conhecimento de senso comum, assim definido, abrange uma grande parte da experiência humana e engloba o conhecimento sobre os aspectos espaciais, físicos, sociais, temporais e psicológicos da vida cotidiana típica. Como se presume que todas as pessoas possuem bom senso, esse conhecimento é normalmente omitido das comunicações sociais. Assim, o conhecimento de senso comum consegue introduzir novos conceitos e relações entre esses conceitos, enriquecendo os termos e reduzindo a baixa variabilidade semântica.

Segundo Liu e Singh (2004), o conhecimento de senso comum é o conhecimento sobre o mundo cotidiano das pessoas, ou seja, um conjunto de fatos básicos e entendimentos possuídos pela maioria das pessoas. Podemos citar alguns exemplos de conhecimento de senso

comum: *Um limão é azedo*; *Para abrir uma porta, você geralmente deve primeiro girar a maçaneta*; *Se você esquecer o aniversário de alguém, essa pessoa pode não ficar feliz com você*.

1.3 Solução Proposta e Contribuições Esperadas

Dada a breve explanação feita sobre o tema de pesquisa abordado nesta tese, é possível identificar duas importantes lacunas na literatura:

- A ausência de conjuntos de dados para sistemas de QA que usam *templates* para KBQA disponíveis publicamente para pesquisa; e
- A ausência de trabalhos de TBQA no contexto de KBQA.

Nesse contexto, a presente tese busca desenvolver um *framework* completo para sistemas de KBQA baseado que usam *templates*, porém, nosso diferencial é geração de *templates* com qualidade e uma alta variabilidade. Escolhemos usar a abordagem baseada em *templates* devido à sua boa aplicabilidade geral e à sua facilidade em adaptação para diversos contextos (AVILA, 2020), além de ser mais vantajosa para lidar com perguntas complexas (ZHENG *et al.*, 2018).

Uma alternativa para melhorar a baixa variabilidade é o uso de bases de conhecimento de senso comum para aumentar a diversidade das questões geradas a partir de um método de *paráfrase*. Porém, o uso de um método de paráfrase pode ter como consequência negativa um número grande de questões com pouca significância na construção de um *template*. Para resolver esse problema, recorremos a recursos externos, como utilizar uma ferramenta de extração de informação (*Open Information Extration*, Open IE). Open IE tem como principal objetivo extrair relações (arg_1, rel, arg_2) de uma sentença em linguagem natural, portanto é razoável assumir que o uso de Open IE deva prover melhores padrões na construção de *templates*.

1.3.1 Questões de Pesquisa

As seguintes questões de pesquisa foram formuladas a partir dos objetivo supracitados e dos desafios de pesquisa evidenciados nos estudos publicados em TBQA (SILVA *et al.*, 2020; ZHENG *et al.*, 2018):

QP1 O uso do conhecimento de senso comum pode aumentar a variabilidade das questões de um TBQA geradas com o uso de técnicas de paráfrase?

QP2 É possível extrair automaticamente, a partir de texto, relações úteis para responder questões

factuais usando a semântica de um KG?

QP3 Como enriquecer semanticamente um *corpus* de questões associadas aos *templates* de um TBQA?

QP4 Como representar e recuperar, de forma eficiente, os *templates* mais adequados para responder uma questão a partir de um grande volume de *templates*?

1.3.2 Hipóteses de Trabalho

As hipóteses de trabalho constituintes desta tese são derivadas diretamente das questões de pesquisa aludidas. Essas hipóteses estão elencadas a seguir:

H1 O uso de uma base de conhecimento de senso comum melhora a variabilidade das questões geradas;

H2 O uso da semântica presente noKGQA por meio do uso de consultas especialmente construídas para associar o *template* e a pergunta pode melhorar a qualidade nas respostas encontradas;

H3 A geração de um conjunto de dados de KBQA extraído de um *corpus* e enriquecido a partir de conhecimento de senso comum melhora a qualidade do sistema de QA e facilita evolução do método; e

H4 O uso de uma ferramenta de *Open IE* na extração de relações do *corpus* melhora a qualidade dos *templates* gerados.

1.3.3 Contribuições

A pesquisa realizada e as contribuições resultantes visam abordar o problema de pesquisa apresentado com base na investigação prática das hipóteses associadas. Para chegar à corroboração dessas hipóteses, pretendemos desenvolver a seguinte metodologia:

1. Investigação de pesquisas de ponta relevantes para o problema identificado. Isso inclui o estudo-da-arte sobre sistemas de QA e sistemas de QA baseados em *templates*, Grafos de Conhecimento, Banco de Dados e Web Semântica;
2. Formulação de questões e hipóteses de pesquisa;
3. Definição de soluções para abordar as hipóteses. Identificação de novas contribuições decorrentes das soluções, bem como o estudo das propriedades formais das soluções propostas; e
4. Implementação das soluções e condução de uma avaliação empírica de seu desempenho.

A avaliação de desempenho é realizada com relação às soluções de ponta, se disponíveis. As experiências serão conduzidas da seguinte forma:

a) Implementação de abordagens de última geração;
b) Conceber uma abordagem de avaliação de acordo com a questão de pesquisa estudada. Reutilizar e, se necessário, ajustar os critérios de referência e avaliação existentes;
c) Execução de experimentos com base nos *benchmarks* para obter dados a fim de tirar conclusões sobre as hipóteses de pesquisa; e
d) Análise dos resultados.

Ao responder essas questões de pesquisa sob a luz da metodologia descrita anteriormente, esta tese de doutoramento traz as seguintes contribuições:

1. A criação e disponibilização de um conjunto de dados de larga escala para QA, o chamado ExQuestions;
2. Desenvolvimento de um método de matching entre consulta em linguagem natural e *template*;
3. A avaliação do desempenho de modelos pré-treinados com dados para verificar a eficácia do algoritmo de paráfrase no contexto de QA;
4. Desenvolvimento de algoritmo de indexação baseado em ANN para retornar os *templates* de forma eficiente; e
5. Desenvolvimento de um sistema QA usando o Framework proposto.

1.4 Organização da Tese

O restante desta tese está organizado em sete capítulos, descritos brevemente a seguir:

– **Capítulo 1. Introdução**

Apresenta o problema de sistema de QA, o seu contexto social e desafios técnicos e científicos envolvidos. Além disso, apresenta uma visão geral da solução proposta pelo trabalho;

– **Capítulo 2. Fundamentação Teórica**

Nesse capítulo, são apresentados os conceitos fundamentais para a compreensão do trabalho, incluindo definições de Natural Language Interfaces (NLI), QA, as dimensões dos sistemas de QA, tipos de sistema de QA e como esses sistemas são avaliados;

– **Capítulo 3. Trabalhos Relacionados**

Apresentamos uma revisão bibliográfica comparativa de abordagens e resultados alcançados na literatura quanto à solução de problemas similares aos enfrentados no trabalho. Além disso, é desenvolvida uma análise comparativa dos trabalhos encontrados, bem como são traçadas algumas conclusões preliminares;

– **Capítulo 4. Um *Framework* baseado em Senso Comum para Sistemas de Perguntas e Respostas sobre Grafo de Conhecimento**

  Descreve o *framework* proposto, o qual é baseado em Senso Comum para Sistemas de Perguntas e Respostas sobre Grafo de Conhecimento. Inicialmente, o problema é definido de forma mais precisa face ao paradigma adotado na solução. Em seguida, o *framework* é apresentado de forma detalhada, incluindo os algoritmos e todas as técnicas desenvolvidas na sua criação;

– **Capítulo 5. Resultados**

  Nesse capítulo, são descritos os resultados encontrados. Abordamos principalmente o *ExQuestions*, um conjunto de dados KBQA obtido como resultado do *framework* proposto;

– **Capítulo 6. Experimentos**

  Capítulo onde descrevemos um conjunto de experimentos iniciais que validam nossa hipótese de pesquisa a partir dos resultados; e

– **Capítulo 7. Conclusões e Trabalhos Futuros**

  Concluímos a tese apontando as contribuições científicas, publicações realizadas durante o doutorado e descobertas a partir dos experimentos. Também são abordadas as limitações das abordagens experimentadas, possíveis ameaças à validade, oportunidades de melhorias e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir, vamos apresentar alguns conceitos usado na nossa estratégia e a definição dessas estratégias relacionadas a QA, classificando-as, e, por fim, discorrer sobre como avaliar os sistemas computacionais.

2.1 Definição do Problema

De uma maneira geral, construir um sistema que responda consultas em linguagem natural requer resolver duas tarefas subjacentes:

- Dados um *Grafo de Conhecimento*, G, e um *corpus* de texto, C, precisamos gerar *templates* de consultas que operem sobre G. O *corpus* de texto C é um conjunto de documentos que se relacionam com o *Grafo de Conhecimento*. Geralmente, C é fácil de obter devido ao grande volume de dados textuais disponíveis, como a Wikipédia. Em contrapartida, o Grafo de Conhecimento, nem sempre é disponível para G. Por isso, usamos a Wikipédia que se relaciona com a DBpedia, formando respectivamente C e G (BIZER *et al.*, 2009). Note-se que o objetivo dessa tarefa é gerar uma diversidade de *templates* que venha a ser capaz de responder virtualmente a todas as perguntas elaboradas sobre G; e
- Resolvido o problema anterior, dados um Grafo de Conhecimento, G, um conjunto de *templates* T e uma consulta de entrada q em linguagem natural, o objetivo é encontrar um *template* t que responda à consulta q.

2.2 Conceitos Centrais Usados na Tese

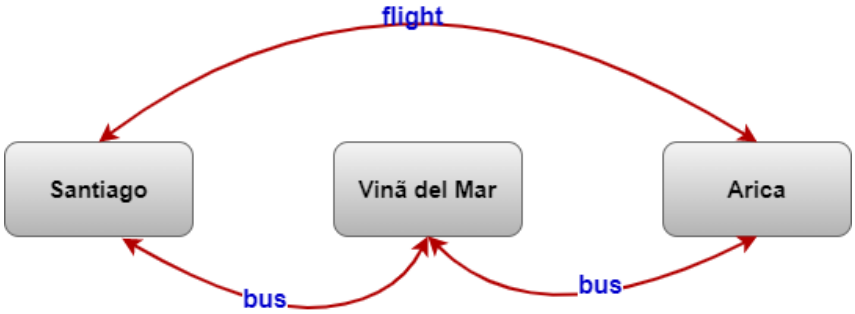
2.2.1 Knowledge Graph

De acordo com Pan *et al.* (2017), um *Knowledge Graph* pode ser definido como um conjunto de dados estruturados que é compatível com o modelo de dados Resource Description Framework (RDF) e pode possuir uma ontologia como seu esquema. Uma das vantagens dos KGs é que, como representam informações *formalmente* em termos de uma ontologia, os sistemas podem recorrer a raciocinadores automáticos para deduzir novos conhecimentos a partir de tais KGs.

Zheng *et al.* (2018) definem formalmente um KG como grafo direcionado $G = (V, E, L)$, tal que V é um conjunto de vértices que representam entidades, tipos e literais, enquanto

E indica as arestas que representam predicados com seus rótulos e L demonina os predicados e as propriedades. Na Figura 3, temos um exemplo de KG, onde o conjunto de V é representado por três elementos(*Arica*, *Santiago* e *Vinã del Mar*). O conjunto E podemos citar como exemplo: *(Arica, flight, Santiago)* e por fim, temos três predicados(*flight* e *bus*).

Figura 3 – Exemplo de Grafo de Conhecimento



Fonte: Adaptado de Hogan *et al.* (2021).

Cada aresta E é chamada de tripla $< v_1, r, v_2 >$, onde v_1 e $v_2 \in V$, e r é um predicado ou propriedade. O conhecimento em um KG pode ser explorado por meio de consultas. Por definição, as consultas são sentenças expressas para recuperar informações. De acordo com (BORDES *et al.*, 2015), as consultas podem ser classificadas em duas categorias: simples (factuais) e complexas (ou não factuais).

2.2.2 Consultas Simples (factuais) e Complexas (não factuais)

De acordo com (BORDES *et al.*, 2015), as consultas factuais são perguntas de domínio aberto simples. A resposta de uma consulta simples pode ser obtida apenas conhecendo uma entidade e uma propriedade, pois como uma propriedade é um atributo que é perguntado sobre uma entidade específica (e.g., o local, o ano ou as pessoas em uma questão). A resposta para tal questão é novamente uma entidade ou um conjunto de entidades presente no KG. Por exemplo, na pergunta *"Quem é a esposa de Barack Obama?"*, *"Barack Obama"* é uma entidade e *é esposa de* é uma propriedade.

Por sua vez, as consultas complexas são questões nas quais a resposta não está acessível de forma direta no documento ou base de conhecimento (BUCHHOLZ; DAELEMANS, 2001). Assim, é necessário realizar algum processamento extra ou inferência para responder a uma questão complexa. Por exemplo, na consulta, *"Quem é a mãe da esposa do Barack Obama?"*. A relação *"mãe da esposa"* não é uma relação direta, necessita de uma inferência em cima da base de conhecimento.

2.2.3 Templates de Consulta

Para definir *templates* de consultas, recorreremos à mesma definição adotada pelo trabalho de Zheng *et al.* (2018): um *template* t contém dois padrões $t.n$ e $t.s$, escritos em linguagem natural e em SPARQL, respectivamente. Nesses padrões, as entidades correspondentes em $t.n$ e $t.s$ são substituídas por seus tipos. Esse processo caracteriza os parâmetros de entrada indicados por *slots*, de modo a favorecer a identificação dos parâmetros de $t.s$ a partir de $t.n$. Um *template binário* refere-se ao *modelo* que contém apenas um fato da representação ontológica. Por exemplo, na Figura 2 temos a demostração de um *template* SPARQL de *Quem é a esposa do Obama*. Isso só foi possível com a extração dos *slots* $(Person?, husband, < Person >)$, onde o primeiro *Person?* se refere-se a variável e os segundo <*Person*> a entidade extraída da pergunta em linguagem natural.

O problema de responder a uma questão q usando um conjunto de *templates* pode ser resolvido por meio dos seguintes passos: (1) recuperar o *template* t adequado para q, possivelmente comparando q com $t.n$; (2) definir os parametros usados em t a partir de q; (3) executar $t.s$ sobre o KG; e (4) converter o resultado para linguagem natural.

2.2.4 Paráfrase

Encontrar respostas nos sistemas de QA é um desafio devido às muitas maneiras diferentes como a linguagem natural expressa a mesma necessidade de informação. Uma simples mudança na questão pode fazer o sistema de QA produzir respostas diferentes, ocasionando em problemas na obtenção da resposta adequada. Por exemplo, considerando as questões "*Quem foi Barack Obama?*" e "*Quem foi o primeiro presidente afrodescendente dos EUA?*", ambas possuem o mesmo sentido que deve ser recuperado em um KGQA.

McKeown (1983) propôs o uso de paráfrase para realizar consultas em uma interface de linguagem natural. Segundo o autor, a principal vantagem dessa abordagem é que, ao ver a paráfrase, o usuário tem a opção de reformular sua pergunta antes que o sistema tente executá-la para obter uma resposta. Assim, se a pergunta não foi interpretada corretamente, o erro pode ser detectado antes que uma busca possivelmente longa na base de dados seja iniciada. No trabalho de (MCKEOWN, 1983), as paráfrases são geradas a partir de regras definidas de forma manual.

No decorrer dos anos, com o advento e evolução dos algoritmos de Aprendizado de Máquina e subsequente introdução dos modelos Seq2Seq usando aprendizado de máquina

profundo, podemos destacar os trabalhos de (PRAKASH *et al.*, 2016) e (HASAN *et al.*, 2016). Prakash *et al.* (2016) usaram uma rede com arquitetura LSTM empilhada sobre uma Seq2Seq para ampliar a capacidade do modelo na geração de paráfrases. Por sua vez, o trabalho de Hasan *et al.* (2016), incorpora o mecanismo de atenção para gerar paráfrases.

Em modelos mais modernos, como as arquiteturas *Transformers*[1], podemos citar o trabalho de (EGONMWAN; CHALI, 2019), que integra um modelo *Transformer* de dependência de longo alcance na sequência de entrada.

2.2.5 Conhecimento de Senso Comum

O conhecimento de senso comum consiste em fatos e conhecimentos espaciais, físicos, sociais, temporais e psicológicos, possuídos pela maioria das pessoas, os quais são frutos da experiência da vida diária (ANACLETO *et al.*, 2007). Por exemplo, quando alguém fala "*Eu comprei doces*", está implícito que: o sujeito usou dinheiro; que o efeito de cair de uma moto é você se machucar; e que objetos rolam de superfícies inclinadas. Essas duas últimas informações podem ser omitidas da comunicação, pois subtende-se que os interlocutores as possuem de antemão.

Esse conhecimento de senso comum pode ser representado em bases semânticas que expressam relações semânticas mais interessantes entre os conceitos. Por exemplo, relações que expressam características funcionais, causais, afetivas e relativas a eventos, as quais podem servir como base para responder a questões complexas. Podemos citar como exemplos de bases de conhecimento de senso comum: WordNet (MILLER, 1995), FrameNet (BAKER *et al.*, 1998), ConceptNet (SPEER *et al.*, 2017) e InferenceNet (PINHEIRO *et al.*, 2010).

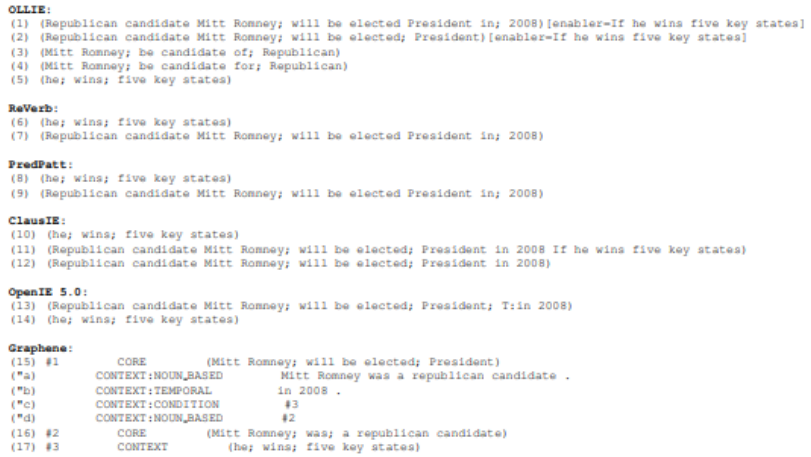
2.3 Open Information Extration

Em meados de 2007, o Banko *et al.* (2007), introduz o conceito de *Open Information Extraction - Open IE*, um paradigma de extração de informação que extraí relações independente do domínio. Para (LI *et al.*, 2011), os sistemas *Open IE* utilizam padrões gerais para extrair todas as possíveis relações entre entidades. De acordo com Jurafsky (2000), a extração de informação transforma a informação não estruturada expressa em texto em linguagem natural em um representação estruturada na forma de tuplas relacionais que consistem em um conjunto de argumentos e uma frase denotando uma relação semântica entre eles: $< arg_1; rel; arg_2 >$. Na

[1] https://github.com/huggingface/transformers

Figura 4, temos a comparação da saída gerada por diferentes sistemas Open IE para a frase de entrada: "*If he wins five key states, Republican candidate Mitt Romney will be elected President in 2008*".

Figura 4 – Exemplo de execução de um Open IE



Fonte: Extraído de Niklaus *et al.* (2018)

2.4 *Natural Language Interfaces*

As décadas de 1970 e 1980 experimentaram um grande desenvolvimento na área de Linguística Computacional, o que culminou na introdução de iniciativas voltadas à construção dos Sistemas de QA (HERZOG; ROLLINGER, 1991; WILENSKY *et al.*, 1988). Além disso, a possibilidade de aplicação das Interfaces em Linguagem Natural (NLI, *Natural Language Interfaces*) para acesso às bases de dados atraiu o interesse da comunidade de pesquisadores em Processamento de Linguagem Natural (NLP, *Natural Language Processing*). Um exemplo de tal interface de alto nível é a *Unix Consultant* (UC) (WILENSKY *et al.*, 1988). Essa interface foi desenvolvida por Robert Wilensky na U.C. Berkeley no final dos anos 1980 para mediar o aprendizado de usuários leigos sobre o sistema operacional *Unix 2* (WILENSKY *et al.*, 1988).

O uso de NLI vem se tornando cada vez mais popular, apresentando-se como uma maneira de interação homem-máquina mais simples, natural e intuitiva para usuários em geral. Existem diversos tipos de NLI com diferentes propósitos, técnicas de interação, grau de expressividade e formalidade. Cada tipo de NLI possui seu uso recomendado junto de suas vantagens e desvantagens, sendo a escolha do tipo correto de NLI de vital importância para o sucesso do sistema final.

A busca pelo desenvolvimento de interfaces de consulta sobre estruturas de dados via

linguagem natural tem como objetivo permitir o retorno de respostas para perguntas ou consultas dadas como entrada pelo usuário. Como um exemplo do uso de NLI como interface de consultas, ou Interfaces de Consulta em Linguagem Natural (NLQI, *Natural Language Query Interface*), temos sistemas de QA que utilizam técnicas de recuperação de informação e processamento de linguagem natural para responder automaticamente a uma pergunta feita em linguagem natural pelo usuário. O foco do nosso trabalho será em sistemas de QA.

2.5 Introdução aos Sistemas de QA

Os sistemas de QA definem um campo de pesquisa razoavelmente antigo da Ciência da Computação se considerarmos documentos de texto e a recuperação de informação como aspectos cruciais de sistemas de consulta (ABUALIGAH, 2019). De acordo com Diefenbach *et al.* (2018), os primeiros sistemas de QA foram desenvolvidos para mediar o acesso às informações encontradas no banco de dados nos anos próximos à virada da década de 1960 para 1970.

Segundo Höffner *et al.* (2017), as abordagens iniciais de QA eram basicamente constituídas por etapas, sendo a primeira responsável por analisar a *pergunta* em linguagem natural para identificar palavras-chaves e a segunda por realizar a *recuperação da informação*. Note-se que, com a evolução e especificação de novas técnicas especialmente voltadas para esse problema, outras etapas foram se consolidando nos sistemas de QA.

2.6 Dimensões dos Sistemas de QA

Lopez *et al.* (2011) propõem uma classificação de sistemas de QA com quatro dimensões:

- A entrada ou tipo de pergunta que pode ser aceita pelo sistema de QA (e.g., fatos, diálogos, documentos, etc);
- Como o sistema de QA lida com os problemas intrínsecos tradicionais que a busca impõe em qualquer sistema de QA não trivial (e.g., adaptabilidade, vocabulário e ambiguidade);
- As fontes das quais podem derivar as respostas (i.e., dados estruturados *versus* dados não-estruturados); e
- O escopo do sistema (i.e., específico do domínio *versus* independente do domínio).

A Tabela 2 apresenta alguns exemplos dessas dimensões:

Tabela 2 – As dimensões do QA proposto por Lopez *et al.* (2011)

Tipo de Entrada	Problemas Encontrados	Fontes de Dados	Escopo
Palavras Chaves Factoides (Wh-, Sim ou Não) Entendimento e raciocínio de causalidade (Por que, Como,..) Raciocínio espacial e temporal Raciocínio sobre conhecimento de senso comum Diálogos Interativos	Escalabilidade Heterogeneidade Vários idiomas Confiabilidade	Estruturados Semi estruturados Texto (Web) Semântico	Domínio Dependente (domínio fechado) Domínio Independente (domínio aberto) Proprietário (bases de dados privadas)

2.7 Tipos de sistemas de QA

Nesta seção, vamos descrever as principais classificações de QA encontradas na literatura.

2.7.1 Classificação Baseada no Tipo de Resposta

A classificação de sistemas QA baseada no tipo de resposta, conforme preconizado pela proposta de Latifi (2018), é mais generalista, por restringir o espaço da busca onde se espera que a resposta seja encontrada. Essa linha de pesquisa adota a seguinte classificação das abordagens utilizadas nos sistemas de QA (LATIFI, 2018): Closed-Domain Question Answering (QADR); QA for Comprehension Reading (QACR); Community Question Answering (QAC); QA Over Domain Ontologies, Ontology-Based QA (QASOBO); e Linked Open Data Question Answering (QALOD).

Na abordagem QADR, as questões e o espaço de pesquisa são restritos a um determinado domínio, como, por exemplo, geografia, turismo, economia, entre outros - ressalta-se que o domínio médico tem atraído um interesse particularmente significativo para diversas aplicações. Os sistemas QADR são geralmente aplicados a tarefas específicas e usam léxicos, terminologias, bases de conhecimento, ontologias e outros recursos léxico-conceituais restritos a domínios (LATIFI, 2018).

Como os espaços de pesquisa são menores, as QADR geralmente têm se demonstrado virtualmente inúteis, a exemplo das que recorrem a uma "*votação*" considerando vários sistemas especialistas. É importante destacar que, dada a especificidade dos domínios, os usuários costumam demonstrar grande expectativa por respostas adequadas nesse contexto. Além disso, trata-se de um cenário no qual é preferível não apresentar resposta do que reportar respostas

erradas.

Consequentemente, o desempenho do sistema geralmente é avaliado sob a ótica da sensibilidade (*recall*) em detrimento da precisão. Também é importante salientar que o processamento das questões e dos documentos da base de conhecimento representa um desafio no contexto de QA. Isso ocorre porque esses materiais frequentemente contêm siglas, conteúdo não textual como tabelas e listas detalhadas, jargões específicos de domínio, entre outros elementos desafiadores (FERRÁNDEZ *et al.*, 2009).

Já na abordagem QACR, as questões estão relacionadas com um documento para verificar a capacidade do usuário de ter entendido o conteúdo do documento. Richardson *et al.* (2013) propõem o MCTest[2], um conjunto de questões destinado à pesquisa sobre a compreensão de texto por máquina. As questões são formuladas a respeito de 660 histórias, de tal modo que cada uma delas exige que o leitor as entenda sob diferentes aspectos.

Também denominada redes sociais de QA, a abordagem QAC lida com um cenário baseado em interações dentro de uma comunidade. No caso, um de seus membros formula uma consulta inicial usando linguagem natural, desencadeando uma linha de intervenções dos membros da comunidade. Isso ocorre de forma análoga às interações observadas nas redes sociais virtuais, nas quais os usuários respondem, refinam e comentam as publicações anteriores.

Já no caso de QASOBO, as respostas são buscadas não em documentos de texto, mas em ontologias. Isso permite aproveitar não apenas os dados linguísticos e terminológicos incluídos na ontologia, mas também suas relações, propriedades e capacidades de inferência.

Por fim, as abordagens QALOD são contextualizadas no âmbito da Web Semântica, a qual tem ganhado maior evidência recentemente devido ao significativo crescimento observado na disponibilização de recursos de domínio abertos e fechados. Nota-se que tanto a quantidade quanto a complexidade desses recursos têm crescido. Ademais, vários desses recursos estão incluídos na iniciativa *Linked Open Data*.

2.7.2 Classificação Baseada na Tarefa

Diefenbach *et al.* (2018) propuseram uma classificação dos sistemas de QA na qual um sistema pode ser dividido em cinco tarefas correspondentes às etapas no processamento da pergunta: (1) análise da pergunta; (2) mapeamento da sentença; (3) desambiguação; (4) construção da consulta; e (5) consulta em bases de conhecimento distribuídas. Cada uma dessas

[2] http://research.microsoft.com/mct

etapas é detalhada a seguir:

- **Análise da Pergunta**. A questão do usuário é analisada com base em recursos (analisadores sintáticos usados a partir de expressões regulares, entre outros), ou seja, em estratégias puramente sintáticas. Os sistemas de QA usam tais recursos para deduzir, por exemplo, a segmentação correta da pergunta em partes menores, como sintagmas nominais (*chunking*), ou mesmo em partes ainda menores, chamadas de *tokens*, como palavras e pontuações. Existem também os *dependence parser*, usados para determinar a instância correspondente em uma determinada sentença, ou seja, classificar uma palavra como sujeito, objeto ou propriedade; além disso, eles indicam se há dependência, e onde ela se encontra, entre os termos de uma determinada sentença.
- **Mapeamento da Sentença**. Essa etapa tem como entrada uma sentença *S* formada por uma ou mais palavras e tenta encontrar, no(s) KG(s) subjacente(s), um conjunto de recursos que correspondam a *S* com alta probabilidade. Note-se que *S* pode corresponder a uma instância, propriedade ou classe do(s) KG(s) consultados;
- **Desambiguação**. Dois problemas de ambiguidade podem surgir ao responder uma pergunta. O primeiro problema surge quando, a partir da etapa de análise da pergunta, a segmentação e as dependências entre as sentenças são ambíguas. Por exemplo, na pergunta "Dê-me todos os países europeus", a segmentação pode ou não agrupar a expressão "países europeus", levando a duas possibilidades. O segundo problema pode ocorrer na etapa de mapeamento da sentença, quando se retorna vários recursos possíveis para uma frase. No exemplo acima, "europeu" pode ser mapeado para diferentes significados da palavra "*Europa*";
- **Construção da Consulta**. Essa fase lida com a forma com a qual o sistema de QA efetivamente constrói a consulta SPARQL que será usada para encontrar a resposta para a pergunta sobre o(s) KG(s) subjacente(s). Note-se que, tipicamente, é necessário fornecer os parâmetros cabíveis para que essa consulta formulada automaticamente seja capaz de produzir uma resposta adequada à pergunta fornecida como entrada; e
- **Consulta em Bases de Conhecimento Distribuídas**. A etapa final é responsável por efetivamente consultar o(s) KG(s) subjacente(s) a fim de obter respostas para a pergunta apresentada pelo usuário. Note-se que a resposta pode ser obtida a partir de um único KG ou, dependendo do sistema e da tarefa, a partir de vários KGs. Quando a consulta opera sobre vários KG, o sistema de QA deve abstrair soluções para determinados problemas

que advêm dessa abordagem: informações conflitantes; variação semântica; agregar informações complementares em diferentes KG.

De acordo com Hannun *et al.* (2014), soluções ‘‘*fim-a-fim*” são uma alternativa “*simples*” porém poderosa para lidar com a tarefa de QA. A ideia principal dessa abordagem consiste em treinar modelos de Aprendizagem de Máquina, geralmente usando o paradigma de aprendizado profundo (LECUN *et al.*, 2015), com o objetivo de substituir a maioria dos módulos que lidam com as etapas de QA por um único modelo “*inteligente*”.

Dentre todas essas classificações apresentadas, nosso trabalho está posicionado como QALOD, já que o KG está contido na Linked Open Data (LOD). É se caracteristica como uma abordagem hibrida pois nos usamos *templates* e técnicas de aprendizagem de maquina na nossa solução.

2.8 Avaliações de Sistemas de QA

Em nosso levantamento bibliográfico, pôde-se constatar que existe uma quantidade razoável de conjuntos de dados os quais são úteis para a avaliação dos sistemas de QA, ou seja, coleções anotadas de perguntas com as respostas conhecidas e outros metadados. Esses conjuntos de dados são conhecidos na literatura como *Golden Standards* ou *Benckmarks*, sendo elementos fundamentais nas avaliações dos resultados em estudos investigativos. Mais especificamente, esses conjuntos de dados têm como papel viabilizar um forte controle de qualidade para evitar problemas na validação dos sistemas de QA.

Normalmente, *Golden Standards* contêm um grande número de perguntas, portanto o controle de resultados ocorre com a realização de verificações manuais em pequenos cenários de teste, essencialmente de caráter qualitativo, visando, assim, assegurar a exatidão[3] dos dados e dos resultados obtidos.

Dos *Golden Standards* encontrados na literatura, há quatro que são mais populares para a tarefa de QA: QUALD (*Question Answering over Linked Data*) (CIMIANO *et al.*, 2013; CIMIANO; MINOCK, 2009); WebQuestions (BERANT *et al.*, 2013); SQuAD (*Stanford Question Answering Dataset*) (RAJPURKAR *et al.*, 2016); e SimpleQuestion (BORDES *et al.*, 2015). A Tabela 3 foi construída para estabelecer um quadro comparativo entre cada um desses conjuntos de dados; já a Tabela 4 apresenta exemplos de questões contidas em cada conjunto de

[3] De acordo com Vetulani *et al.* (2011), podemos dizer que um algoritmo está correto quando ele está correto em relação à sua especificação.

dados.

As perguntas do QALD[4] são preparadas anualmente pelos organizadores do desafio. Elas, em geral, podem ser respondidas usando até três relações binárias e frequentemente precisam de agregadores como **ORDER BY** e **COUNT**. Além disso, algumas das questões estão fora do escopo. O conjunto de treinamento é bastante pequeno, com cerca de 50 a 250 perguntas, o que resulta num *dataset* pequeno para aprendizado supervisionado.

O WebQuestions, por sua vez, contém cerca de 5.810 perguntas criadas a partir do KG *Freebase*. Aproximadamente 97% desses exemplos podem ser respondidos usando apenas uma única instrução retificadora com potencialmente algumas poucas restrições, que, em geral, são temporais ou de tipo, ou seja, compostas por pergunta simples.

O SQuAD [5] é um conjunto de dados para compreensão da leitura e consiste em perguntas coletadas usando o Amazon Mechanical Turk (AMT) em um artigo definido na Wikipédia. A resposta para cada pergunta é um segmento de texto ou sua extensão de uma passagem de texto; caso contrário, a pergunta poderá ser propositalmente respondida enquanto estiver sendo escrita para parecer semelhante a perguntas respondíveis (RAJPURKAR *et al.*, 2016). Cerca de cem mil perguntas foram extraídas de questões validadas usando o projeto DBPedia[6], o que, por sua vez, assegura uma relação direta com a ontologia.

SimpleQuestions é um conjunto de dados de perguntas factuais criado a partir do KG *Freebase*. Devido à sua natureza factual, este *Golden Standard* contém cerca de 108.442 perguntas que podem ser respondidas usando uma relação binária. Petrochuk e Zettlemoyer (PETROCHUK; ZETTLEMOYER, 2018) avaliaram os resultados das perguntas simples e concluíram que cerca de 33,9% das perguntas não podem ser respondidas devido aos problemas subjacentes à ambiguidade; portanto, o desempenho é vinculado a 83,4%, pois muitas perguntas têm mais de uma interpretação plausível.

Tabela 3 – *Golden Standards* para avaliação de QA

Golden Standard	**Número de exemplos**	**Tipo de Pergunta**
QALD	250	Factoid & Non-Factoid
WebQuestions	5.810	Factoid
SQuAD	100.000	Factoid & Non-Factoid
SimpleQuestions	108.442	Factoid

[4] https://project-hobbit.eu/challenges/qald-8-challenge/
[5] https://rajpurkar.github.io/SQuAD-explorer/
[6] https://www.dbpedia.org/

Tabela 4 – Exemplos de questões para cada *Golden Standard*. O problema de QA está longe de ser simples. SQuAD propositalmente contém questões sem resposta, como o exemplo em itálico, para fins de controle de qualidade.

Golden Standard	**Exemplos**
QALD	*List all the musicals with music by Elton John.*
	Which films did Stanley Kubrick direct?
	How many companies were founded in the same year as Google?
WebQuestions	*Where was Barack Obama born?*
	Who are famous people from Spain?
	What year Lebron James came to the NBA?
SQuAD	*The extinction of what led to the decline of rainforests?*
	How is the physical image placed on the blank CD?
	Which endemic species has symbolic significance to Mexicans?
SimpleQuestions	*Which known professional football player was born in Dublin?*
	Which Brazilian composer was born in Rio de Janeiro?
	What is the name of a line that stops in the Seoul Station?

2.9 Conclusões Preliminares

Ao longo deste capítulo é possível constatar que o tema de pesquisa abordado na presente tese é complexo, envolvendo conceitos de diversas áreas. Aspectos importantes porém periféricos a esta tese, tais como o desbalanceamento nos avanços em diferentes idiomas, foram omitidos em favor de uma discussão pautada nos métodos e nos desafios de pesquisa.

Apesar de haver bastante esforço nessa linha de pesquisa, foram evidenciados de forma mais precisa os desafios em aberto (mencionados anteriormente no Capítulo 1) e as metodologias de avaliação adotadas tradicionalmente na literatura. É possível perceber uma clara tendência na utilização de técnicas de aprendizado profundo para lidar com as sutilezas subjacentes aos desafios que permeiam os sistemas de QA. Assim como será mostrado no nesse trabalho. Em particular, é preciso ressaltar a demanda por conjuntos de dados que contenham *templates* de perguntas e respostas que sejam confiáveis para lidar com o problema de KBQA.

3 ESTADO DA ARTE

Neste capítulo, apresentamos as principais técnicas e algoritmos usados em sistemas de QA, ressaltando as principais vantagens e desvantagens de cada abordagem. Por fim, realizamos uma comparação entre as abordagens. .

3.1 Abordagens que usam QA com Ontologias

Nos sistemas de QA baseados em ontologias (*QA Over Domain Ontologies, Ontology-Based QA* - QASOBO), as respostas não são buscadas diretamente em documentos de texto simples e não estruturados, mas em ontologias (DIEFENBACH *et al.*, 2018). Tal paradigma permite tirar vantagem não apenas de dados linguísticos e terminológicos incluídos na ontologia, mas também de suas relações, propriedades e capacidades de inferência.

Todavia, o acesso a dados estruturados em Knowledge Base (KB)s tais como DBPedia (BIZER *et al.*, 2009), YAGO (SUCHANEK *et al.*, 2007) e Freebase (BOLLACKER *et al.*, 2008) se dá por meio de consultas escritas em linguagens como SPARQL (SEABORNE; PRUD'HOMMEAUX, 2008)[1]. Essas linguagens de consulta se mostram bastante poderosas, mas requerem conhecimentos prévios dos usuários sobre uma sintaxe complexa e sobre as bases de conhecimento envolvidas nas consultas (DIEFENBACH *et al.*, 2018). Outra alternativa possível é recorrer à construção de sofisticadas interfaces com os usuários, as quais são repletas de formulários. Infelizmente, essa abordagem demanda significativos investimentos em automação e na personalização para se adequar a cada domínio específico de aplicação.

A ideia por trás de um sistema de QA de base de conhecimento é recuperar as informações solicitadas pelo usuário em linguagem natural usando uma consulta operando sobre KB. Portanto, o problema crítico ao projetar sistemas de QA que operam em KBs e LOD pode ser definido como traduzir a necessidade de informações do usuário em um formato avaliável, usando para tanto técnicas de processamento de consulta de banco de dados padrão ou inferências automáticas. Por isso, o uso de sistemas de QA que usam ontologias na construção da consulta se mostra uma alternativa que se mostra bastante viável (LATIFI, 2018; DIEFENBACH *et al.*, 2018).

De acordo com Diefenbach *et al.* (2018), o processo de construção de uma resposta para uma pergunta em um sistema de QA pode ser dividido em cinco tarefas: (1) análise de

[1] http://www.w3.org/TR/sparql11-query/

perguntas; (2) mapeamento de frases; (3) desambiguação; (4) construção de consultas; e (5) execução de consultas em bases de conhecimento distribuídas.

Pondo em foco abordagens que usam a ontologia como estratégia, cada etapa do processo proposto por Diefenbach *et al.* (2018) será delineada, aqui, da seguinte forma:

- **Análise da Questão:** Nessa fase, a questão do usuário é analisada com base em recursos puramente sintáticos. Os sistemas de QA, por exemplo, usam recursos sintáticos para deduzir qual seria a segmentação correta da questão. Isso significa definir (a) qual elemento da sentença corresponde a uma instância (sujeito ou objeto), propriedade ou classe e (b) determinar a dependência entre as diferentes partes da frase. Na Análise da Questão, também são usadas técnicas para reconhecimento de entidades nomeadas (*Named Entity Recognition*, NER), *Part-Of-Speech* (POS) e identificação de dependências entre os elementos das sentenças;
- **Mapeamento para Ontologia:** O principal objetivo dessa etapa é realizar uma correspondência entre os elementos da questão e os elementos de uma ontologia. Essa correspondência pode ser feita ao nível de instância, propriedade ou classe;
- **Ambiguidade:** Um problema bem comum na maioria dos sistemas de QA é a ambiguidade, i.e., o fenômeno de uma mesma frase possuir diferentes significados. A ambiguidade pode ser estrutural e sintática (em *"Exigiu o dinheiro do marido"*, por exemplo, o dinheiro é do marido ou apenas estava com ele?) ou lexical e semântico (a palavra "*banco*" pode se referir a um banco de praça ou a um banco como instituição financeira) (HÖFFNER *et al.*, 2017). Como mostrado por Petrochuk e Zettlemoyer (2018), o tratamento da ambiguidade propicia uma melhora significativa nos sistemas de QA em termos das métricas de avaliação;
- **Construção da Consulta:** Determinada a correspondência, a próxima etapa é construir uma consulta, geralmente em SPARQL. Infelizmente, pode ocorrer um problema chamado *lacuna semântica*, que é compreender e delinear a intenção do usuário em termos da sintaxe usada na construção da consulta. A linguagem de consulta apresenta uma sintaxe essencialmente técnica que é inadequada para os usuários finais, pois construir consultas sofisticadas requer conhecimentos prévios sobre a sintaxe da linguagem e os dados na qual a consulta será realizada, compreensão que o usuário típico não possui; e
- **Consulta em Bases de Conhecimentos Distribuídas:** Até agora, as técnicas foram discutidas pensando em responder questões a partir de uma única KB. Assumindo múltiplas

fontes de conhecimento, a questão pertinente seria a seguinte: O que mudaria no caso de considerar a consulta sobre várias KB? Apenas alguns sistemas de QA abordaram esse problema, que, segundo Höffner *et al.* (2017), pode ser classificado em dois grupos. O primeiro grupo pressupõe que os KBs são grafos disjuntos, enquanto o segundo pressupõe que os KBs estão interligados, ou seja, os recursos que se referem à mesma entidade são identificados através dos KBs por meio de *links owl:sameAs*.

3.1.1 Metodologia para Comparação de Abordagens QA baseadas em Ontologias

De acordo com Latifi (2018), os dois principais desafios encontrados pelos sistemas de QA baseados em ontologias são a (1) interpretação das questões e (2) a aquisição de conhecimento. A interpretação ou entendimento de questões reflete a estratégia adotada por um sistema de QA para mapear questões em consultas sobre a base de conhecimento. Assim, a interpretação da questão envolve inferir a necessidade de informação dessa questão e, em seguida, formular um plano para obter uma resposta. O segundo desafio relaciona-se com a seguinte problemática: *Como um sistema de QA adquire e representa o conhecimento necessário para responder às questões*?

As pesquisas analisadas demonstram que, com esforço manual suficiente, é possível projetar um sistema de QA capaz de responder a questões elaboradas sobre um determinado tópico, a exemplo das questões de domínio específico. No entanto, até onde foi possível examinar a literatura, não existe um sistema único que possa responder a questões complexas em vários domínios. Isso é sumarizado pela Tabela 3.1.1, que enumera os principais sistemas de QA encontrados em nossa investigação e são analisados a seguir.

ID	Sistemas de QA	Características	Limitações	Base Utilizada	Avaliação
T001	ONLI (MITHUN *et al.*, 2007)	Consultas em LN com nRQL. Análise Sintática. Mapeamento de Ontologia. Interface de consulta para o sistema Racer.	Dependente de Domínio. Tipos limitados de perguntas analisadas.	Próprios (30 perguntas e respostas pares de consultas nRQL para duas ontologias relacionadas ao genoma)	Mean Reciprocal Rank (MRR): 0,35
T002	PANTO (WANG *et al.*, 2007)	Usa WordNet e medidas de similaridade de string em algoritmos de mapeamento de ontologia. Questões em LN como entrada e Questões em SPARQL como saída. QueryTriples como representação intermediária. Converte consultas em triplas em OntoTriples.	Escalabilidade: só funciona para pequenas ontologias. Não usa técnicas de indexação de dados. Limita o escopo da consulta. Fraca interação do usuário.	Mooney (http://www.cs.utexas.edu/users/ml/nldata.html) Base geográfica, restaurante e emprego.	Acurácia: 88,05%, 90,87%, 86,12% Acurácia: 85,86%, 96,64%, 89,17%
T003	AquaLog (LOPEZ *et al.*, 2007)	Gramática de domínio independente. Consulta em LN. Usa algoritmos de similaridade de string. Usa GATE e WordNet. Similaridade de relacionamento baseada em ontologia.	Falta de serviços de raciocínio adequados definidos pela ontologia. Não entende as consultas no formato "*How much*". Não explora quantificadores de escopo: ("cada", "todos", e "algum")	Base própria construída com 69 pares de perguntas e respostas. Domínio Fechado.	Acurácia: 58%
T004	QuestIO (TABLAN *et al.*, 2008)	Domínio Aberto. Traduz LN e palavras-chave para gerar a consulta em SPARQL. Pesquisa ontológica de dicionário. Transformação Iterativa até um consulta SeRQL seja obtida.	Falta interação do usuário. Baseda na Sessão. Não é possível resolver a ambiguidade de termos de palavras-chave.	36 questões de domínio específico.	Acurácia: 50%
T005	QACID (FERRÁNDEZ *et al.*, 2009)	Testado para espanhol no domínio do cinema. Conjunto de consulta criado por meio de *clusters*. Mapeamento de consulta entre LN e bases de conhecimento usando métricas de distância,	Caro devido à dependência do domínio. Só pode ser aplicado com cobertura limitada. Vazamentos de capacidades conscientes de contexto temporal e espacial.	Domínio Fechado. 162 pares de questões.	Acurácia: 89%
T006	FREyA (DAMLJANOVIC *et al.*, 2010)	Domínio Aberto. Identificação e verificação de conceitos de ontologia. Geração de consultas SPARQL. Identificação do tipo de pergunta. Aprendizagem por reforço para melhorar a classificação das sugestões. Sessão baseado na interação.	Requer testes com grandes conjuntos de dados. A avaliação não é centrada no usuário.	250 questões do *Mooney Geoquery*	Acurácia: 92,4% MRR: 78%
T007	QASYO (HOGAN *et al.*, 2011)	Consultas em LN. YAGO como entrada. Consultas em LN é traduzido em um conjunto de representações intermediárias, query *traps*. Traduz-se em triplas compatíveis com ontologia.	Falta informação sobre a natureza e complexidade de possivelmente mudanças necessárias na ontologia e na componente linguística.	-	Acurácia: 84,7%
T008	Pythia (UNGER; CIMIANO, 2011)	Lida com uma ampla gama de questões linguisticamente complexas envolvendo quantificadores, numerais, comparações, superlativos e negação. Mapeia corretamente os termos LN nos conceitos de ontologia correspondentes, apesar de serem superficialmente diferentes. O léxico específico do domínio é construído automaticamente a partir de uma especificação de realizações linguísticas do conceito de ontologia.	Portabilidade (requer um novo modelo LexInfo para um novo domínio a ser construído); Requer esforço não desprezível para domínios maiores (DBpedia, por exemplo)	880 qyestões do Mooney Geoquery	Acurácia: 82%
T009	DEQA (LEHMANN *et al.*, 2012)	Aplicativo para a Web de Dados. Usa o algoritmo TBSL. Sistema Abrangente Web QA. Extrações da Web usando OXPath. Usa LIMES para calcular links complexos para especificação.	Precisa abranger mais tipos de perguntas. Não suporta operadores complexos. Não oferece suporte a vários idiomas.	100 questões de domínio específicos	Acurácia: 57%
T010	QAAL (KALAIVANI; DURAISWAMY, 2012)	Usa *matching* de conceitos do grafo. Adota SPARQL. Usa PLN para analise de QA. Algoritmo de ativação difusa.	Modelo normal de pesquisa por palavra-chave. Não consegue responder a perguntas complexas em casos ambíguos. Domínio Fechado.		*Mean Accuracy Distribution*
T011	QAKiS (CABRIO *et al.*, 2012)	Domínio Aberto. QA sobre base de conhecimento. Informações relevantes de forma não estruturada.	Não é possível lidar com questões booleanas e com n relacionamento. Não pode executar consultas procedural, temporal ou espacial.	QALD-2 (DBPedia).	Acurácia: 39%
T012	PARALEX (FADER *et al.*, 2013)	Domínio Aberto. Transforma de texto em tuplas. Aprendizagem orientada para questões de interpretação de parágrafos. Nenhum modelo manual deve ser criado.	Incapaz de trabalhar com questões complexas. Falta de "*responsabilidade*".	WebQuestions (2,032). TREC (517). WikiAnswers (7310).	
T013	SINA (SHEKARPOUR *et al.*, 2015)	Domínio Aberto. Tenta reduzir a lacuna lexical. Usa Cadeia de Markov. *Baseado em Templates*.	Não é possível realizar consultas procedimentais, temporais ou espaciais. Não suporta operações complexas. Não oferece suporte a vários idiomas.	QALD-3.	Acurácia: 32% MRR: 0,8
T014	DEANNA (YAHYA *et al.*, 2012; YAHYA *et al.*, 2013)	Abordagem de Programação Inteira Linear (ILP). Extensão de consulta para SPO e SPOX. Relaxamento da consulta quando nenhum resultado for encontrado.	Não é possível realizar consultas procedimentais, temporais ou espaciais. Não suporta operações complexas. Não suporta vários idiomas. Não usa *templates*.	QALD-1. NAGA.	Acurácia: 55%
T015	ScoQAS (LATIFI *et al.*, 2017; LATIFI, 2018)	Abordagem Hibrida. Domínio Aberto e Fechado. Inferências em Grafos	Não suporta operações complexas;	QALD-2. QALD-3. QALD-4.	Acurácia: 60%

Utilizamos dois critérios para fazer a busca desses trabalhos:

1. A busca inicial foi realizada em torno de trabalhos de QA que utilizam a ontologia no processo de construção da estratégia para solução; e
2. Foram considerados tanto sistemas de domínio aberto de QA quanto sistemas de domínio fechado.

Por fim, pretendemos discutir os principais desafios e apontar trabalhos futuros em sistemas de QA que usam ontologias.

3.1.2 Comparação de Abordagens QA baseadas em Ontologias

***Ontology Natural Language Interaction*, ONLI$^+$** (MITHUN *et al.*, 2007): trata-se de um sistema de QA com Natural Language Processing (NLP) usado como *front-end* para o raciocinador Racer (HAARSLEV; MÖLLER, 2001) e linguagem nRQL, *new Racer Query Language* (HAARSLEV *et al.*, 2004). A nRQL aumenta e estende a API funcional do sistema Racer para consultar uma base de conhecimento usando tuplas no formato TBox e ABox, que correspondem, respectivamente sobre conceitos e indivíduos, às assertivas provenientes do arcabouço matemático da Lógica Descritiva (BAADER *et al.*, 2003). Assim, o sistema ONLI$^+$ é capaz de fornecer uma função de consulta para recuperar todos os indivíduos mencionados em uma caixa de seleção que são instâncias de um determinado conceito de consulta, por exemplo. O

método foi avaliado através da realização de diferentes experimentos usando a *Mean Reciprocal Rank (MRR)*. Os resultados experimentais exibidos pelos autores mostram que esse sistema foi incorporado ao ONLI$^+$ sem perder desempenho em termos de transformar consultas de linguagem natural em consultas nRQL, mas, definitivamente, aumenta a expressividade do usuário.

***Portable nAtural laNguage inTerface to Ontologies*, PANTO** (WANG *et al.*, 2007): esse sistema modela uma interface de linguagem natural portátil para ontologias. O sistema PANTO aceita entradas de Linguagem Natural (LN) em formulários e gera consultas SPARQL. É baseado na premissa de fazer um mapeamento entre a ontologia (conceitos, instâncias e relações) e a linguagem natural usando uma árvore de análise sintática que é construída a partir do analisador sintático de Standford (MARNEFFE; MANNING, 2008). Além da árvore sintática, o sistema utiliza de forma combinada a WordNet (MILLER, 1995) e os algoritmos de similaridade de *strings* (COHEN *et al.*, 2003) para aumentar a qualidade do mapeamento. O resultado desse algoritmo é convertido para a forma de tripla da consulta no OntoTriples (abreviação de *Ontology Triple*, uma tripla que, na forma de <*subject, predicate, object*>, é compatível com algumas declarações na ontologia e nela são representadas como entidades) (WANG *et al.*, 2007). As consultas geradas pelo OntoTriples são finalmente interpretadas na forma de SPARQL. O principal problema do PANTO é a forte dependência do Analisador Sintático de Stanford.

AquaLog (LOPEZ *et al.*, 2007): um sistema QA portátil que recebe como entrada consultas expressas em LN e uma ontologia, retornando respostas extraídas de uma ou mais KBs. Segundo Lopez *et al.* (2007), com o tempo de uso, o sistema é capaz de aprender jargões do usuário em vias de melhorar sua experiência. No AquaLog, dois modelos são usados como o componente linguístico: o primeiro é usado para converter as questões em linguagem natural em formato de tripla de consulta; o segundo é o Serviço de Similaridade de Relação, que transforma tripla de consulta para tripla no formato da ontologia. O modelo de dados consiste em triplas RDF. Um dos principais problemas encontrados em AquaLog é o seu desempenho questionável em relação à sua capacidade de resolver questões complexas.

***Question Answering System Applied to the Cinema Domain*, QACID** (FERRÁNDEZ *et al.*, 2009): uma abordagem aplicada e testada na língua espanhola usando uma modelagem ontológica para o domínio do Cinema. O sistema QACID conta com cinco componentes principais: a ontologia, os dados, um dicionário, as coleções de consultas de usuários e o mecanismo de vinculação. O QACID busca preencher a lacuna entre a expressividade dos usuários e

a representação formal do conhecimento com base em coleções de consultas de usuários. Uma forte limitação dessa abordagem é a grande dependência do domínio utilizado — nesse acaso, especifico que o sistema foi construído para o domínio de cinema. Uma consequência dessa limitação é um cobertura restrita a resolver questões envolvendo tempo e espaço.

***Question-based Interface to Ontologies*, QuestIO** (TABLAN *et al.*, 2008): utiliza uma *Natural Language Interface (NLI)* para acessar informações estruturadas independentes do domínio. Segundo Tablan *et al.* (2008), essas informações são fáceis de usar por dispensarem uma etapa de treinamento. QuestIO busca oferecer a simplicidade da interface de pesquisa do portal Google para a recuperação conceitual. Para tanto, esse sistema converte automaticamente as consultas conceituais curtas em consultas formais que podem ser executadas sobre qualquer repositório semântico. Podemos citar como principais limitações: a falta de interação do usuário (a abordagem não usa o retorno fornecido pelo usuário); a base em sessão (não guarda o contexto da pergunta para melhorar os resultados); e, por fim, não ter tratamento algum para resolver a ambiguidade presente na busca a partir de palavras-chaves.

***Feedback, Refinement and Extended Vocabulary Aggregation*, FREyA** (DAML-JANOVIC *et al.*, 2010): Sucessor do sistema QuestIO, FREyA proporciona melhorias no que diz respeito a uma compreensão mais profunda do significado semântico das questões. Essa abordagem visa lidar melhor com ambiguidade quando as ontologias estão abrangendo diversos domínios. FREyA permite ao usuário inserir consultas de qualquer forma e recorre a uma árvore de análise sintática para identificar o tipo de resposta das questões em vias de apresentar respostas concisas. O principal problema encontrado nesse trabalho é como foi feita a sua avaliação. Primeiro, a metodologia da avaliação proposta requer um grande conjunto de dados. Além disso, essa avaliação não é centrada no usuário apesar do que preconiza a proposta desse sistema de QA.

***Question Answering System for YAGO Ontology*, QASYO** (HOGAN *et al.*, 2011): sistema de QA ao nível de sentença que integra PLN, ontologias e tecnologias de Recuperação de Informação (RI) (MANNING *et al.*, 2010) em uma mesma abordagem. QASYO aceita como entradas consultas expressas em linguagem natural e na ontologia YAGO (SUCHANEK *et al.*, 2007; SUCHANEK *et al.*, 2008) e fornece respostas extraídas a partir das marcações semânticas disponíveis. A análise semântica de questões é realizada para extrair palavras-chaves usadas nas consultas de recuperação e para detectar o tipo de resposta esperada. O sistema QASYO está dividido em quatro etapas: (1) classificação da questão; (2) componente linguístico; (3) gerador

de consultas; e (4) processador de consulta. Podemos citar como desafios dessa abordagem a forte dependência do YAGO, o que dificulta uma generalização da solução, além de tratar consultas baseadas em palavras-chaves, o que limita bastante a expressividade das consultas e trata apenas de consultas mais simples.

***Ontology-based Question Answering on the Semantic Web*, Pythia** (UNGER; CIMIANO, 2011): esse sistema constrói representações de significado composicionalmente usando um alinhamento de vocabulário para uma ontologia. Ao fazê-lo, Pythia se baseia em uma análise linguística profunda, a qual permite a construção de consultas formais, mesmo para questões complexas de linguagem natural. Por exemplo, Pythia consegue elaborar consultas envolvendo quantificação e superlativos. O Pythia parte de duas premissas para o seu funcionamento: a primeira é usar representações linguísticas para construir composicionalmente representações gerais de significado que podem ser subsequentemente traduzidas em consultas formais — para realizar isso, ele utiliza uma gramática; a segunda premissa é o uso de uma interface para realizar uma especificação léxico-ontológica que explica possíveis ligações linguísticas de conceitos de ontologia. Podemos citar como a principal limitação dessa abordagem a baixa portabilidade, ou seja, a dificuldade de generalizar uma solução e a restrição de trabalhar com bases pequenas.

***Deep Web Extraction for Question Answering*, DEQA** (LEHMANN *et al.*, 2012): arcabouço conceitual que combina tecnologias semânticas com extração efetiva de dados. Para tanto, DEQA mapeia questões expressas em linguagem natural em padrões SPARQL e aborda o problema como uma combinação de três áreas de pesquisa: (1) extração de dados da *web* para obter ofertas de sites de imóveis, onde não existe uma interface estruturada para os dados, o caso de todas as agências imobiliárias de Oxford; (2) integração de dados para vincular os extraídos com conhecimento prévio, tais como informações geoespaciais sobre pontos de interesse relevantes; e (3) QA para fornecer ao usuário uma NLI capaz de compreender até mesmo consultas complexas. A abordagem apresenta as seguintes limitações principais: um número reduzido de padrões de perguntas, o que tem como consequência uma cobertura limitada, não abrangendo perguntas complexas; a ausência de suporte a operadores complexos como perguntas do tipo “menor que” e “maior que”, entre outras.

QAAL (KALAIVANI; DURAISWAMY, 2012): resultado de pesquisas sobre vários tipos de sistemas de QA baseados em ontologias e modelos da Web Semântica com diferentes formatos de consulta. Kalaivani e Duraiswamy (KALAIVANI; DURAISWAMY, 2012) comparam diferentes tipos de entrada, métodos de processamento de consulta e os formatos de entrada

e saída de cada sistema, além de analisarem e discutirem diferentes métricas de desempenho e suas respectivas limitações. Os autores propuseram o uso de um algoritmo para correspondência de grafos no intuito de combinar a consulta (COLLINS; LOFTUS, 1975). Podemos citar como principal vantagem dessa abordagem a técnica baseada em *matching* de grafos adotando o SPARQL como padrão para saída da consulta, além de usar técnicas de PLN para melhorar a precisão. O principal problema dessa abordagem está na busca por palavras-chaves, que possui escopo limitado e, consequentemente, não cobre as questões complexas. Ademais, o sistema não trata ambiguidades e é restrito a um domínio fechado.

***Question Answering wiKiframework-based System*, QAKiS** (CABRIO *et al.*, 2012): sistema de QA para domínio aberto sobre *Linked Data*. O sistema QAKiS aborda o problema da interpretação de questões como uma correspondência com a ontologia baseada em relações, na qual os fragmentos da questão correspondem às relações binárias da ontologia. Sua principal contribuição é desenvolver um mapeamento baseado em relações para interpretação de questões no intuito de converter a questão do usuário em uma linguagem de consulta (por exemplo, SPARQL). Ele tenta primeiro estabelecer uma correspondência entre fragmentos da questão e padrões textuais (*templates*) coletados automaticamente da Wikipédia. Um problema do sistema é uma forte dependência da Wikipédia e DBpedia.

PARALEX: segundo Fader *et al.* (2013), para fornecer uma interface declarativa natural, uma das propriedades desejadas de um sistema de QA é a robustez sobre variações nas questões em linguagem natural. Anthony Fader e seus colaboradores propuseram um sistema de controle de qualidade de domínio aberto interativo que mapeia questões em consultas simples sobre extrações feitas por um sistema de extração de informação aberto, a saber, *Open Information Extraction (IE)* (BANKO *et al.*, 2007). Portanto, PARALEX é executado sobre uma base de conhecimento extraída juntamente com frases do portal WikiAnswers [2] para aprender uma função de consulta e, por fim, realizar consultas sobre uma KB. A principal limitação encontrada nessa abordagem é a incapacidade de trabalhar com questões complexas.

SINA (SHEKARPOUR *et al.*, 2015): sistema escalável de pesquisa proposto por Shekarpour *et al.* (2015) para responder às consultas dos usuários. SINA transforma as palavras-chaves fornecidas pelo usuário ou as consultas expressas usando LN em consultas SPARQL sobre um conjunto de fontes de dados interligadas. O sistema SINA recorre a um Modelo de Cadeia de Markov para determinar, a partir de diferentes conjuntos de dados, quais os recursos

[2] https://www.answers.com/

mais adequados para responder à consulta apresentada pelo usuário. A principal vantagem dessa abordagem é que ela é independente do esquema de dados da ontologia, pois gera um conjunto de *templates* predefinidos que é dimensionado para grandes bases de conhecimento de fácil uso. Tem como principais desvantagens: a não realização de consultas procedimentais, temporais ou espaciais, não realiza consultas completas e nem é multilíngue.

DEANNA (YAHYA *et al.*, 2012): proposto inicialmente por Yahya *et al.* em 2012, o sistema apresentou melhorias no ano seguinte (YAHYA *et al.*, 2013). Até onde foi possível pesquisar, é o único sistema que aborda o QALD a partir de uma perspectiva de Programação Linear Inteira (PLI). Os autores recorreram a esse arcabouço de otimização no intuito de resolver simultaneamente os problemas de decomposição e desambiguação das questões. O modelo proposto por Yahya *et al.* combina a seleção de frases e seu mapeamento em alvos semânticos. Os autores introduziram restrições que garantem que as frases sejam selecionadas de forma a preservar suas dependências frasais na imagem do mapeamento para os destinos semânticos. Essas restrições tem como principal consequência negativa a não realização de consultas procedimentais, temporais ou espaciais. Além de outras limitações, como, por exemplo: o não suporte a outros idiomas.

***A Semantic-based Closed and Open Domain Question Answering System*, ScoQAS** (LATIFI *et al.*, 2017; LATIFI, 2018): esse sistema propõe uma abordagem hibrida, lidando com questões factuais para os domínios aberto e fechado. ScoQAS utiliza o QALD para avaliar a estratégia. A abordagem tem como principal diferencial usar um conjunto de inferências em grafo para o domínio fechado. O principal problema dessa abordagem é não tratar questões complexas.

3.2 Abordagens que usam QA e Aprendizado de Máquina Profundo

Como mencionado anteriormente, segundo Diefenbach *et al.* (2018), o processo de construção de um sistema de QA pode ser realizado em várias etapas. Entretanto, existem sistemas de QA que têm apenas uma etapa. Tal abordagem é conhecida na literatura como “fim a fim” ou “ponta a ponta”. De acordo com Hannun *et al.* (2014), os modelos de aprendizado de máquina profundo (*Deep Learning*), apresentam uma solução alternativa simples, porém poderosa. A proposta dessa estratégia é treinar de ponta a ponta, ou seja, usar o aprendizado profundo para substituir a maioria dos módulos do modelo conceitual de Diefenbach *et al.* (2018) por um único modelo “inteligente”.

Deep Learning (LECUN *et al.*, 2015) rapidamente se transformou em uma tendência tecnológica à medida que os algoritmos de otimização subjacentes começaram a apresentar resultados sólidos em uma série de aplicações, ultrapassando até mesmo o nível humano em muitas tarefas complexas, tais como reconhecimento facial, tradução automática e reconhecimento de fala (HE *et al.*, 2015; RAO *et al.*, 2017).

Todavia, treinar um modelo de aprendizado profundo para uma tarefa de QA requer uma quantidade considerável de dados, geralmente acompanhados de anotações que guiam o processo de otimização. Esse problema é conhecido como "*data hunger*". Dentre os *corpora* usados para treinar e avaliar modelos de QA, vamos considerar sistemas de QA treinados ou avaliados sobre o *corpus* SimpleQuestions (BORDES *et al.*, 2015), um *factoid question answering (FQA)* construído sobre o KB Freebase (BOLLACKER *et al.*, 2008). Essa decisão foi tomada porque SimpleQuestions é um dos *corpora* mais recorrentes adotados por diversos trabalhos da literatura. De fato, questões factuais são de grande interesse para pesquisadores (BORDES *et al.*, 2014; YAO; DURME, 2014; BAST; HAUSSMANN, 2015).

Os sistemas de QA baseados em aprendizado profundo de fim a fim mostram na literatura resultados proeminentes, visto a quantidade considerável de amostras de dados que geralmente está disponível no campo de QA (CIMIANO *et al.*, 2013; BERANT *et al.*, 2013; BORDES *et al.*, 2015; RAJPURKAR *et al.*, 2016). Os primeiros resultados mostraram melhorias significativas em relação às abordagens então consideradas como estado-da-arte (BORDES *et al.*, 2015; JOULIN *et al.*, 2017). Esses métodos de QA são adequados para o público em geral devido à ampla disponibilidade de tecnologias de suporte, como bibliotecas e estruturas de aprendizagem profunda. Ademais, as métricas de avaliação são incorporadas naturalmente em um processo experimental que está se tornando menos demorado e cada vez mais favorável à disponibilização de formas de se reproduzir os resultados publicados (LUKOVNIKOV *et al.*, 2017; LUKOVNIKOV *et al.*, 2019). Por outro lado, esses métodos consomem muitos recursos (e.g., processamento, memória, comunicação e energia elétrica). Além disso, geralmente demandam *hardware* especializado para acelerar os cálculos, pelo menos durante o estágio de treinamento que normalmente é realizado de forma *off-line*. Por fim, esse campo de pesquisa experimenta um desenvolvimento notavelmente acelerado, o que pode prejudicar uma percepção clara das contribuições pela comunidade científica.

3.2.1 Metodologia para Comparação de Abordagens que usam QA e Aprendizado de Máquina Profundo

Foram realizadas buscas sobre trabalhos de QA que usam técnicas de aprendizado de máquina profundo sobre conjunto de dados factuais com o objetivo de responder às seguintes questões de pesquisa:

RQ1 Qual é o estado da arte de abordagens de consulta em linguagem natural que usam técnicas fim a fim?

RQ2 Quais são as principais metodologias utilizadas para avaliar cada trabalho?

RQ3 Quais são os principais desafios de pesquisa em cada uma das abordagens?

Para fazer a busca desses trabalhos, utilizamos dois critérios, que vamos apresentar e explicar detalhadamente em seguida:

1. Inicialmente, a busca foi feita em cima de trabalhos de QA que usam técnicas de aprendizado de máquina profundo; e
2. Foram considerados os trabalhos que usam a base de questões factuais SimpleQuestions como *Golden Standard*.

3.2.2 Trabalhos que usam QA e Aprendizado de Máquina Profundo

O trabalho de Bordes *et al.* (2015) é um dos primeiros publicados na literatura de acordo com os critérios de busca supracitados. Os autores apresentam duas grandes contribuições. A primeira, um conjunto de dados de perguntas e respostas em grande escala baseado no dataset SimpleQuestions. A segunda contribuição consiste em apresentar um sistema de QA desenvolvido sob a estrutura das Redes de Memória (MemNNs). A precisão do modelo chega a 63,9%.

Diversos trabalhos utilizando dados factuais sobre a SimpleQuestions foram publicados em 2016. O primeiro deles que analisaremos é de Jain (2016), que introduz a *Factual Memory Network*, uma rede neural que "aprende a responder". Em linhas gerais, é uma nova arquitetura baseada em redes de memória proposta por Bordes *et al.* (2015), a qual pode ser treinada de ponta a ponta usando pares de perguntas e respostas no conjunto de treinamento. São duas as principais contribuições do trabalho de Jain (2016). A primeira é a criação de redes de memória factuais, que são usadas para responder perguntas em linguagem natural (por exemplo, "*Onde nasceu Bill Gates?*") usando fatos armazenados na forma de triplas (sujeito, predicado,

objeto) em um KB (por exemplo, *"Bill Gates", "local de nascimento", "Seattle")*, reduzindo a supervisão do método (já que armazena a informação por mais tempo), diferentemente do trabalho do Bordes *et al.* (2015), que utilizou uma forte reforço na forma de dar suporte aos fatos de pergunta durante o treinamento para, assim, conseguir melhorar seu desempenho. A segunda contribuição é aumentar a eficiência do modelo em termos de várias medidas de desempenho, além de fornecer uma melhor cobertura de fatos relevantes, selecionando inteligentemente quais nos deve expandir. Assim, essa abordagem chega a 55,6% de precisão.

No trabalho publicado por Yin *et al.* (2016), foram apresentados dois conceitos bastante difundidos em diversos trabalhos de QA. O primeiro conceito é propor ligar a entidade presente na pergunta a uma entidade presente na Freebase. Essa ideia simples melhorou substancialmente o estado da arte dos trabalhos uma vez que diversas técnicas passaram a utilizá-la. A segunda mudança proposta é utilizar o conceito de atenção (*attentive*) em uma CNN. A ideia principal é usar uma nova técnica de *maxpooling* com *attentive* empilhada sobre a palavra-CNN, de modo que, de uma forma mais efetiva, a representação do predicado pode ser combinada com a representação da questão, mas com foco no predicado. Nesse trabalho, foi observada uma precisão de 67,2%.

Dai *et al.* (2016) apresentam um *Framework* Condicional Probabilístico (CFO) utilizando uma CNN combinada com um modelo probabilístico que serve de inspiração para diversos trabalhos. As contribuições do artigo são o uso de um tratamento totalmente probabilístico com uma nova parametrização condicional usando redes neurais. Isso é seguido pelo método de poda focado para reduzir o espaço de busca durante a inferência e, por fim, o uso de duas variações do método visando melhorar a generalização de representações altamente esparsas para milhões de entidades sob supervisão. Essa abordagem chega a uma precisão de 75,7%.

A partir das indicações de Dai *et al.* (2016), Aghaebrahimian e Jurčíček (2016) propõem um sistema de QA escalável e um modelo de reconhecimento de entidade usando a pesquisa em grafo de conhecimento. O trabalho foi feito a partir da extração de metadados disponíveis no grafo de conhecimento e a integração destes no sistema usando a estrutura de Modelo Condicional Restrito (CCM) para desambiguar entidades. Com essa abordagem, a precisão chega a 65,2%.

Outro método foi proposto por Golub e He (2016), que tem como ideia principal criar uma rede neural que codifica as questões e a base de conhecimento ao nível de caractere, ou seja, *letra por letra*. Na construção da rede neural, é utilizada uma *Long Short-Term Memory (LSTM)*

(HOCHREITER; SCHMIDHUBER, 1997), para codificar a questão. A principal vantagem dessa abordagem é que a modelagem mostrou-se apta a generalizar bem novas palavras não vistas durante a etapa de treinamento. A precisão desse método chega a 70,9%.

Os resultados de Golub e He (2016) inspiraram o trabalho de Lukovnikov *et al.* (2017), publicado no início de 2017. A ideia principal é realizar a codificação em nível de caractere e em nível de palavra para modelar uma *Gated Recurrent Unit (GRU)*. A modelagem em nível de caractere se mostrou interessante para o manuseio de palavras fora do vocabulário, enquanto a modelagem em nível de palavra confere uma maior capacidade semântica ao modelo. Unir essas duas abordagens se mostra eficiente para gerar representações independentes de base de conhecimento de entidades e relações que são construídas somente a partir de informações textuais associadas às entidades ou relações. A precisão do método fica em torno de 71%. Ressalte-se como contribuição desse trabalho a condução de uma análise qualitativa sobre 250 perguntas, destacando os principais problemas da modelagem.

Zhu *et al.* (2017) observaram que certas partes de uma questão geralmente se sobrepõem aos nomes de seu assunto e à relação correspondente na base de conhecimento. Os autores propuseram uma rede neural com arquitetura sequência a sequência (Seq2Seq) usando uma Long Short-Term Memory (LSTM) que codifica um par de candidatos sujeito-relações e as decodifica para a questão dada. Assim, a probabilidade estimada de decodificação é usada como critério para selecionar a melhor resposta. Com essa técnica, o sistema chega a ter uma acurácia de 72,4%.

Na contribuição de Ture e Jojic (2017), encontramos a maior acurácia para o conjunto de dados do SimpleQuestions até o momento da escrita desta tese, em torno de 88,3%. A estratégia utilizada pelos autores é treinar uma Rede Neural Recorrente (*Recurrent Neural Network - RNN*) para resolver cada subproblema, requisito essencial para sistemas de QA em tempo real. Porém, muitas pesquisas não consideram essa abordagem na literatura, pois os autores desse trabalho não disponibilizaram nem o código nem o modelo, impossibilitando a reprodução dos experimentos e dos resultados.

Mohammed *et al.* (2017) mostram que LSTM e Gated Recurrent Unit (GRU) básicas, além de algumas heurísticas, aproximam-se do estado da arte. Além disso, as técnicas que não usam redes neurais também têm um desempenho razoavelmente bom, chegando no patamar de 74,9% de acurácia. Mohammed *et al.* (2017) chegam à conclusão de que esses resultados sugerem que, enquanto as redes neurais de fato contribuem para avanços significativos nessa

tarefa, alguns modelos exibem complexidade desnecessária e os melhores deles geram ganhos modestos.

Em 2018, destaca-se o trabalho de Petrochuk e Zettlemoyer (2018), que mostram que a ambiguidade dos dados limita o desempenho dos sistemas pois, muitas vezes, há perguntas que têm mais de uma interpretação igualmente plausível. Partindo dessa hipótese, são apresentadas novas evidências de que SimpleQuestions pode ser quase resolvido por métodos já conhecidos, chegando ao limite de desempenho em 83,4%. Os autores também introduzem um novo conjunto de dados derivado do SimpleQuestions, atingindo uma acurácia de 78,1% usando métodos já conhecidos como LSTM.

Buzaaba e Amagasa (2019) propõem uma abordagem simples que funciona razoavelmente bem em comparação com as abordagens complexas encontradas anteriormente na literatura. Esses autores apresentam um novo índice que depende do tipo de relação para filtrar entidades de assunto de uma lista de candidatos, de modo que a entidade de objeto com a pontuação mais alta seja escolhida como a resposta à pergunta. Esse trabalho apresenta duas contribuições principais: (1) em contraste com suas contrapartes complexas (BORDES *et al.*, 2015; GOLUB; HE, 2016; LUKOVNIKOV *et al.*, 2017), os autores adotam uma abordagem mais rápida para treinar LSTM e GRU que aplicam rede neural de ponta a ponta em uma tarefa semelhante de simples resposta de perguntas; e (2) a própria estratégia de indexação baseada em tipos de relação. Os autores obtiveram uma precisão de 74,6%.

A proposta de Lukovnikov *et al.* (2019) é de uma nova abordagem para sistemas de QA sobre questões simples. Lukovnikov e seus colaboradores investigaram a aprendizagem por transferência para responder a perguntas sobre KGQA com base em modelos para modelagem de linguagem. O BERT foi o modelo de escolha para o ajuste fino, então os pesos existentes foram transferidos usando SimpleQuestions como a tarefa-alvo, resultando em uma precisão de 77,3%.

A Tabela 5 apresenta uma classificação dos sistemas QA existentes com base no nível de codificação de texto (caractere, palavra ou hierárquico), adoção de representações vetoriais (*embeddings*) existentes (por uso direto ou ajuste fino) e se o sistema usa qualquer tipo de memória no modelo (LSTM, GRU, etc.). Esses resultados sugerem que a geração de uma boa representação via *embeddings* é um componente crucial para o desempenho de última geração. Em particular, BERT (DEVLIN *et al.*, 2018) e arquiteturas semelhantes parecem ser uma adição muito promissora ao campo de QA.

Tabela 5 – Artigos classificados com base no nível de codificação, reutilização de incorporação existente e existência de módulos de memória no modelo. O nível de codificação hierárquico combina caracteres e palavras. Melhores resultados em SimpleQuestions (TURE; JOJIC, 2017; PETROCHUK; ZETTLEMOYER, 2018; LUKOVNIKOV *et al.*, 2019) recorrem consistentemente a embeddings de palavras existentes (Word2Vec, GloVe + *fastText*, e BERT, respectivamente).

Artigo	**Nível de Codificação**	**Reuso *Embedding*?**	**Usa Memória?**
(GOLUB; HE, 2016)	character	Não	Sim
(DAI *et al.*, 2016)	word	Não	Não
(BORDES *et al.*, 2015)	word	Não	Sim
(JAIN, 2016)	word	Não	Sim
(ZHU *et al.*, 2017)	word	Não	Sim
(BUZAABA; AMAGASA, 2019)	word	Não	Sim
(AGHAEBRAHIMIAN; JURČÍČEK, 2016)	word	Sim	Não
(TURE; JOJIC, 2017)*	word	Sim	Não
(MOHAMMED *et al.*, 2017)	word	Sim	Sim
(PETROCHUK; ZETTLEMOYER, 2018)*	word	Sim	Sim
(YIN *et al.*, 2016)	character and word	Não	Não
(LUKOVNIKOV *et al.*, 2017)	hierarchical	Sim	Sim
(LUKOVNIKOV *et al.*, 2019)*	hierarchical	Sim	Sim

A Tabela 6 resume uma comparação de todas as técnicas mostradas nessa subseção em termos de informações de desempenho disponíveis, se houver.

3.2.3 Desafios

Mesmo afirmando que as pesquisas sobre as abordagens de fim a fim para sistemas de QA podem estar atingindo o limite de seu potencial, nossos esforços investigativos foram capazes de identificar alguns desafios de pesquisa importantes nesse campo, como observado por nós no trabalho do Petrochuk e Zettlemoyer (2018). As questões apresentadas por nós são algumas das mais desafiadoras que encontramos com base em nossa revisão sistemática da literatura e em trabalhos recentes sobre o problema (DIEFENBACH *et al.*, 2018; HÖFFNER *et al.*, 2017; SILVA *et al.*, 2020; LIANG *et al.*, 2021). A Tabela 7 resume os desafios atuais encontrados nos sistemas de QA e as respectivas abordagens mais proeminentes para esforços de pesquisa futuros.

Tabela 6 – Comparação de desempenho nos trabalhos publicados.

Método	**Execução**	**Performance**	**Resultado**
Redes de Memória (BORDES *et al.*, 2015)	-	-	-
Redes de Memória Factuais (JAIN, 2016)	Intel Core i 5 CPU with 8GB RAM	Tempo de treinamento Consulta Média Tempo de Resposta	740 min 200 ms
CNN Atenta (YIN *et al.*, 2016)	-	-	-
Foco Condicional Redes de Memória(CFO) (DAI *et al.*, 2016)	-	-	-
Modelos Condicionais Restritos (AGHAEBRAHIMIAN; JURČÍČEK, 2016)	-	questões por segundos (detecção de entidades)	20
Character-Level Attentive LSTM (GOLUB; HE, 2016)	GPU	Tempo de Treinamento	3 dias
Character-Level and Word-Level Gated Recurrent Unit (LUKOVNIKOV *et al.*, 2017)	Titan X GPU	Tempo de Treinamento	6 dias
Seq2Seq (ZHU *et al.*, 2017)	-	-	-
Simple Recurrent Neural Networks (TURE; JOJIC, 2017)	CPU	Latência	76 ± 16 ms
LSTM + GRU (MOHAMMED *et al.*, 2017)	GeForce GTX 1080 GPU	-	-
Linear-Chain Conditional Random Field Tagger (CRF) + BiLSTM (PETROCHUK; ZETTLEMOYER, 2018)	-	-	-
Relation-Type Index + LSTM/GRU (BUZAABA; AMAGASA, 2019)	PC	Tempo de Treinamento	8h
Pre-Trained BERT + Índice Investido + Rank (LUKOVNIKOV *et al.*, 2019)	Titan X GPU	-	-

3.3 Abordagens que usam QA com Templates

3.3.1 Abordagens Template-Based Question Answering (TBQA)

Dentre muitas alternativas de sistemas de QA, o *Template-Based Question Answering* (TBQA) surge como uma proposta promissora para resolver este problema (UNGER *et al.*, 2012). Os sistemas TBQA visam classificar a pergunta fornecida como entrada pelo usuário usando linguagem natural em padrões de consulta (*query templates*), que são expressos em linguagens de consulta formais como SPARQL e SQL.

Tabela 7 – Desafios do estado da arte para sistemas de QA avaliados sobre *dataset* SimpleQuestions.

Desafios	**Descrição curta**	**Tarefa**	**Próximos passos**
Lexical Gap	Consultas em banco de dados não deve ser expresso usando o mesmo vocabulário ou em o mesmo nível de abstração. Exemplo: casa, casarão, casebre.	Mapeamento	combinando recursos, reuso de bibliotecas, uso auxiliar de base de conhecimento
Ambiguidade	Consultas com palavras que podem ser interpretadas por diferentes entidades ontológicas ou construções semanticamente fracas. Exemplo: palavra "manga" pode se referir a parte da roupa ou a fruta.	Desambiguação	implícito, base de conhecimento sistemas conscientes
Consulta sobre Múltiplas Bases de Conhecimento	Combinam informações estruturadas e não estruturadas dentro de uma resposta.	Análise da Questão, Mapeamento	domínio especifico adaptadores, procedural
QA Multilíngue	Mediando usuários expressando necessidades de informação em seus próprios idiomas e os dados semânticos. Fortemente dependente de gramáticas.	Mapeamento	multilíngue bases de conhecimento
Questões Complexas	As consultas não são necessariamente estruturadas na base de conhecimento da mesma forma que na pergunta.	Construção de Consulta, Desambiguação	questões não factuais, independente de domínio

Segundo Diefenbach *et al.* (2018), uma das principais vantagens desses métodos é o uso de modelos em linguagens formais de consulta que capturam padrões complexos, mas que possuem aplicabilidade geral e relativamente conveniente, bastando que os *slots* sejam corretamente preenchidos por parâmetros informados pelo usuário.

Os sistemas TBQA podem ser classificados em dois grupos a partir da forma que geram os *templates*. O primeiro grupo corresponde aos sistemas que criam *slots* e o segundo é formado pelos sistemas que *extraem* os *templates* a partir de um *corpus*. Os trabalhos que usam *slots* tipicamente geram os *templates* de forma manual ou semiautomática. Esses *templates* são classificados para serem recuperados de acordo com a questão que é utilizada como entrada. Muitas dessas estratégias recorrem a um *parser* baseado em gramática para auxiliar nesse processo de casamento (*matching*) entre o *template* e a questão para preenchimento dos *slots*.

Em oposição a essa abordagem, temos os sistemas que buscam extrair os *templates* a partir de um ou mais *corpora* que possuem alguma relação com a ontologia a ser consultada. Dessa forma, esses sistemas tipicamente apresentam uma etapa demorada para extração dos *templates*, geralmente realizada de modo *off-line*. Note-se que essa etapa não se preocupa em

produzir um mapeamento entre a pergunta e a ontologia, mas entre a pergunta e o *template*. Várias abordagens adotam métricas de similaridade entre a pergunta e o *template* para resolver a associação do melhor *template* candidato a uma determinada pergunta.

Os sistemas TBQA apresentam algumas limitações, como a variabilidade linguística dos *templates* e a generalização da classificação do *template* para cada questão em linguagem natural. Segundo Diefenbach *et al.* (2018), a variabilidade linguística dos *templates* é a capacidade de gerar um bom número de *templates* que representam um possível conjunto de entradas em linguagem natural. Em relação à classificação do *template*, o desafio é identificar qual *template* é o mais adequado para responder a uma determinada questão.

3.3.2 Principais abordagens de Template-Based Question Answering (TBQA)

Unger *et al.* (2012) propõem uma análise da questão para produzir um *template* em SPARQL que espelhe diretamente a estrutura interna daquela pergunta. Para tanto, essa abordagem utiliza um *parser* baseado em um gramática para construção do *template*. A principal vantagem é o uso de quantificadores (e.g., *maior que*, *menor que*, etc) na criação de um modelo.

UncertainTQA (ZHENG *et al.*, 2015) constrói os *templates* de forma automática através de conjuntos de perguntas em linguagem natural e consultas SPARQL. A principal contribuição dessa estratégia é um algoritmo de similaridade de grafos baseado em junção sobre grandes grafos sujeitos à restrição de distância de edição. A similaridade ocorre sobre os vértices e as arestas de concatenados entre os grafos. Essa estratégia apresenta como principal limitação o gerenciamento dessa busca no subgrafo, o que pode causar graves problemas de desempenho.

Os sistemas **KBQA** (CUI *et al.*, 2017) e **QUINT** (ABUJABAL *et al.*, 2017) aceitam conjuntos de pares de questões e respostas como entrada e geram os *templates* a partir dessa informação. O KBQA extrai de textos os *templates* a partir de um conjunto de regras previamente definidas, utilizando ainda um *parser* probabilístico para melhorar o desempenho. Por sua vez, QUINT se propõe a aprender automaticamente os *templates* de consulta em enunciados alinhados à função das perguntas do usuário emparelhadas com suas respostas. Essa abordagem tenta vislumbrar a sequência completa de derivação do enunciado até a resposta final, de forma "explicável". Ou seja, explica como a estrutura sintática da pergunta foi usada para produzir a estrutura de uma consulta SPARQL.

Na abordagem proposta por Zheng *et al.* (2018), os *templates* são construídos a partir das entidades presentes nas sentenças existentes na Wikipédia. O autores utilizaram a ferramenta

DBPediaSpolight (MENDES *et al.*, 2011) para reconhecer cada entidade em uma sentença e depois construir um padrão desta a ser usada nos passos posteriores do algoritmo de construção da consulta SPARQL.

O trabalho de Zafar *et al.* (2018) apresenta um sistema de QA que foca na geração de questões e, mais especificamente, nas questões complexas. A principal contribuição deses autores é um modulo gerador de questões que pode ser incluído em qualquer sistema de QA. Esse módulo é construído a partir do treinamento de modelo de classificação baseado em Tree-LSTM e que leva em consideração a estrutura sintática da questão.

Em Zheng *et al.* (2019), o sistema de QA sugerido utiliza a interação como forma de refinar a consulta e obter um melhor resultado. Os usuários interagindo com o sistema podem reduzir ambiguidades e possíveis erros. Para diminuir o custo da interação, os autores propõem uma formalização para a realização de uma interação mais eficiente, além de uma técnica de busca prévia da resposta objetivando a redução do tempo de espera do usuário pela resposta.

Huang *et al.* (2019) propuseram uma solução de sistema de QA baseado em *embedding* de KBQA visando recuperar o conhecimento presente no texto, chamado de KEQA. A ideia principal do trabalho é representar as entidades e o predicado em um vetor de baixa dimensionalidade para que as informações presentes no grafo de conhecimento sejam preservadas. O KEQA utiliza uma métrica de distância cuidadosamente projetada para retornar a entidade mais próxima das outras codificadas. Os experimentos mostrados por Huang *et al.* (2019) demonstram que o KEQA supera os métodos de QA-KG de última geração.

Bakhshi *et al.* (2020) sugere uma nova estrutura de similaridade orientada a grafo que utiliza *templates* e gera consultas a partir de itens semânticos previamente mapeados. Dessa forma, um problema de construção de consulta é mapeado em um problema de alinhamento de grafos. Para a redução do custo envolvido em comparações da similaridade, o algoritmo reduz a complexidade desta etapa analisando as arestas do grafo.

Lu *et al.* (2021) propuseram um sistema de QA baseado em *templates* para o domínio de *bugs* de *software*. A estratégia sugerida compreende em extrair os *templates* de forma automática a partir de textos sobre *bugs* postados em fóruns.

Dataset	**Ano**	**LS** ($\geq 5K$)	**QC**	**QM**	**S**	**QP**	**T**	**SC**	**Lang**
Free917 (CAI; YATES, 2013)	2013	Não	Sim	MAN	Não	Não	Não	Não	en
WebQuestions (BERANT *et al.*, 2013)	2013	Sim	Não	API	Não	Não	Não	Não	en
SimpleQuestions (BORDES *et al.*, 2015)	2015	Sim	Não	CROW	Sim	Não	Não	Não	en
ComplexQuestions (BAO *et al.*, 2016)	2016	Não	Sim	API	Não	Não	Não	Não	en
GraphQuestions (SU *et al.*, 2016)	2016	Sim	Sim	API	Sim	Não	Não	Não	en
WebQuestionsSP (YIH *et al.*, 2016)	2016	Não	Sim	CROW	Sim	Não	Não	Não	en
SimpleQuestions2Wikidata (DIEFENBACH *et al.*, 2017)	2017	Sim	Não	API	Não	Não	Não	Não	en
30M Factoid QA Corpus (SERBAN *et al.*, 2016)	2017	Sim	Não	GAB	Não	Não	Não	Não	en
LC QuAD (TRIVEDI *et al.*, 2017)	2017	Sim	Sim	GTP	Sim	Sim	Sim	Não	en
ComplexWebQuestions (TALMOR; BERANT, 2018)	2018	Sim	Sim	GTP	Sim	Sim	Sim	Não	en
ComplexSequentialQuestions (SAHA *et al.*, 2018a)	2018	Sim	Sim	GAB	Não	Não	Sim	Não	en
QALD9 (NGOMO, 2018)	2018	Não	Sim	MAN	Sim	Não	Não	Não	mult
LC-QuAD 2.0 (DUBEY *et al.*, 2019)	2019	Sim	Sim	GTP	Sim	Sim	Sim	Não	en
FreebaseQA (JIANG *et al.*, 2019)	2019	Sim	Sim	QUEST	Não	Não	Não	Não	en
ConvQuestions (CHRISTMANN *et al.*, 2019)	2019	Sim	Sim	GTP	Não	Não	Não	Não	en
CFQ (KEYSERS *et al.*, 2019)	2020	Sim	Sim	GAB	Sim	Sim	Sim	Não	en
RuBQ (KORABLINOV; BRASLAVSKI, 2020)	2020	Não	Sim	QUEST	Sim	Não	Não	Não	rs
VQuAnDa (KACUPAJ *et al.*, 2020)	2020	Sim	Sim	GTP	Sim	Não	Não	Não	en

3.4 Principais conjuntos de dados de QA sobre Grafos de Conhecimento - *QA over Knowledge Graphs (KGQA)*

Nesta seção, daremos uma visão geral sobre as principais coleções de dados em KBQA desenvolvidas até o momento. Nesse propósito, apresentamos mais adiante a Tabela 3.3.2, que relaciona os trabalhos encontrados na literatura, e a Tabela 8, que aponta os aspectos analisados nas KBQA quanto às técnicas utilizadas.

Considerado o primeiro conjunto de dados para KBQA, o Free917 (CAI; YATES, 2013) foi gerado de forma manual por duas pessoas sem que consultassem uma base de conhecimento. O único requisito levado em consideração era a diversidade de tópicos de perguntas. Além disso, cada questão não era escrita em linguagem natural, mas em forma lógica para consultas na Freebase.

Os trabalhos iniciais de KBQA (CAI; YATES, 2013; BERANT *et al.*, 2013; BORDES *et al.*, 2015) propunham apenas questões simples (factuais) no seu conjunto de dados. Assim, o primeiro conjunto de dados a propor questões complexas no seu conjunto de dados foi o trabalho de Bao *et al.* (2016). O trabalho de Bao *et al.* (2016), utiliza como base o conjunto de questões proposto por Berant *et al.* (2013), adicionando múltiplas restrições e transformando um recorte das questões simples em questões complexas.

Tabela 8 – Aspectos analisados nos trabalhos sobre conjunto de dados de QA sobre Grafos de Conhecimento

Técnica Usada	**Descrição da forma que a técnica funciona**
MAN	Criadas de forma totalmente manual.
API	Criadas de forma semi automática com API que sugere informações no momento de construção da consulta.
CROW	Geração manual utilizando ferramentas de crowdsourced
GTP	Geração de forma automática da consulta a partir de templates anteriormente definidos realizando paráfrase.
QUEST	Geradas de forma semi automática com o uso de questionários.
GAB	Geradas de totalmente automática baseada em modelos de aprendizado de maquina profundo

Berant *et al.* (2013) propõem o WebQuestions, um conjunto de dados para KBQA em linguagem natural com cerca de cem mil questões e respostas. O processo de construção se deu da seguinte maneira: as questões foram coletadas por meio da API de sugestões do Google; os autores alimentaram partes da pergunta inicial na API e repetiram o processo com as perguntas retornadas até que um milhão de questões fossem alcançadas; depois disso, cerca de cem mil questões amostradas de forma aleatória foram apresentadas aos trabalhadores do *Amazon Mechanical Turk (AMT)*, cuja tarefa era encontrar uma entidade de resposta no Freebase. Vale a pena ressaltar que a WebQuestions foi usada como base para a geração de conjunto de dados de questões complexas (BAO *et al.*, 2016; TALMOR; BERANT, 2018).

O conjunto de dados SimpleQuestions (BORDES *et al.*, 2015) é o maior disponível para KBQA criado de forma manual até o momento. Em vez de fornecer análises lógicas para questões existentes, a abordagem fornece uma questão em linguagem natural que é gerada com o auxílio de *Amazon Mechanical Turk (AMT)*.

Su *et al.* (2016) apresentaram um conjunto de dados construído de forma semiautomática para KBQA. O principal diferencial dessa abordagem é que, em vez de coletar questões de forma totalmente manual, as questões são geradas a partir de um conjunto de formas lógicas predefinidas. Assim, as questões resultantes passam apenas por uma validação manual feita por humanos.

Para citar trabalhos que propuseram melhorias em conjuntos de dados já existentes, Yih *et al.* (2016) propõem o enriquecimento semântico de Berant *et al.* (2013) com a utilização de consultas SPARQL. Como resultado, mostrou uma melhoria de 81,5% e, por consequência,

indicou uma melhoria substancial na qualidade do KBQA. Já no trabalho de Diefenbach *et al.* (2017), foi feito o enriquecimento semântico da SimpleQuestions fazendo uma correspondência semiautomática com dados da Wikidata.

Serban *et al.* (2016) apresentaram um grande conjunto de dados que foi gerado de forma totalmente automática. O objetivo principal ser utilizado como fonte de dados para treinamento de modelos baseados em tradução automática. A rede neural treinada pela proposta foi usada para gerar 30 milhões de pares de perguntas e respostas. Esses autores afirmam que as questões factuais geradas são qualitativamente comparáveis às questões reais geradas por humanos.

O trabalho de Ngomo (2018) propõe fornecer *benchmarks* atualizados para avaliar e comparar sistemas de QA, expressando sua necessidade de informação em linguagem natural e dados RDF. Como já falamos anteriormente, esse conjunto de dados é usado como padrão ouro em diversas abordagens de sistemas de QA,. Com base nesses conjuntos de dados, foram propostas duas derivações deles: LC-QuAD (TRIVEDI *et al.*, 2017) e LC-QuAD 2.0 (DUBEY *et al.*, 2019). O projeto LC-QuAD tem como principal objetivo expandir o conjunto de questões existentes no QALD. A principal diferença entre LC-QuAD (TRIVEDI *et al.*, 2017) e LC-QuAD 2.0 (DUBEY *et al.*, 2019) é o numero crescente de questões que a base apresenta, que subiu de cinco mil para trinta mil.

Em relação à geração de conjunto de questões complexas, destacam-se os trabalhos de ComplexWebQuestions (TALMOR; BERANT, 2018) e ComplexSequentialQuestions (SAHA *et al.*, 2018a). O primeiro trabalho fornece um conjunto de questões complexas inspiradas no WebQuestions que são respondidas através de um modelo de consulta na Web. Já no trabalho ComplexSequentialQuestions (SAHA *et al.*, 2018a), temos como principal novidade um conjunto de dados KBQA para questões complexas com replicas que são usadas para a construção de chatbots.

A abordagem por trás do conjunto de dados FreebaseQA (JIANG *et al.*, 2019) é um grande conjunto de dados de questões e respostas factuais, todas revisadas por humanos.

ConvQuestions (CHRISTMANN *et al.*, 2019) é um conjunto de dados criado com o auxílio de *Amazon Mechanical Turk (AMT)* sobre cinco domínios diferentes (Livros, Filmes, Futebol, Musicas e Series de TV). Assim como o trabalho de Saha *et al.* (2018a), suas questões possuem réplicas e podem ser usadas no desenvolvimento de chatbots.

Em Keysers *et al.* (2019), é proposto um novo método para construir sistematica-

mente um conjunto de dados, maximizando a divergência composta, o que garante uma pequena divergência para que o conjunto de treino e teste sejam substancialmente diferentes, produzindo portanto uma generalização do *corpus*. O conjunto de dados gerado contém 239.357 perguntas com 34.921 padrões de consulta e foi avaliado analisando a capacidade de generalização sobre três arquiteturas de algoritmos de aprendizado de máquina.

Korablinov e Braslavski (2020) propuseram a primeira base de conhecimento conjunta de dados para questões e respostas em russo (RuB/Q). O RuBQ é um conjunto de dados de alta qualidade e consiste em 1.500 questões russas de complexidade variável, suas traduções automáticas em Inglês, consultas SPARQL para o Wikidata, respostas de referência, bem como uma amostra do Wikidata de triplas contendo entidades em russo.

Nos trabalhos de VQuAnDa (KACUPAJ *et al.*, 2020) e ParaQA (KACUPAJ *et al.*, 2021), há um KBQA verbalizado. Neles, a resposta é verbalizada, podendo, assim, ter diversas aplicações no treinamento de modelos de aprendizado de máquina. A principal diferença entre VQuAnDa (KACUPAJ *et al.*, 2020) e ParaQA (KACUPAJ *et al.*, 2021) é que este utiliza uma técnica de paráfrase para as respostas verbalizadas, resultando em uma maior variação no conjunto de dados. ParaQA contém 5.000 pares de perguntas e respostas com 2 a 8 respostas parafraseadas exclusivamente para cada pergunta. Assim, considerando VQuAnDa e ParaQA, o último usa uma técnica de paráfrase para as respostas verbalizadas, resultando em maior variação no conjunto de dados.

3.5 Comparação entre as Abordagens Existentes para QA

As abordagens ontológicas vêm se destacando nos sistemas de QA. Isso pode ser constatado pelo grande número de trabalhos encontrados na literatura. Essas abordagens são classificadas em relação ao domínio, que pode ser aberto ou fechado.

Dentre outros pontos fortes em relação aos sistemas de QA que usam ontologias, destacam-se: o uso do *schema* da ontologia para enriquecer a construção da consulta; não necessitar de dados para a realização de uma etapa de treinamento; e a construção de consultas mais ricas. Como principais pontos fracos, podemos destacar a necessidade de conhecimento prévio do *schema* utilizado na ontologia e a realização de etapas manuais no processo (e.g., o mapeamento de elementos).

Em relação aos sistemas que usam QA e Aprendizado de Máquina profundo, analisamos abordagens de QA factuais construídas a partir de redes neurais profundas sobre o

corpus SimpleQuestions devido ao escopo relacionado ao nosso trabalho. A principal vantagem desse método é o uso de uma arquitetura simples e direta para a criação de um QA. A principal desvantagem é a utilização de *corpus* com uma grande quantidade e diversidade de exemplos.

No caso dos sistemas de QA que usam *templates* previamente criados, uma das principais vantagens é a redução da distância semântica entre a ontologia e as questões. A principal desvantagem se demonstra tanto na dificuldade em automatizar a geração de *templates* quanto no mapeamento com a questão em linguagem natural gerada.

Outro aspecto importante é a existência de um conjunto de dados KBQA. Apresentamos os principais conjuntos de dados de KBQA existentes no estado da arte e ressaltamos como principal vantagem, em termos de oportunidades de pesquisas, a quantidade desses conjuntos. Porém, identificamos a baixa variabilidade nas questões presentes nas bases como principal problema desses conjuntos de dados.

4 UM FRAMEWORK BASEADO EM SENSO COMUM PARA SISTEMAS DE PERGUNTAS E RESPOSTAS SOBRE GRAFO DE CONHECIMENTO

Neste capítulo, é apresentado o nosso *framework* baseado em *template* para resolver o problema de QA sobre KBQA. A resolução do problema é feita por meio das etapas de extração, expansão e recuperação dos dados. A etapa de extração é comparada sob duas estratégias, sendo uma abordagem gulosa e outra que faz uso dos recursos da expansão semântica. Também são explorados os temas relativos à geração de questões, ao modelo de vetorização e às técnicas de indexação para recuperação de respostas.

4.1 Visão Geral da Abordagem

4.1.1 Um Framework baseado em Senso Comum para Sistemas de Perguntas e Respostas sobre Grafo de Conhecimento

A nossa estratégia, *Um Framework baseado em Senso Comum para Sistemas de Perguntas e Respostas sobre Grafo de Conhecimento*, consiste em um *framework* para sistemas de QA baseado em *templates* com um conjunto de dados KBQA — usando conhecimento de senso comum para melhorar a qualidade dos *templates* e dos dados extraídos. Esse *framework* faz a extração de relações do texto em linguagem natural usando uma ferramenta de OpenIE. A partir dessas relações, geramos um conjunto de *templates* associados a KGs e que são expandidos usando uma base de conhecimento de senso comum. Dessa extração, delineamos um conjunto de dados de KBQA que será usado no sistema de QA baseado em *templates*. Por fim, os *templates* gerados são associados às perguntas de entrada construídas em linguagem natural por meio de uma métrica de similaridade acoplada a uma estratégia de indexação. Assim, a consulta em SPARQL é gerada e executada no *triplestore*. Apresentamos uma visão geral do nosso trabalho na Figura 6.

Assim como outros sistemas propostos na literatura e que se baseiam na construção de *templates* a partir da extração de informação, nossa arquitetura é dividida em duas etapas: *offline* e *online*.

4.1.1.1 Etapa Offline

Esta etapa tem com o objetivo de "minerar" *templates* e demais metadados que permitam identificar o *template* mais adequado para uma pergunta q. É realizada de forma *offline*

por ser responsável pelas atividades mais intensivas em termos de processamento e memória. A etapa offline é dividida nos seguintes módulos: *Natural Language Pattern Generation*; *Template Expansion*; e *Template Vectorization*.

Natural Language Pattern Generation: Devido à flexibilidade da linguagem natural, um predicado no grafo de conhecimento pode ser descrito usando diferentes expressões idiomáticas. Isso afeta fortemente a habilidade de compreensão da questão em linguagem natural por algoritmos. Para resolver o problema, é proposta uma abordagem para construir *templates* que apenas explora o grafo de conhecimento G e um *corpus* de texto C. A ideia principal é encontrar padrões de linguagem natural de C para cada tripla $e \in G$. Aqui propomos duas estratégias para implementar essa etapa, sendo uma abordagem "gulosa" e outra que utiliza o OpenIE.

Na estratégia gulosa, primeiro encontram-se as entidades presentes nas sentenças do *corpus* de texto C. Partindo da premissa de que uma sentença $s \in C$ contém as entidades v_1 e v_2, que são candidatas para a tripla e. Para nos auxiliar nessa tarefa, usaremos um reconhecedor de entidades nomeadas. Além disso, assume-se que é possível substituir v_1 e v_2 em s por seus tipos(*e.g.* relação de subsunção entre conceitos) para obter um padrão de linguagem natural. Essa estratégia será explicada mais detalhadamente a seguir.

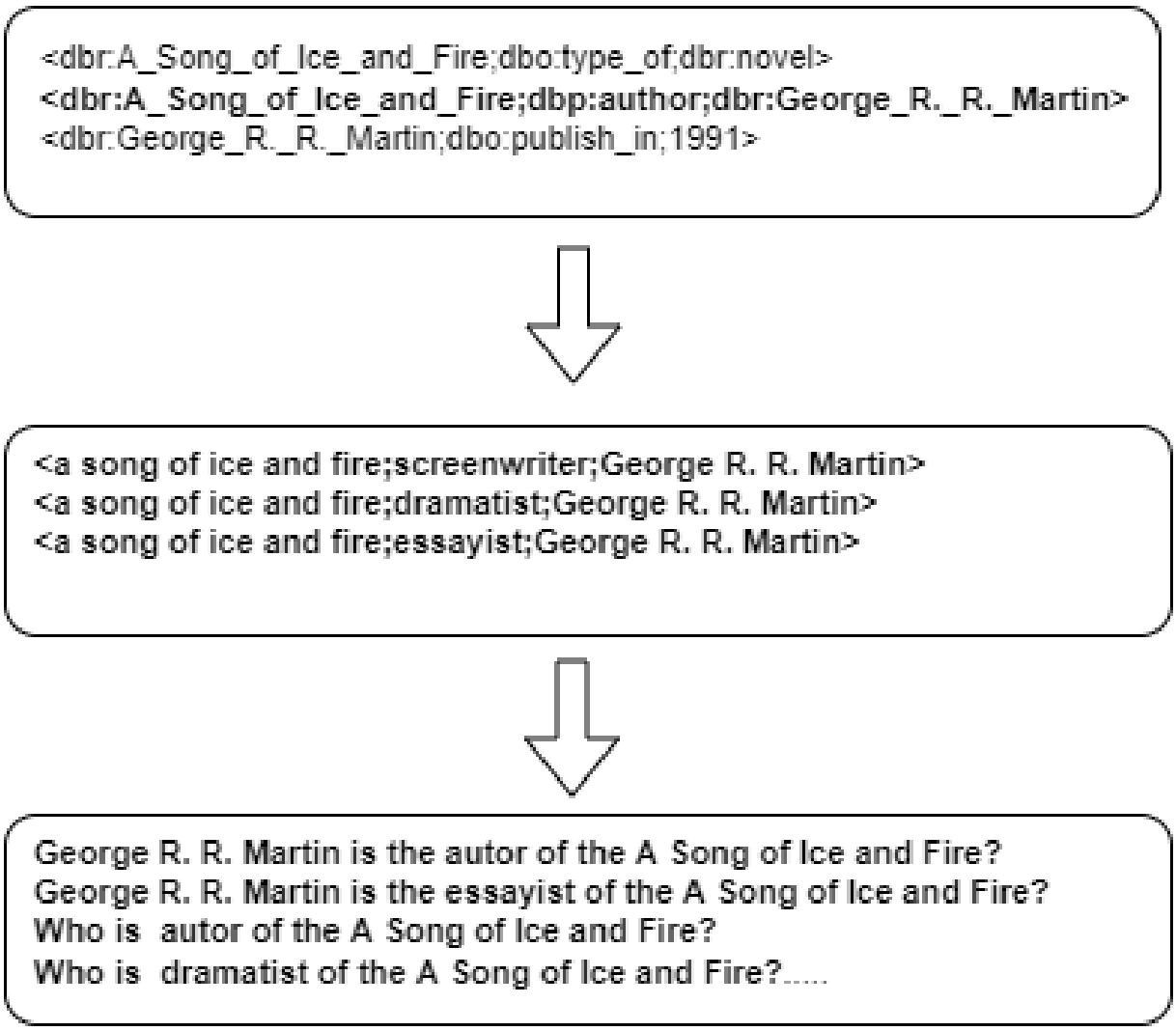
Já na estratégia *Open IE*, diferentemente da estratégia gulosa, está ferramenta é aplicada sobre o conjunto de sentenças S do *corpus* de texto C. Existindo uma sentença $s \in S$ que contém as entidades v_1 e v_2, candidatas para a tripla e, procedemos da seguinte maneira: após aplicar o OpenIE sobre s, obtemos como resultado uma relação no formato $\langle p_1, r, p_2 \rangle$. Então, verificamos se p_1 e p_2 condizem com as entidades v_1 e v_2. Dessa forma, esperamos não precisar procurar as entidades na sentença completa, reduzindo, assim, o espaço de busca. Com isso, esperamos identificar as entidades v_1 e v_2, que são candidatas para tripla e. E em seguida, substituímos v_1 e v_2 em s por seus tipos para obter um padrão de linguagem natural. Maiores detalhes dessa estratégia serão apresentados mais adiante neste capítulo.

Template Expansion (TE): Essa módulo é responsável pela expansão linguística de padrões em linguagem natural extraídos usando uma base de conhecimento comum como base auxiliar. Além disso, módulo também realiza a geração de um conjunto de questões em linguagem natural que será usado na fase de *matching* – entre questão do usuário e questão associada ao template.

A expansão linguística é alcançada a partir das relações encontradas usando uma ou

mais bases de conhecimento. As relações são escolhidas de acordo com as variações semântico-léxicas desejadas pelo usuário. Por exemplo, na base *ConceptNet*, há relações do tipo *isA*, *SimilarTo*, *Synonyms* e *relatedTerm*, entre outras. Serão analisadas nessa fase apenas as entidades presentes nas triplas, de modo que as relações extraídas entre as entidades não serão modificadas no processo. Na Figura 5, temos um exemplo de como ocorre essa expansão.

Figura 5 – Exemplo de como é feita a expansão dos templates.



Fonte: Elaborado pelo autor (2022).

Em relação à geração de questões em linguagem natural, concentramo-nos na geração automática de questões do tipo *WH-Questions* factuais. Nosso objetivo nessa subetapa é criar um conjunto de perguntas em linguagem natural a partir das sentenças ou das relações (quando há o uso do OpenIE), para que estas sejam vetorizadas na próxima etapa. Para tanto, será utilizada alguma técnica de geração de *embeddings* a partir de texto. Como resultado, obtemos

um conjunto de dados de KBQA que será utilizado pelo nosso sistema de QA em fases seguintes.

Template Vectorization: Módulo com o objetivo principal de produzir uma codificação das questões geradas na etapa anterior de modo que tal codificação seja útil para guiar o processo de associação entre *templates* e perguntas. Tradicionalmente, essa codificação era produzida usando métodos heurísticos ou funções de *hash*, técnicas que foram gradativamente substituídas pelas redes neurais convolucionais profundas em diversos domínios, especialmente recuperação da informação (LECUN *et al.*, 2015).

Isso posto, essa codificação pode ser produzida por meio de um modelo de aprendizado de máquina profundo treinado especialmente para a tarefa do sistema de TBQA, resultando em vetores multidimensionais altamente discernitivos, porém a um considerável custo computacional (SILVA *et al.*, 2020). Alternativamente, pode-se usar um modelo pré-treinado para tarefas gerais de NLP (DEVLIN *et al.*, 2018), seja diretamente ou para transferência de aprendizado.

4.1.1.2 Etapa Online

A etapa *online*, por sua vez, lida com a parte do sistema que envolve interação com o usuário para produzir a resposta à sua questão. Essa etapa é formada pelos seguintes módulos: *Question Matching*; *Query Formulation*; *Query Processing*.

Question Matching: Inicialmente, temos como entrada a pergunta em linguagem natural inserida pelo usuário. Essa questão é codificada em um vetor de baixa dimensionalidade e está vinculada ao *template*, de modo que seja possível realizar um *matching* por meio de uma medida de similaridade ou de um algoritmo de aprendizado de máquina. Vale a pena ressaltar que essa tarefa de *matching* tem um alto custo computacional ($O(n)$) devido à busca sobre uma grande quantidade de *templates* gerados – por isso, também propomos o uso de estratégia de indexação para resolver esse problema.

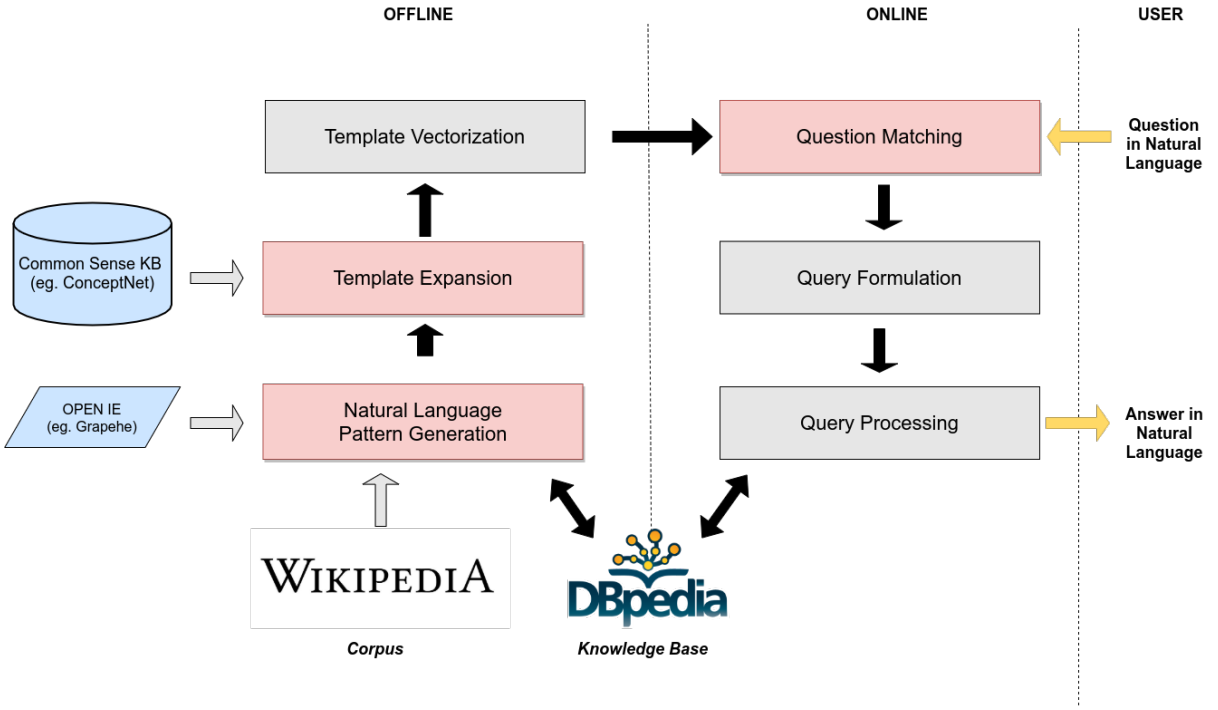
Query Formulation: Esse módulo é responsável por realizar a decodificação dos *templates* recuperados na etapa anterior, *Question Matching*, a fim de, efetivamente, construir as consultas em SPARQL que responderão às perguntas do usuário. Nossa técnica produz dois tipos de consultas: consultas factuais e consultas do tipo sim/não (i.e., binárias).

Query Processing: A última fase do sistema, *Query Processing*, realiza a execução da consulta em um *triplestore*[1]. O *triplestore* é um sistema de banco de dados projetado para armazenar e recuperar identidades que são construídas a partir de coleções de triplas.

[1] https://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html

Uma visão geral dessa arquitetura é ilustrada na Figura 6, onde estão destacados em vermelho os módulos nos quais se encontram as principais contribuições desta tese. Cada componente de cada etapa desta arquitetura será detalhado e exemplificado a seguir.

Figura 6 – Arquitetura da solução de QA proposta neste trabalho, com ênfase nas contribuições já implementadas, destacadas em vermelho.



Fonte: Elaborado pelo autor (2020).

A Figura 7, por sua vez, exibe uma visão geral sobre todo o processo a partir de um fluxograma.

4.2 *Natural Language Pattern Generation* (NLPG)

Dado um corpus textual C, nossa proposta é extrair cada tripla $e = v_1, r, v_2 \in G$ e processar todas as sentenças de C que estão relacionadas a e. Diz-se que uma sentença s está relacionada com e se s contém as duas entidades v_1 e v_2 presente no texto. Observe que o *corpus* do texto pode ser muito grande, portanto seria muito custoso explorar todo o corpus D exaustivamente para a verificação de cada tripla. Propomos, então, duas estratégias para construir de maneira eficiente os padrões de linguagem natural: usando uma estratégia "*gulosa*" e outra através do *Open IE* para auxiliar a criação desses padrões. A seguir, descrevemos e exemplificamos cada uma delas.

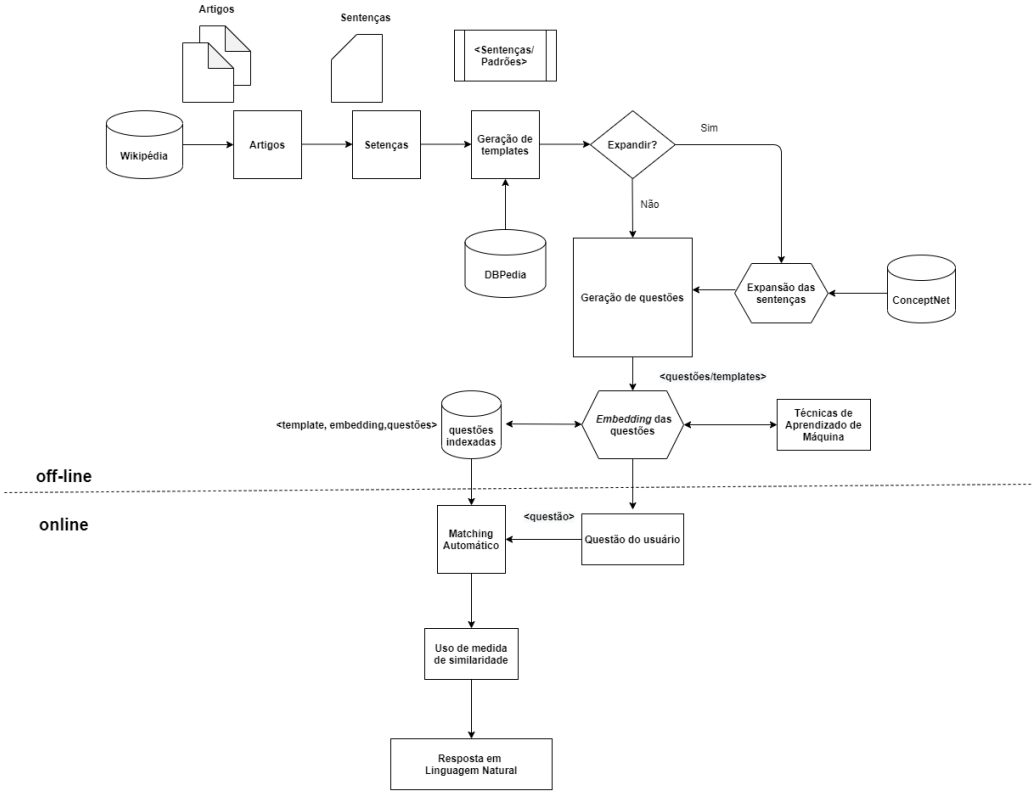
4.2.1 Estratégia Gulosa

A primeira estratégia é chamada de *gulosa*, pois nela buscamos encontrar todas as entidades presentes no texto para, assim, extrair o padrão em linguagem natural. Essa estratégia apesar de ser custosa é bem parecida com a utilizada por Zheng *et al.* (2018). Na qual para cada sentença s do corpus C, primeiro identificamos o conjunto de entidades presentes nela, denotado por E. Objetivando encontrar as entidades presentes no texto, utilizamos o *DBpedia Spotlight* (DAIBER *et al.*, 2013; MENDES *et al.*, 2011). Aplicamos o *DBpedia Spotlight* em s, o que promove a descoberta de todas as palavras w com entidades dentro da DBpedia. Para um melhor funcionamento do algoritmo, excluímos sentenças muito grandes, limitando o processamento apenas àquelas que possuem no máximo 200 palavras.

Depois de aplicar o *DBpedia Spotlight* (MENDES *et al.*, 2011), vamos encontrar um conjunto das entidades V, onde $v_1, v_2, \cdots v_n \in V$. Para cada entidade $v_1, v_2, \cdots v_n$, temos um tipo/*type* associado $c_1, c_2, \cdots c_n$. Por fim, para cada par de entidades (v_1, v_2), temos associado um conjunto de palavras W que foram encontradas entre as entidades. Esse conjunto de palavras é candidato a gerar relações entre as entidades.

Os padrões de linguagem natural gerados nesse processo podem não ser simples

Figura 7 – Fluxograma mostra uma visão geral sobre o processo da nossa proposta.

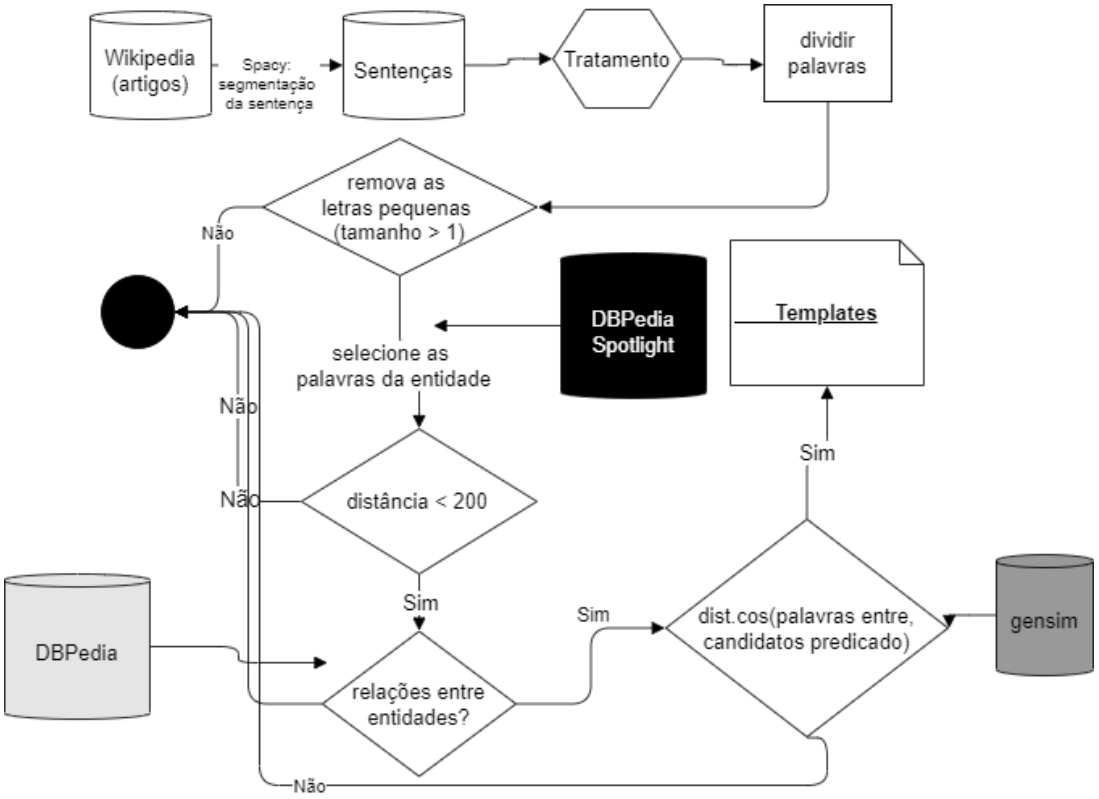


Fonte: Elaborado pelo autor (2022).

— há possibilidade de conter várias relações —, indicando que tais padrões não podem ser empregados para decompor a questão de entrada ou um "*template*". Assim, é preciso descartar alguns padrões de linguagem natural que não possuam relações válidas.

Em geral, duas entidades podem ter várias relações no grafo de conhecimento subjacente. Para selecionar as relações válidas, usamos duas estratégias combinadas. A primeira é verificar, tomando as entidades (v_1, v_2), qual relação já existente se aproxima da relação extraída no texto. No exemplo *Mel Gibson é o diretor, roteirista e protagonista do filme Coração Valente*, temos as triplas: $< Coração\_Valente, diretor, Mel\_Gibson >$; $< Coração\_Valente, roteirista, Mel\_Gibson >$; e $< Coração\_Valente, protagonista, Mel\_Gibson >$. Realizamos, então, uma consulta no KG, retornando o conjunto R que contém todas as relações existentes entre o par de entidades $< Coração\_Valente, Mel\_Gibson >$.

Figura 8 – Fluxograma relativo ao algoritmo guloso.



Fonte: Elaborado pelo autor (2021).

A segunda estratégia é associar a relação extraída do texto com a relação retornada na consulta. Para fazer isso, geramos os *embeddings* das duas relações encontradas e realizamos a sua similaridade semântica. Para a construção dos *embeddings*, utilizamos um modelo pré-treinado *fastText* (BOJANOWSKI *et al.*, 2016). A escolha desse modelo se deve aos bons resultados mostrados para modelos treinados a partir da Wikipédia (JOULIN *et al.*, 2016). Para

medida de similaridade, usamos a distância cosseno adotada pelo modelo*fasText* e que já é amplamente usada em diversos outros *embeddings* (MIKOLOV *et al.*, 2013). A partir disso, ordenamos os pares de relações mais similares e substituímos o predicado que possui o maior valor de similaridade. Por fim, extraímos a relação e montamos o *template* no formato $t =< e_1; rel; e_2 >$, onde e_1 e e_2 são entidades e *rel* representa um predicado no *KG*. Exemplificamos o Algoritmo Guloso por meio da Figura 8.

A função do algoritmo guloso foi implementada com o nome "**nlpGreedy**". Cujos parâmetros de entrada é representado pela seguinte tupla <sentences, model, conn, initial_min_similarity, threshold_sentence, version>:

- sentences: um vetor de tuplas que contém id_artigo e o texto que deve ser tratado;
- model: objeto com o modelo para vetorizar as palavras, o trabalho aceita tanto "gensim" ou "fasttext";
- conn: objeto que estabelece conexão com banco de dados, em SQLITE, e possui os métodos de interface;
- initial_min_similarity(=0.87, padrão): As palavras são comparadas em sua versão vetorizada, dadas pelo modelo adotado, seja gensim ou fasttext ou outro, devem ser comparadas e esta variável defini o valor mínimo;
- threshold_sentence(=10, padrão): limite de distância entre as palavras dentro da sentença;
- version(='v0', padrão): versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

O retorno da função é o tempo gasto para execução, mas a saída mais importante não é retornada pela função, mas escrita no banco de dados. Ao escrever os resultados no banco de dados evita-se o acúmulo memória durante a execução da função. O resultado de cada saída é uma inserção em uma tupla na tabela "*triple*". Em resumo a tupla inserida é <id_sentence, id_open_ie, predicate, subject_spotlight, subject_type, object_spotlight, object_type, pred_word_text, order_entity, version> e cada elemento é descrito a seguir:

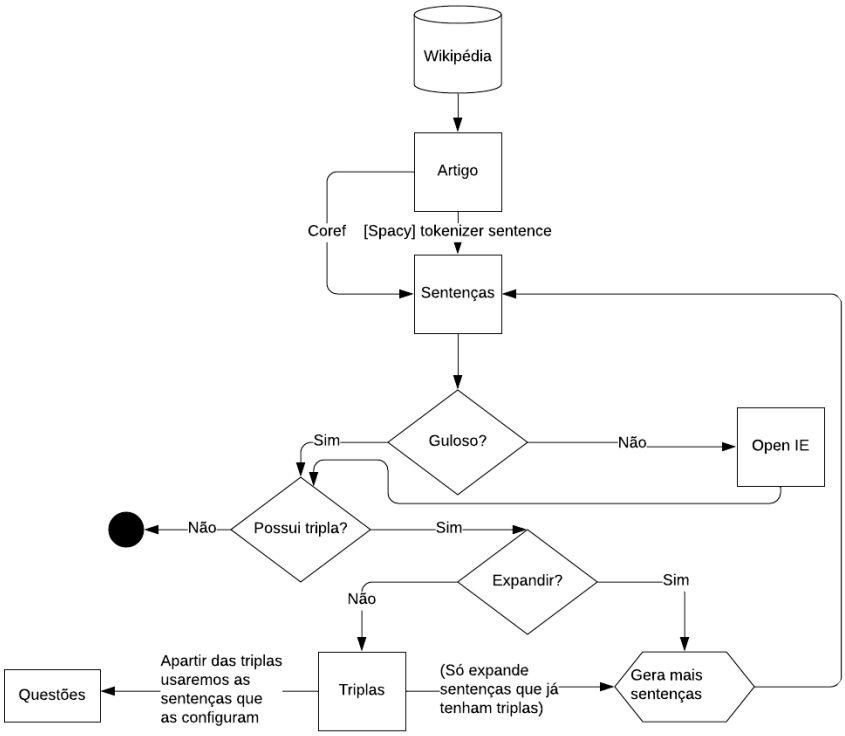
- id_sentence: identificador da sentença original;
- id_open_ie: identificador do Open IE que neste caso deve ser adicionado como -1, uma vez que não existe uma referência a tripla da Open IE;
- predicate: texto (*string*) com o predicado identificado;
- subject_spotlight: o sujeito refere-se a quem faz ação do predicado e é representado por um conjunto de metadados relacionados a extração do DBPedia Spotlight ((MENDES *et*

al., 2011)) registrados em formato JSON;

- subject_type: tipo de sujeito de acordo com a ontologia;
- object_spotlight: o objeto refere-se a quem sofre ação do predicado e é representado por um conjunto de metadados relacionados a extração do DBPedia Spotlight (MENDES *et al.*, 2011) registrados em formato JSON
- object_type: tipo de objeto de acordo com a ontologia;
- pred_word_text: termo em formato canônico do predicado;
- order_entity: o sujeito e o objeto pode ser encontrado na ontologia nessa ordem ou o contrário com o objeto vindo primeiro;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

4.2.2 Estratégia OpenIE

Figura 9 – Fluxograma relativo ao algoritmo que usa OpenIE.



Fonte: Elaborado pelo autor (2021).

Como dito anteriormente, a abordagem se propõe a utilizar uma ferramenta de *Open*

IE para realizar a extração de relações em sentenças — tanto por conveniência, quanto para diversificar os predicados identificados e aumentar a capacidade de generalização do nosso sistema para diversos contextos. Outra característica importante dessa abordagem é a restrição do espaço de busca na geração de *templates*. Isso vai ao encontro da nossa hipótese **H2**, que busca melhorar a qualidade dos *templates* gerados.

De acordo com Jurafsky (2000), uma ferramenta de *Open IE* transforma as informações não estruturadas, expressas no texto em linguagem natural, em uma representação estruturada. Isso é apresentado na forma de tuplas relacionais, que consistem em um conjunto de argumentos e um texto que denota uma relação semântica entre eles: $< arg_1; rel; arg_2 >$. Note-se ainda que, como o *Open IE* não se limita a um pequeno conjunto de relações de destino previamente conhecidas, espera-se que a ferramenta seja capaz de maximizar os tipos de relações encontrados em um texto, o que contribui para o enriquecimento semântico dos *templates* a serem gerados.

Neste ponto, deve estar claro para o leitor que a entrada da ferramenta de *Open IE* é uma sentença na qual se produz um conjunto de triplas como saída. Isso pode ser exemplificado usando o *corpus* Wikipédia e DBpedia como a ontologia. Nossa estratégia é usar o **OpenIE5** (SAHA *et al.*, 2018b) como ferramenta de *Open IE* para extração das relações, pois essa versão apresenta bons resultados para o estado-da-arte na área, principalmente considerando a extração de papéis semânticos (*Semantic Role Labeling*) (NIKLAUS *et al.*, 2018; CHRISTENSEN *et al.*, 2011). Note-se que a arquitetura proposta prevê a possibilidade de que o **OpenIE5** seja substituído por outra ferramenta no futuro.

A nossa estratégia *OpenIE* foi construída em duas etapas: identificação das entidades e identificação das relações. A identificação das entidades procede da seguinte forma. ara cada sentença s do *corpus* D, aplicamos o *Open IE*. Como resultado, teremos um R que contém um conjunto de $r_1, r_2, \ldots r_n$, onde $r_k = arg_1; rel; arg_2$. Então, aplicamos o DBpedia Spotlight (DAIBER *et al.*, 2013; MENDES *et al.*, 2011) nos pares (arg_1, arg_2) para encontrar o para correspondente (e_1, e_2) no grafo de conhecimento.

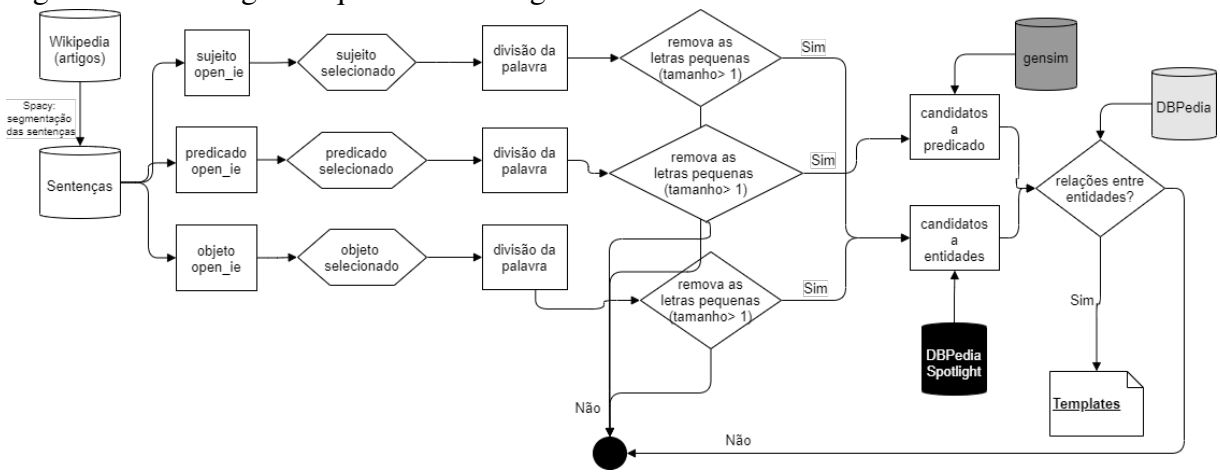
Depois de aplicar o DBpedia e encontrar os pares (e_1, e_2) para (arg_1, arg_2), associamos a cada par de entidades os seus respectivos tipos (*type*) associados c_1, c_2. É possível, agora, partir para a segunda etapa, que consiste em identificar as relações presentes nos *templates*.

O processo de identificação das relações é análogo ao adotado pela estratégia "*gulosa*" nessa mesma tarefa, consistindo em selecionar as relações válidas, utilizando uma combinação

de duas estratégias. A primeira é verificar, tomando as entidades (e_1, e_2), qual relação já existente se aproxima da relação extraída no texto. Usando o *OpenIE* no exemplo *Mel Gibson é o diretor, roteirista e protagonista do filme Coração Valente*, temos os triplas <*Coração_Valente, diretor, Mel_Gibson*>, <*Coração_Valente, roterista, Mel_Gibson* > e <*Coração_Valente, protagonista, Mel_Gibson*>. Realizamos, então, uma consulta no KG, retornando o conjunto *R*, que tem todas as relações existentes entre <*Coração_Valente, Mel_Gibson*>.

A segunda estratégia é associar a relação extraída do texto com auxílio do *OpenIE* e a relação retornada na consulta. Para tanto, geramos os *embeddings* das duas relações encontradas e realizamos a sua similaridade semântica. Para a construção dos *embeddings*, utilizamos um modelo pré-treinado Fastext (BOJANOWSKI *et al.*, 2016) assim como na estratégia "*gulosa*". A partir disso, ordenamos os pares de relações mais similares e substituímos o predicado que tem um maior valor de similaridade, assim extraímos a relação e montamos o *template* no formato $t =< e_1; rel; e_2 >$, onde e_1 e e_2 são entidades e *rel* representa um predicado no *KG*. Exemplificamos o algoritmo na Figura 10.

Figura 10 – Fluxograma que detalha o algoritmo.



Fonte: Elaborado pelo autor (2021).

A função do algoritmo Open IE foi implementada com o nome "**getPatternsSentences**", encontrado em "nlg.py", para extrair da sentença os "patterns" que são as triplas (sujeito, predicado e objeto) e depois é submetido há um processo similar ao "nlpGreedy", chamado de "**getTripleByOpenIE**", encontrado em "nlp_oie.py". Onde "**getPatternsSentences**" possui como entrada a tupla <sentences, openIE, database_filename, debug=True, version>:

- sentences: um vetor de tuplas que contém id_artigo e o texto que deve ser tratado;
- openIE: objeto com a referência a função OpenIE a ser utilizada, por padrão está o Reverb, mas foi o utilizado o OpenIE5 nas chamadas;

- database_filename: caminho para o arquivo do banco de dados SQLite;
- debug: modo para imprimir resultados;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

A saída desta função são as triplas com sujeito, predicado e objeto, aqui elas são chamadas de patterns e são adicionadas a tabela “open_ie”. Portanto, o seu retorno é a tupla <id_sentence, sub_word_text, pred_word_text, obj_word_text, expanded, version>, listada a seguir:

- id_sentence: identificador numérico da sentença;
- sub_word_text: sujeito extraído;
- pred_word_text: predicado extraído;
- obj_word_text: objeto extraído;
- expanded: chave que identifica se essa tripla extraída é derivada de outra, caso seja, então ela deve possuir o identificador da original caso contrário possui -1. No caso desta função “getPatternsSentences” o retorno sempre será -1, pois são triplas originais;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

A função “**getTripleByOpenIE**” é similar ao “nlpGreedy”, possuindo como parâmetros de entrada a seguinte tupla: <patterns, conn, model, initial_min_similarity, version>, vide lista:

- patterns: um vetor de tuplas que contém id_sentence, id_open_ie e os textos de sujeito, predicado, objeto extraídos pelo OpenIE;
- conn: objeto que estabelece conexão com banco de dados, em SQLITE, e possui os métodos de interface;
- model: objeto com o modelo para vetorizar as palavras, o trabalho aceita tanto “gensim” ou “fasttext”;
- initial_min_similarity: As palavras são comparadas em sua versão vetorizada, dadas pelo modelo adotado, seja gensim ou fasttext ou outro, devem ser comparadas e esta variável defini o valor mínimo;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

Observe que ao contrário da versão gulosa não existe o parâmetro “threshold_sentence”,

uma vez que ele é usado para definir as potenciais palavras dentro de uma janela para encontrar a tripla. Algo que tornou-se dispensável porque é feito através do OpenIE.

4.3 *Template Expansion* (TE)

Na etapa anterior (Natural Language Pattern Generation (NLPG)), temos como resultado um conjunto n de pares (s,t), onde s é a sentença que foi extraída do *corpus* C e t é um *template* no formato $t =< e_1; r; e_2 >$ gerado por qualquer uma das estratégias apresentadas anteriormente ("*Gulosa*" ou OpenIE). Consideremos esse conjunto como entrada na etapa de Template Expansion (TE).

A etapa de TE é responsável pela expansão linguística das triplas extraídas usando uma base de conhecimento comum como uma base auxiliar. Essa expansão linguística é realizada a partir das relações encontradas em uma ou mais bases de conhecimento (*KB*) utilizadas inicialmente.

Para definir uma *KB* de conhecimento de senso comum, usaremos a definição de Young *et al.* (2018). Assumimos que uma base de conhecimento de senso comum é composta por um conjunto de assertivas A sobre os conceitos *CC*. Cada assertiva $a \in A$ assume a forma de um tripla $< c_1, r, c_2 >$, onde $r \in R$ é uma relação entre $c1$ e $c2$, como *IsA, CapableOf*, entre outras. Temos c_1, c_2 como conceitos em *CC*.

Note-se que o conjunto de relações R é normalmente muito menor que *CC*. Um fato interessante é que c pode ser uma única palavra (por exemplo, *"cachorro"* e *"livro"*) ou uma expressão com várias palavras (por exemplo, *"fique de pé"* e *"faça compras"*).

Das várias *KB* de conhecimento comum existentes, as mais conhecidas são WordNet (MILLER, 1995), FrameNet (BAKER *et al.*, 1998), ConceptNet (SPEER *et al.*, 2017) e InferenceNet (PINHEIRO *et al.*, 2010). Na nossa abordagem, usaremos a ConceptNet por ter demostrado ser o estado-da-arte dos recursos linguísticos, inclusive com melhorias nas aplicações de *embeddings* de palavras, bem como com a resolução de analogias no estilo The Satisfiability Problem (SAT) (SPEER *et al.*, 2017).

A ConceptNet (LIU; SINGH, 2004; HAVASI *et al.*, 2007; SPEER; HAVASI, 2012; SPEER *et al.*, 2017) é uma rede semântica que representa o conhecimento de senso comum coletado por meio do projeto *Open Mind Common Sense* (OMCS) (SINGH *et al.*, 2002), cujo objetivo é coletar sentenças que expressam fatos da vida comum a partir de colaboradores voluntários pela Internet.

De acordo com Pinheiro *et al.* (2010), a motivação do projeto que mantém a ConceptNet é expressar os fatos que as pessoas sabem comumente sobre o mundo — conhecimento de senso comum —– em uma rede de relacionamentos entre conceitos, a qual possa servir como importante recurso linguístico para suportar o raciocínio semântico de sistemas computacionais. Esse tipo de conhecimento é importante porque, quando as pessoas se comunicam, seus proferimentos acontecem sobre suposições implícitas e básicas, as quais suportam e explicam boa parte dos raciocínios necessários para um bom nível de entendimento e, consequentemente, uma boa comunicação entre os interlocutores.

A ConceptNet tem um conjunto amplo de relações. As relações são escolhidas de acordo com as variações semântico-léxicas desejadas pelo usuário. Por exemplo, na base *ConceptNet* temos relações do tipo *isA*, *SimilarTo*, *Synonyms*, *relatedTerm*, entre outras. Vale a pena limitar o número de relações utilizadas na consultas por questões de eficiência e viabilidade, pois se trata de uma *KB* que possui uma enormidade de nós. Isso posto, a API ConceptNet-Lite[2] será utilizada para realizar as consultas sobre essa *KB*.

A expansão é realizada em cima do s que está no par (s,t). Assim o fazemos, em cima da sentença, porque a usaremos mais à frente no processo de *matching* com a pergunta de entrada no sistema. Além disso, é importante destacar que a expansão não é feita sobre toda a sentença, mas apenas de algumas palavras de determinadas classes gramaticais. Vamos expandir as palavras das classes[3] Substantivo (***NN, NNP, NNPS, NNS***) e Adjetivo (***JJ, JJR, JJS***). Essas classes foram escolhidas devido às suas relevâncias semânticas no entendimento do texto.

Assim, teremos como resultado dessa etapa de TE um conjunto de sentenças atrelado a um *template*. Na Figura 11, apresentamos um fluxograma que retrata o funcionamento dessa etapa de expansão.

O processo de expansão pode ser enviado para duas funções diferentes, uma referente a expansão de sentenças, do algoritmo guloso, chamada de "**getConceptNetExpansion**" e outra referente a expansão pelo OpenIE chamada "**getConceptNetExpansion4OpenIE**", ambas em "te.py". A tupla de entrada do "getConceptNetExpansion" é <sentences, conn, expansion_function, limit_words, limit_expansions, version>, listado abaixo:

- sentences: tupla com dados da sentença, incluindo o texto e o identificador da mesma;
- conn: objeto que estabelece conexão com banco de dados, em SQLITE, e possui os métodos de interface;

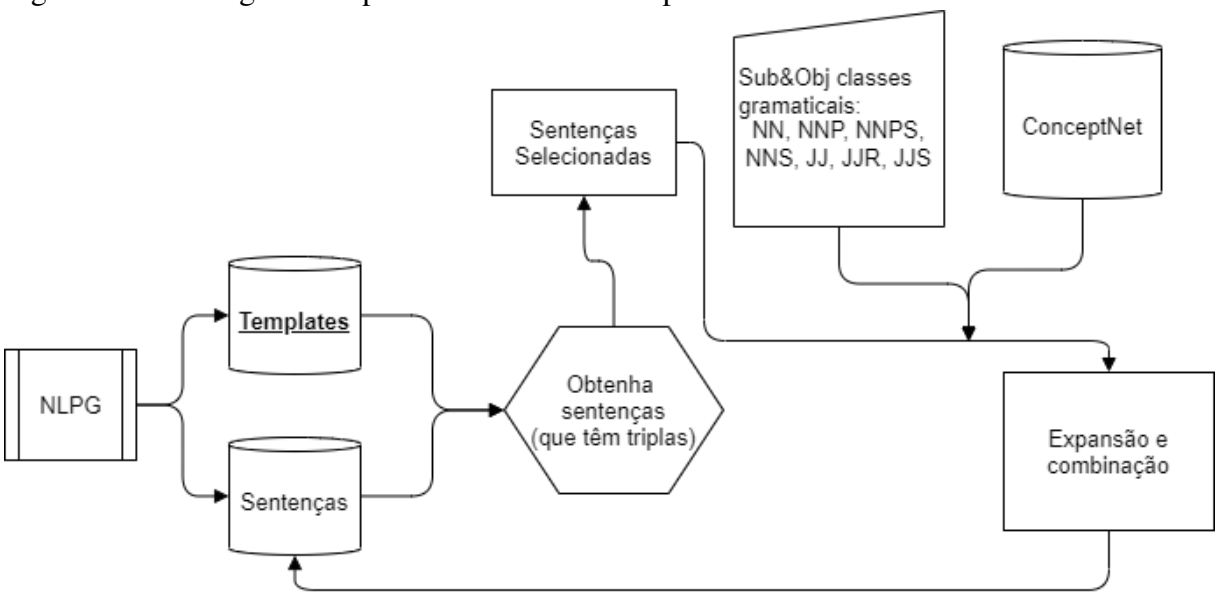
[2] https://pypi.org/project/conceptnet-lite/
[3] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

- expansion_function: referência a função que gera expansão, neste caso o padrão é apontado para API do ConceptNetLite (“cl.getConceptsByRelation”);
- limit_words: se o número de palavras candidatas for muito alto existe a possibilidade de serem termos fora de contexto, por isso, pode-se limitar o número de palavras candidatas, por padrão são 16 palavras;
- limit_expansions: as palavras candidatas geram sentenças variadas e pode-se estabelecer um limite caso o número de combinações seja muito alto;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

As entradas da função “getConceptNetExpansion4OpenIE” é apresentado na tupla <patterns, conn, limit_words, limit_expansions, version>, similar a versão gulosa.

- patterns: tupla com dados da tripla (sujeito, predicado e objeto), incluindo o texto e o identificador da mesma;
- conn: objeto que estabelece conexão com banco de dados, em SQLITE, e possui os métodos de interface;
- limit_words: se o número de palavras candidatas for muito alto existe a possibilidade de serem termos fora de contexto, por isso, pode-se limitar o número de palavras candidatas, por padrão são 16 palavras;
- limit_expansions: as palavras candidatas geram sentenças variadas e pode-se estabelecer um limite caso o número de combinações seja muito alto;

Figura 11 – Fluxograma Expansão usando o ConceptNet.



Fonte: Elaborado pelo autor (2021).

– version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

A saída de ambas as funções é o registro em três tabelas no banco de dados SQLite. Uma delas registra as novas sentenças na tabela “sentence” enquanto a versão expandida registra na tabela “open_ie”, porém em ambas as tabelas terão a coluna de auto-referência “expanded” com valor maior do que -1, apontando para sentença ou tripla (pattern) original. Além dessas tabelas duas tabelas citadas, uma terceira é utilizada em ambas as funções, a chamada tabela “expanded_word” que registra cada palavra de expansão separada, permitindo análises futuras a respeito das mesmas. A tupla que representa tabela “expanded_word” é listada a seguir:

– id_sentence: identificador numérico da sentença;
– id_open_ie: identificador numérico da tripla (ou “pattern”, com sujeito, predicado e objeto) extraída pelo OpenIE, caso seja uma expansão sobre a sentença então id_open_ie é igual a -1;
– pos_word: posição de onde começa a palavra na sentença original, a contagem é feita sobre os caracteres;
– len_word: tamanho da palavra original;
– word_src: palavra original na sentença;
– word_changed: nova palavra que substitui o termo original;
– type_variation: qual a classe da relação do termo original com o expandido, por exemplo “isA”;
– version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

4.3.1 Geração Automática de Questões (GAQ)

Como resultado da etapa anterior, temos um par (S,t), onde S é um conjunto de sentenças e t é um *template*. Note-se que a expansão só foi realizada com sucesso se e somente se $|S| > 1$. A etapa de GAQ tem por objetivo gerar um conjunto de questões factuais (Natural Language Query (NLQ)) a partir das sentenças extraídas e expandidas. Isso será feito dada a importância NLQ para análise das consultas *SPARQL* e para o treinamento de modelos de aprendizado de máquina.

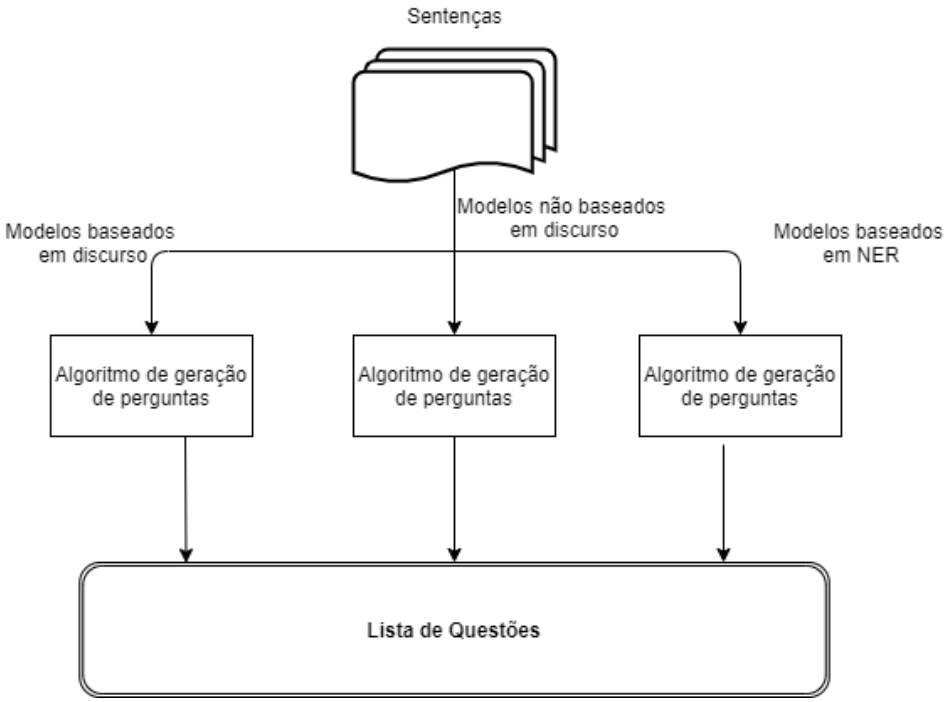
A geração de perguntas de forma automática foi proposta por Rus *et al.* (2008). Segundo os autores, a geração de questões de forma automática é a tarefa de gerar perguntas

automaticamente a partir de várias entradas, como texto bruto, banco de dados ou representação semântica. Essa definição indica que o tipo de entrada para a geração da pergunta pode variar: pode ser, por exemplo, uma frase, um parágrafo ou outro tipo. A geração automática das NLQ factuais tem diversas aplicações, a exemplo da geração automática de provas no âmbito educacional e, de forma geral, em áreas importantes da NLP como sistemas de perguntas e respostas, geração de diálogos, recuperação de informações e sumarização.

Para geração automática das NLQ factuais, utilizaremos a estratégia proposta por Jain *et al.* (2019). Essa escolha se deve principalmente à sua simplicidade na construção das NLQ. Em particular, a abordagem proposta por Jain *et al.* (2019) soa razoavelmente adequada à proposta desta tese por buscar satisfazer os seguintes critérios:

– O padrão de sentença extraído deve ser independente de domínio; e
– Deve extrair pontos importantes na frase de origem e criar uma pergunta factual.

Figura 12 – Fluxograma que detalha o funcionamento da geração de questões proposto por Jain *et al.* (2019).



Fonte: Jain *et al.* (2019)

Na Figura 12, podemos observar como a proposta de Jain *et al.* (2019) é estruturada. No caso, a técnica se baseia em três módulos para geração de questões, os quais são detalhados a seguir:

– **Modelos baseados em marcadores de discurso:** Nessa abordagem, é procurada na sentença a palavra com a maior relevância para construção de questões Tais palavras são

chamadas de *marcadores*. Podemos citar como marcadores “porque”, “como resultado” e “desde”, dentre outros. Quando esses marcadores são encontrados, é feita uma normalização da sentença (padronização para forma minúscula, *tokenização* e *POS-Tagger*). Logo após, é verificado se a sentença possui verbo auxiliar. Se possuir, é verificado se o verbo está presente em uma lista preestabelecida e é feita uma construção da questão com base em padrões preestabelecidos. Caso contrário, a questão é construída com base apenas no verbo presente no sujeito presente;

- **Modelos que não utilizam marcadores de discursos:** As sentenças que não têm nenhum marcador de discurso entram nessa categoria. Podemos citar como exemplo, as sentenças assertivas ou declarativas do tipo “sim” ou “não”. Nesse caso, é usado o verbo auxiliar como a palavra inicial da questão. Outro procedimento utilizado é gerar a combinação das *tags*. Essa técnica é semelhante ao procedimento usado nos modelos baseados em marcadores do discurso, estratégia apresentada anteriormente; e
- **Modelos baseados em NER (*Name Entity Recognition*):** Essa estratégia é usada quando não existe nenhum marcador de discurso e as sentenças analisadas são declarativas simples ou frases do tipo “sim” ou “não”. Como não há marcadores, utilizamos o *POS-Tagger* para encontrar os substantivos e depois verificamos quais desses substantivos são entidades nomeadas usando um reconhecedor de entidade nomeada[4]. Encontrada a entidade nomeada, ela é associada a um padrão de *Wh-Question* existente.

Como resultado dessa etapa, temos triplas na forma (s,t,q), onde s é uma sentença extraída do *corpus*, t é um *template* e q é uma questão factual.

A implementação da função de geração de questões, chamada de “insertQuestion_tn”, inclui a inserção delas na tabela “question”, portanto, ele recebe apenas três parâmetros na tupla <triples, version, conn>, contudo os parâmetros de “version” e “conn” já foram citados a exaustão neste trabalho, a maior diferença é a lista de tuplas da tabela “triple”, com a tupla <id_sentence, id_open_ie, predicate, subject_spotlight, subject_type, object_spotlight, object_type, pred_word_text, order_entity, version>, como é apresentado na lista no final da subseção 4.2.1 e 4.2.2. A saída na tabela “question” é listado a seguir:

- id_triple_greedy: identificador da tabela “triple”, apesar do nome desta coluna não representa um vínculo ao algoritmo guloso. Em realidade, o nome desta coluna precisa ser refatorada para “id_triple”;

[4] https://spacy.io/api/entityrecognizer

- id_sentence: identificador da tupla de origem na tabela “sentence”;
- id_open_ie: identificador da tupla de origem da tabela “open_ie”;
- question: texto da questão gerada;
- question_types: é a questão reescrita com o tipo dos objetos identificados;
- type_question: as questões podem ser de um dos tipos: *’What’, ’Who’, ’Where’, ’When’, ’Why’, ’Whose’, ’Whom’, ’Which’, ’How’*;
- version: versão é uma marcação feita para controlar execuções, alterações e testes durante etapas do pipeline.

4.3.2 Geração de Questões em SPARQL

Durante a etapa de geração de questões em SPARQL, o algoritmo recebe como entrada um conjunto de pares (*questão*, *tripla*), gerando como resultado para cada par um conjunto de **padrões de questões** que o representam. Para cada par, uma *questão* é um objeto que armazena o texto em linguagem natural gerado na etapa anterior, acrescido do seu tipo (e.g., “*Yes/No*”, “*What*”, “*Who*”). Por sua vez, uma $tripla$ de um dado par é um objeto contendo os tipos de sujeitos e objetos esperados para o padrão, além do predicado utilizado pelo padrão. Um padrão de questão gerado ao fim dessa etapa é composto por seu respectivo padrão de consulta SPARQL junto do padrão de questão em linguagem natural tratado.

O processo de construção dos padrões é executado de maneira *offline* em lotes, onde cada lote representa um conjunto de pares de *questão* e *tripla* armazenados no módulo de persistência. Note-se que é possível configurar o tamanho e a quantidade (caso seja necessário processar apenas um recorte dos dados) de lotes utilizados. Ao final do processamento de cada lote, os resultados computados são armazenados no módulo de persistência.

Para cada par (*questão*, *tripla*), é executado o método **CreateSPARQLTemplates** dos Algoritmos 1 e 2. Inicialmente, esse método cria variáveis com identificadores únicos para o sujeito e objeto da tripla do padrão (linhas 3 e 4); em seguida, são geradas *statements* SPARQL, definindo cada variável e respeitando seus tipos devidos (linhas 5 e 6); dando continuidade, é construído o corpo da consulta SPARQL utilizando as variáveis definidas anteriormente junto do predicado do padrão (linhas 7-11); Após, o corpo da consulta gerado é utilizado para construir a consulta final dependendo do tipo da questão (linhas 12-21). Nessa etapa, consultas do tipo “*Yes/No*” são mapeadas para consultas *ASK*, enquanto os demais tipos são mapeados para consultas tradicionais do tipo *SELECT*, onde são projetadas todas as variáveis; por fim, são

construídos os padrões de questões possíveis para o dado par (linhas 14-16,19-20).

Na criação do padrão de questão executada pelo método **CreateTemplate**, são definidas as variáveis de entrada e saída do padrão junto do seu padrão de questão em linguagem natural. Para todos os tipos de questões são gerados, no mínimo, dois padrões (linhas 18-19), onde o primeiro padrão define as variáveis "sujeito" e "objeto" como entrada e saída respectivamente, enquanto o segundo padrão realiza o processo inverso. Especialmente no caso de questões do tipo "*Yes/No*", é construído um terceiro padrão contando com ambas as variáveis como entradas e o resultado da função *ASK* (booleano) como saída. Durante este processo, menções às variáveis de saída são removidas do padrão de questão em linguagem natural (linhas 25-27), sendo esse novo padrão o utilizado para comparação com as questões dos usuários. Por fim, o método **SaveTamplete** (linha 28) é responsável por armazenar temporariamente o padrão criado (tanto linguagem natural, quanto SPARQL) e as variáveis de entrada e saída para serem consolidados no módulo de persistência ao final do processamento do lote.

Durante a etapa de consulta, é preciso garantir que a questão em linguagem natural dada como entrada pelo usuário esteja no mesmo formato do padrão em linguagem natural gerado. Para isso, a questão de entrada deve ser pré-processada para serem identificadas suas entidades referenciais, sendo estas substituídas pelas URIs de seus tipos.

Para ilustrar o resultado dessa etapa, tomemos como entrada a dupla de questão "*<http://dbpedia.org/ontology/Person> is the president of the <http://dbpedia.org/ontology/Country>?*", tendo tripla (dbo:Country, dbp:leaderName, <dbo:Person). Esse padrão gerará como resultado os três seguintes padrões de questões:

- QA Template utilizando sujeito e objeto como entradas:
  - Questão: "<http://dbpedia.org/ontology/Person> is the president of the <http://dbpedia.org/ontology/Country>?"
  - SPARQL:PREFIXES... ASK?country a dbo:Country. ?person a dbo:Person. ?country dbp:leaderName ?person.
  - Entrada: $[?country, ?person]$; Output: $[ASK]$
- QA Template utilizando sujeito como entrada:
  - Questão: "<http://dbpedia.org/ontology/Person> is the president of the country?"
  - SPARQL:PREFIXES... ASK?country a dbo:Country. ?person a dbo:Person. ?country dbp:leaderName ?person.
  - Entrada: $[?country]$; Saída: $[ASK]$

– QA Template usando objeto como entrada:
  – Questão: “ a person is the president of the <.../Country>?”
  – SPARQL:PREFIXES... ASK?country a dbo:Country. ?person a dbo:Person. ?country dbp:leaderName ?person.
  – Entrada: $[?person]$; Saída: $[ASK]$

Um possível exemplo de questão para esse padrão seria “*Is Joe Biden the president of the united states of america?*”. Para a construção de uma consulta SPARQL no intuito de responder à pergunta de um usuário, o sistema só precisa substituir as variáveis de entrada (denotadas no *template* de QA) no modelo de consulta SPARQL pelo URI da entidade encontrado na pergunta do usuário. Portanto, no exemplo anterior, a consulta resultante seria a seguinte:

PREFIXES... ASKdbr:United$_S tatesadbo : Country.dbr : Joe_B idenadbo : Person.dbr : United_S tatedbp : leaderNamedbr : Joe_B iden.$

Algoritmo 1: Algoritmo para criação do template de consulta SPARQL

Entrada: question q,triple t

question q,triple t **início**

$triplesBody \leftarrow CreateBody()$;
$subjectVar \leftarrow CreateVar(t.subjectType)$;
$objectVar \leftarrow CreateVar(t.objectType)$;
$subjectDef \leftarrow DefineVar(subjectVar, t.subjectType)$;
$predicate \leftarrow CreatePredicate(t.predicate)$;
$triple \leftarrow CreateTriple(subjectVar, predicate, objectVar)$;
$triplesBody.addTriple(subjectDef)$;
$triplesBody.addTriple(objectDef)$;
$triplesBody.addTriple(triple)$;
$q.type =$ 'ASK' $sparql \leftarrow ASK(triplesBody)$
CreateTemplate(q,$sparql$,$[subjectVar, objectVar]$, $AskVar()$) ;
CreateTemplate(q,$sparql$,$[subjectVar]$, $AskVar([objectVar])$) ;
CreateTemplate(q,$sparql$,$[objectVar]$, $AskVar([subjectVar])$) ;
$sparql \leftarrow SELECT(triplesBody)$;
CreateTemplate(q,$sparql$,$[subjectVar]$,$[objectVar]$) ;
CreateTemplate(q,$sparql$,$[objectVar]$,$[subjectVar]$) ;
CreateTemplate(q,$sparql$,$[subjectVar, objectVar]$,[]) ;

fim

Algoritmo 2: Algoritmo para salvar o template gerado em Algoritmo 2

SalveTemplate **Entrada:** question q, $sparql$, $inputVars$, $outputVars$

SaveTemplate(q,$sparql$,$inputVars$,$outputVars$) **início**

  $questionTemplate \leftarrow q$;

  para $outputVar$ ***in*** $outputVars$ **faça**

    $q \leftarrow RemoveCitation(q, outputVar)$;

  fim

  SaveTemplate(q,$sparql$,$inputVars$,$outputVars$) ;

fim

A Figura 13 ilustra como uma sentença de entrada pode ser expandida. Termos e conceitos são substituídos por opções equivalentes para gerar sentenças parafraseadas. Uma única sentença pode conter vários termos e conceitos, cada um com muitas opções correspondentes. Consequentemente, nosso método pode produzir inúmeras sentença parafraseadas e perguntas a partir de uma única sentença utilizando uma base de conhecimento de senso comum.

4.4 *Template Vectorization*

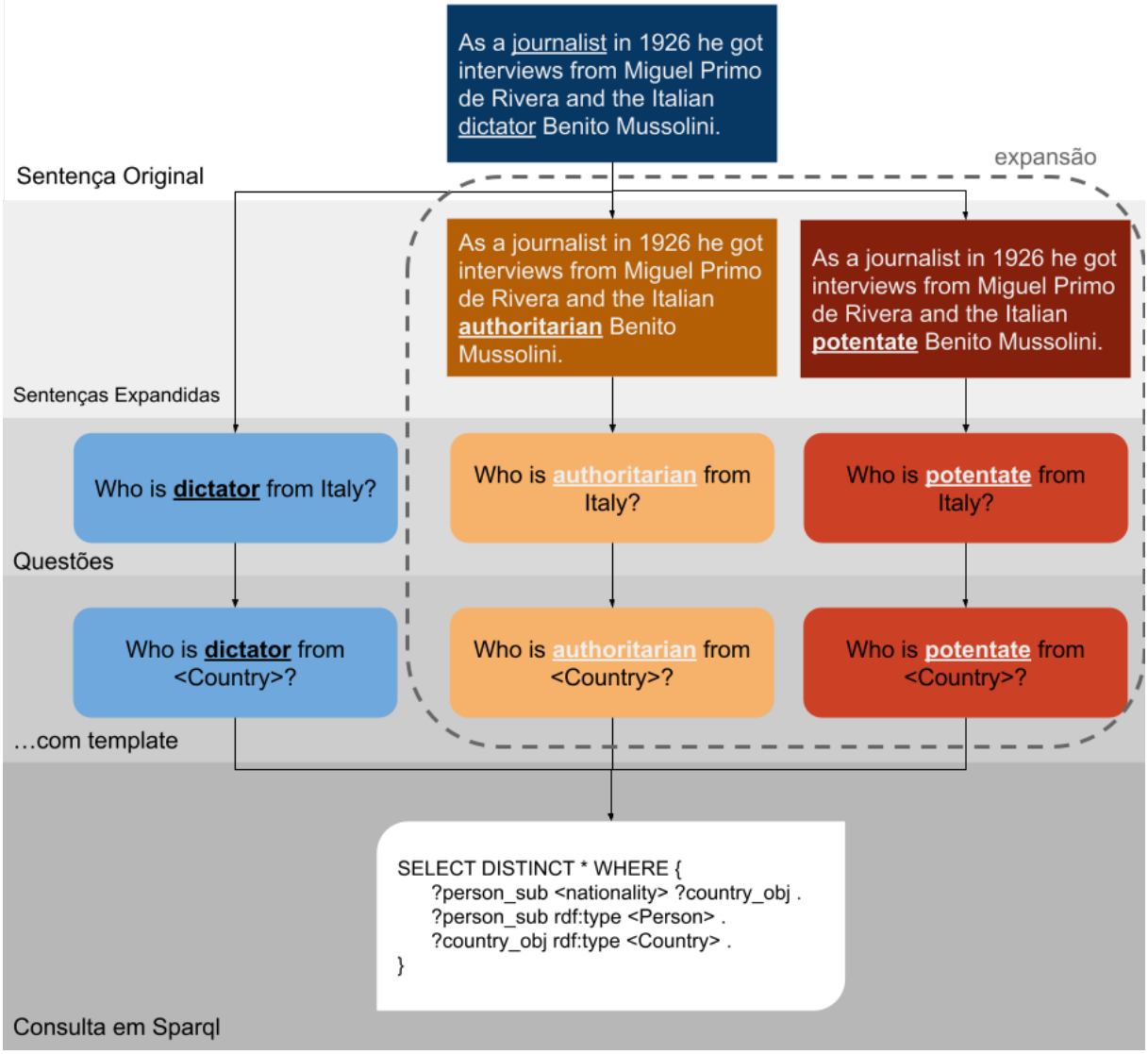
A etapa de *Template Vectorization* tem como principal objetivo produzir uma codificação das triplas geradas na etapa anterior de modo que tal codificação seja útil para guiar o processo de associação entre *templates* e perguntas. Após gerar os *templates* codificados, estes

são persistidos em um banco de dados ou estrutura de busca para facilitar a recuperação na etapa *online*. Como estratégia de codificação inicial, escolhemos o *fastText* porque se trata de uma abordagem de domínio público sob os termos da Licença MIT[5] e que disponibiliza um modelo pré-treinado com um *corpus* da Wikipédia[6].

A técnica de codificação usada no *fastText* recorre a uma rede neural "rasa" e fornece um vetor em relativa baixa dimensionalidade, contendo 300 componentes de ponto-flutuante, para *cada palavra* encontrada no modelo pré treinado de *embedding*. Entretanto, é necessário gerar um vetor para *cada questão* gerada pelo nosso *framework*. Como estratégia para resolver esse problema, cada questão vetorizada será construída a partir da média aritmética dos vetores

[5] https://github.com/facebookresearch/fastText/blob/main/LICENSE
[6] https://fasttext.cc/docs/en/english-vectors.html

Figura 13 – Um exemplo de como a expansão de uma sentença introduz a variabilidade da pergunta enquanto a resposta ainda é obtida pela mesma consulta.



Fonte: Elaborado pelo autor (2021).

das palavras. A seguir, a Equação 4.1, que formaliza nossa estratégia:

$$X = \frac{\sum_1^n \overline{x_i}}{n} \tag{4.1}$$

Também investigamos a estratégia de usar o vetor da palavra com maior valor para representar a questão vetorizada. Porém, essa estratégia não apresentou bons resultados em testes preliminares realizados. Além do *fastText*, também foi investigado o uso do *Universal Sentence Encoder* (CER *et al.*, 2018) pré-treinado, na sua versão 4[7] (USE4). Esse modelo foi desenvolvido para codificar textos de tamanho considerável, tais como frases e parágrafos curtos, visando atividades de cunho geral em NLP que demandem o reconhecimento de sequências de palavras. USE4 é disponibilizado para o domínio público sob os termos da licença Apache 2.0[8]. Esse modelo produz vetores com 512 componentes, o que pode ser considerada uma dimensão razoavelmente compacta para acomodar frases inteiras, especialmente se compararmos sua capacidade com o *fastText*. O modelo BERT pré-treinado (DEVLIN *et al.*, 2018) também foi investigado, porém foi abandonado devido aos resultados pouco convincentes obtidos nos experimentos preliminares, além do seu elevado custo computacional.

Como resultado dessa etapa, temos um conjunto de vetores representando cada questão. Uma observação importante é que, com o módulo *offline* do *framework*, podemos disponibilizar os *templates* para as aplicações que podem gerar os seus QA, mostrando um importante aspecto do *framework*, a portabilidade.

4.5 *Question Matching*

A etapa de *Question Matching, QM* é a primeira a ser executada pela porção *online* da abordagem proposta neste trabalho. Inicialmente, temos como entrada uma pergunta em linguagem natural inserida pelo usuário. Essa pergunta é codificada pelo mesmo método usado para codificar o *template*, de modo que a codificação resultante da pergunta se assemelhe à codificação do *template* de acordo com algum critério objetivo. Assim como na *Template Vectorization*, aqui utilizamos *fastText* (MIKOLOV *et al.*, 2018) ou USE4 (CER *et al.*, 2018) para gerar o *embedding* da questão de entrada.

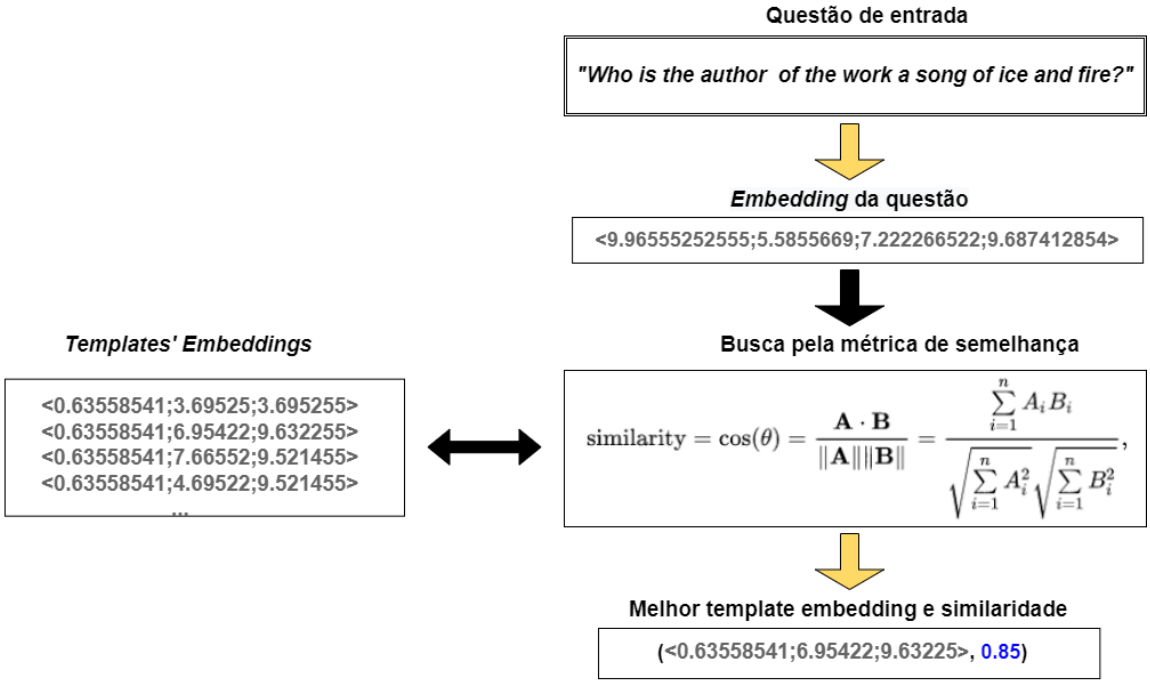
[7] https://tfhub.dev/google/universal-sentence-encoder/4
[8] https://opensource.org/licenses/Apache-2.0

4.5.1 Comparando Questões e Templates

A etapa de Question Matching (QM) recorre a uma métrica de similaridade entre a pergunta e o conjunto de respostas para realizar o *matching*. Note-se que isso pode ser realizado de maneira mais eficiente caso exista uma estrutura de dados especializada para lidar com essa métrica de similaridade.

Nossa investigação incluiu um estudo de diferentes medidas de distância para a etapa de QM: a distância cosseno, distância euclidiana e outras. Analisando as distâncias, não houve nenhuma discrepância entre elas. Escolhemos, enfim, a distância cosseno por ser mais consolidada entre as abordagens vistas na bibliografia. Computadas as distâncias, o módulo realiza o ranqueamento para que retorne o *template* cujo *embedding vector* seja o mais próximo, de acordo com a métrica de distância adotada, ao *embedding vector* da pergunta fornecida pelo usuário. Esse processo é exemplificado pela Figura 14.

Figura 14 – Funcionamento da etapa de *Question Matching*



Fonte: Elaborado pelo autor (2021).

Nosso *framework* gera um número grande de questões associadas a *templates*; um problema associado a isso é a busca pelo *template* da base. Para resolver, incluímos uma estratégia de busca baseada em *indexação* (*index*), que vamos explicar detalhadamente na próxima seção.

4.5.2 Indexação dos Templates

Devido à grande quantidade de informação gerada, a busca pelo *template* na execução do *matching* pode ser um dos momentos mais custosos da solução, i.e., um gargalo computacional a ser evitado na etapa *online*. Visando resolver tal problema, propomos um método de indexação usando técnicas de *Approximate Nearest Neighbors (ANN)* (GE *et al.*, 2013).

A principal ideia do ANN é pesquisar entre os vizinhos mais próximos de um vetor usado como entrada (*query*), pois, em muitos casos, os pontos da vizinhança são tão bons quanto o ponto exatamente mais próximo. Isso é particularmente interessante quando uma medida de distância é utilizada capturar com precisão as diferenças entre as questões, ou seja, pequenas diferenças não alteram significativamente no resultado final da busca Muja e Lowe (2009). A complexidade computacional dessa técnica é $O(n \log n)$.

Como trabalhamos com uma grande quantidade de questões, é necessário gerar um índice com antecedência, o que ocorre durante a etapa *offline*. O índice é construído com algoritmos ANN em memória e depois são salvos em disco para o uso eficiente durante a busca por um *template*. A principal vantagem de usar ANN é que, depois que o índice é construído, a pesquisa é realizada de forma extremamente rápida durante a etapa de QM no *framework* proposto, superando as outras abordagens de busca em tipicamente duas ordens de grandeza (MUJA; LOWE, 2014; FRANCO *et al.*, 2020a).

Assim, a ideia principal por trás da abordagem de ANN é obter uma melhoria considerável no desempenho da busca pelo *template*, porém sob o custo de uma perda aceitável na precisão, de modo que a solução use um resultado aproximadamente correto, mas rápido e viável para aplicações (MUJA; LOWE, 2009; AUMÜLLER *et al.*, 2020; FRANCO *et al.*, 2020a).

Vale ressaltar que essa abordagem é bastante usada em trabalhos relativos a imagens, porém não encontramos na literatura a aplicação dessa abordagem no contexto de KBQA. Por isso, nós escolhemos quatro métodos representativos de ANN que são amplamente usados em pesquisas e produtos reais (MUJA; LOWE, 2009; FRANCO *et al.*, 2020a; MUJA; LOWE, 2014; MALKOV; YASHUNIN, 2018; IWASAKI; MIYAZAKI, 2018; JOHNSON *et al.*, 2019) para serem avaliados em nossa estratégia. A seguir, listamos os principais métodos de ANN que utilizaremos no nosso trabalho:

– FLANN (MUJA; LOWE, 2009; MUJA; LOWE, 2014): foi uma das primeiras bibliotecas de domínio público a fornecer métodos ANN. Tem como principal característica incorporar

vários algoritmos aprimorados de busca de vizinho mais próximo em espaços vetoriais, como o uso de estrutura de dados como a *kd-tree* e *hashing* sensível à localidade. Por padrão, FLANN adota *kd-tree* para lidar com alta dimensionalidade e conjuntos de dados muito grandes, além de ser bastante eficiente na construção de índices. Essa implementação é geralmente a mais veloz nas etapas de construção e de acesso ao índice, apesar de nem sempre oferecer a melhor precisão (FRANCO *et al.*, 2020a);

– ANNOY (*Approximate Nearest Neighbors Oh Yeah*) (AUMÜLLER *et al.*, 2020): é usado pelo Spotify para recomendação de música. Sua implementação recorre a uma estrutura de dados de árvore chamada de *Binary Space Partition (BSP)*, que usa hiperplanos semelhantes aos usados para fins de renderização nos primeiros gráficos 3D. Uma floresta aleatória de árvores Binary Space Partition (BSP) é construída para melhorar os resultados da consulta, ou seja, quanto mais árvores, mais resultados precisos e menos "*desempenho (tempo) para maior precisão (qualidade)*";
– HNSW (*Hierarchical Navigable Small World Graphs*) (MALKOV; YASHUNIN, 2018): é um método baseado em grafo que separa links entre nós em um grafo de pesquisa de acordo com sua escala de comprimento em camadas diferentes, realizando pesquisas em um grafo em várias camadas. Consequentemente, a avaliação do nó mais próximo ocorre apenas em uma parte fixa necessária das conexões para cada elemento. As principais vantagens do método Hierarchical Navigable Small World Graphs (HNSW) é sua robustez e seu suporte à indexação incremental contínua;
– NGT (*Neighborhood Graph and Tree for Indexing Highdimensional Data*) (IWASAKI; MIYAZAKI, 2018): é um método de ANN baseado em grafos proposto inicialmente para trabalhar com espaço vetorial de grande escala e alta dimensão. NGT melhora o grafo de k-vizinho mais próximo (KNNG), ajustando o caminho e os graus de borda derivados de um KNNG, ou seja, uma otimização em termos do número de bordas de entrada e bordas de saída, influenciando a precisão da pesquisa e o tempo de consulta, respectivamente. Em particular, os autores descrevem como uma aceleração significativa pode ser obtida removendo arestas que podem ser substituídas por caminhos de busca alternativos. O NGT também oferece suporte à adição e remoção de itens da indexação, o que é um recurso importante para aplicativos do mundo real; e
– FAISS (*Facebook Artificial Intelligence Similarity Search*) (JOHNSON *et al.*, 2019): é um método criado inicialmente para otimizar buscas de vídeos e imagens utilizando vetores

de baixa dimensionalidade textuais. Basicamente o FAISS, utiliza uma técnica de busca não exata baseada em produto quantizado usando k-NN, que por fim, é executado em cima de uma Graphics Processing Unit (GPU).

4.6 Query Formulation

A etapa de *Query Formulation* é responsável por realizar a decodificação dos *templates* recuperados na etapa anterior, QM, a fim de, efetivamente, construir as consultas em SPARQL que responderão às perguntas do usuário.

Durante a etapa de construção da consulta, é preciso garantir que a questão em linguagem natural dada como entrada pelo usuário esteja no mesmo formato do padrão em linguagem natural gerado. A questão de entrada deve ser, então, pré-processada para serem identificadas suas entidades referencias, sendo estas substituídas pelas URIs de seus tipos.

Assim, vamos usar o mesmo algoritmo apresentado em 4.3.2 para preencher os campos necessários no *template* e repassar a consulta em SPARQL a fim de ser executada no próximo módulo.

4.7 Query Processing

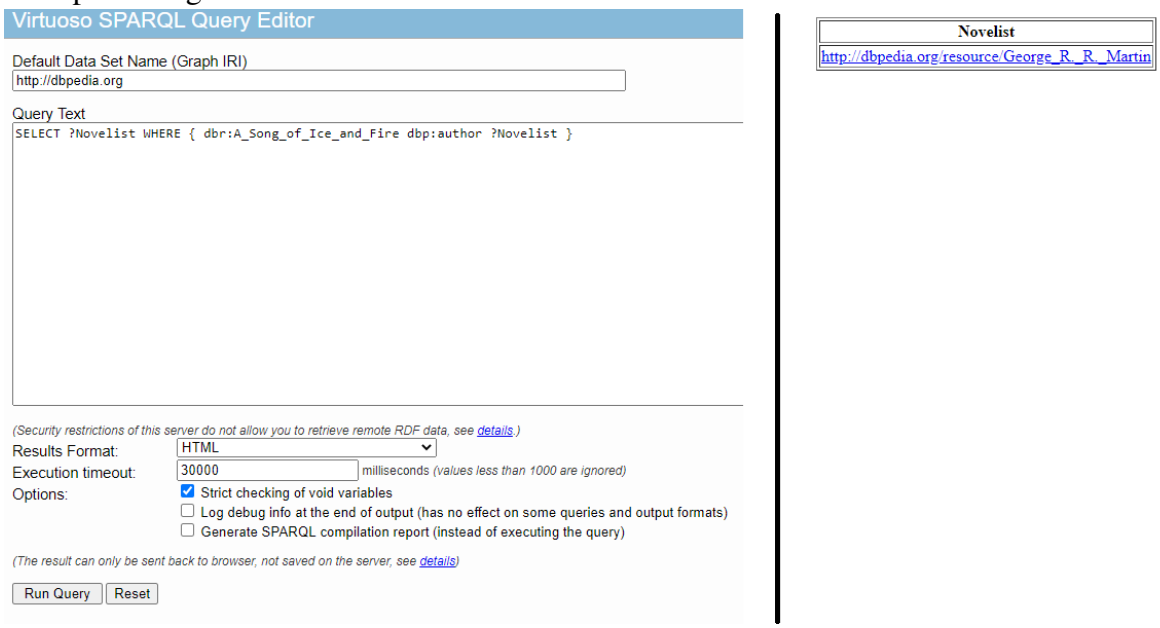
A última fase do sistema, *Query Processing*, realiza a execução da consulta em um *triplestore*[9]. O *triplestore* é um sistema de banco de dados projetado para armazenar e recuperar identidades que são construídas a partir de coleções triplas. Essas coleções triplas representam um relacionamento entre sujeito-predicado-objeto que corresponde mais ou menos à definição apresentada pelo padrão RDF.

Podemos citar como exemplos de *triplestore*, entre outros, MarkLogic[10], Jena[11] e Virtuoso[12]. Assim, a etapa de QP executa a consulta gerada na etapa anterior, Query Formulation (QF), no *triplestore*.

Na Figura 15, verificamos o exemplo da consulta executada no *triplestore* para o exemplo desenvolvido na Figura 14 usando a consulta formulada.

[9] https://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html
[10] https://www.marklogic.com/
[11] https://jena.apache.org/
[12] https://virtuoso.openlinksw.com/

Figura 15 – Processamento da consulta no *Endpoint* na etapa de *Query Processing* para o exemplo da Figura 14.



Fonte: Elaborado pelo autor (2022).

5 RESULTADOS

Detalhamos nesse capítulo os principais resultados obtidos, mais precisamente o conjunto de dados que geramos com associação simultânea de *templates*.

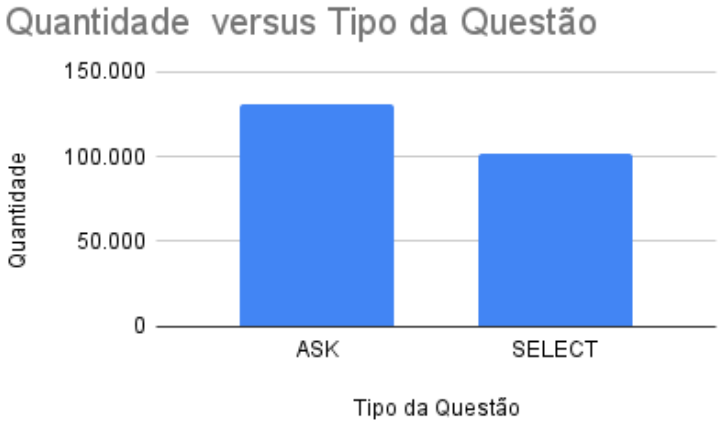
5.1 Conjunto de Dados *ExQuestions*

Inicialmente, detalhamos as contribuições do *framework* nesta tese. Trata-se de um volumoso conjunto de dados para KBQA, com questões parafraseadas a partir de uma base de conhecimento de senso comum. A seguir, detalhamos o resultado obtido.

5.2 Análise Estatística do *ExQuestions*

ExQuestions possui cerca de 128.578 pares de consultas SPARQL únicas, construídas a partir dos algoritmos propostos nesta tese. Desses pares de consultas, temos 104.791 correspondendo às questões geradas a partir da expansão promovida com o auxílio da base de senso comum — isso significa que 81,49% do volume total de questões advém de expansões. O conjunto de dados possui 7.350 entidades geradas, sendo 4.579 do tipo sujeito e 2.771 do tipo objeto. Os predicados contidos no conjunto de dados são 83 e o número de *templates* gerados é 25.831. As questões de linguagem natural geradas utilizando a estratégia proposta nesta tese atingiram um total de 233.369.

Figura 16 – Distribuição de SPARQL gerado pelo Algoritmo proposto a nossa Tese.



Fonte: Elaborado pelo autor (2021).

Em relação à variedade de questões geradas, é importante salientar que o conjunto

de dados *ExQuestions* contém apenas questões factuais, as quais podem ser do tipo ***ASK*** ou ***SELECT***. Questões do tipo ***ASK*** também são conhecidas como questões booleanas. Temos a seguinte distribuição: 131.199 questões do tipo ***ASK*** e 102.170 do tipo ***SELECT***. Na Figura 16, apresentamos a distribuição dos tipos de questões. As questões classificadas como ***SELECT*** estão divididas da seguinte forma: 42.053 do tipo ***Quem***; 40.640 do tipo ***Onde***; 19.424 do tipo ***Quando***; e 53 são do tipo ***Por que***. Na Figura 17, mostramos a distribuição entre as questões do tipo ***SELECT*** geradas.

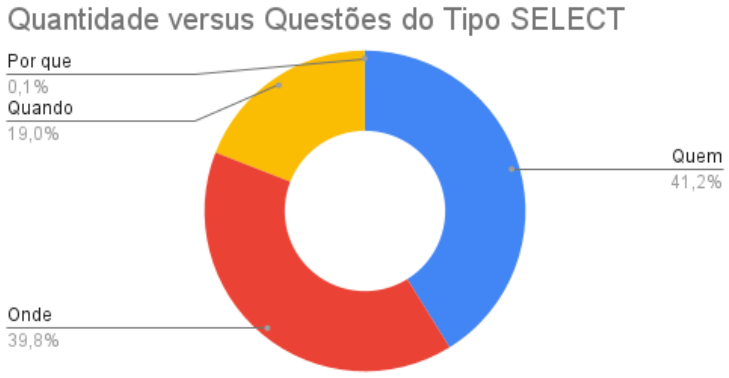
5.3 Caracterização do *ExQuestions*

A Tabela 9 descreve um exemplar de amostra fornecido no conjunto de dados *ExQuestions*. Em primeiro lugar, essa amostra contém a sentença, escrita da mesma forma que foi extraída do *corpus*. Logo após, temos a nova versão da frase após aplicar a estratégia proposta a partir de *OpenIE*. Em seguida, a questão Linguagem Natural (vide 3ª linha) gerada por nosso *framework*, lembrando que essa questão pode ser parafraseada usando uma base de senso comum. Depois, temos as questões com os tipos referentes às suas entidades, bem como um campo com sujeito, predicado e objeto. Adiante, as questões geradas com o *type*, além de sua construção em SPARQL. Por fim, temos o resultado da consulta.

A seguir, são listados alguns exemplos presentes no conjunto de dados, os quais foram gerados a partir da expansão:

- **Ask**: *"Does university of North Carolina at Chapel Hill help the Raleigh region?"*
- **Who**: *"Who did University of North Carolina at Chapel Hill have helped?"*
- **Where:** *"Where did University of North Carolina at Chapel Hill have helped?"*
- **When**: *"When was Nikos Kazantzakis born?"*

Figura 17 – Distribuição de questões do tipo SELECT.



Fonte: Elaborado pelo autor (2021).

Tabela 9 – Exemplo do conjunto de dados - ExQuestions

Sentença Original	North Carolina State, Duke University, and University of North Carolina at Chapel Hill, have helped the Raleigh region (Research Triangle) attract an educated workforce and develop more jobs.
Sentença usando OpenIE	University of North Carolina at Chapel Hill have helped the Raleigh region
Questão em LN	Does university of North Carolina at Chapel Hill help the Raleigh region?
Questão em Type	Does university of <http://dbpedia.org/ontology/PopulatedPlace>at Chapel Hill help the <http://dbpedia.org/ontology/City>region?
Sujeito	http://dbpedia.org/resource/North_Carolina
Predicado	http://dbpedia.org/ontology/capital
Objeto	http://dbpedia.org/resource/Raleigh,_North_Carolina
Type do Sujeito	http://dbpedia.org/ontology/PopulatedPlace
Type do Objeto	http://dbpedia.org/ontology/City
SPARQL	ASK{ ?populatedplace_sub <http://dbpedia.org/ontology/capital> ?city_obj. ?populatedplace_sub rdf:type <http://dbpedia.org/ontology/PopulatedPlace>. ?city_obj rdf:type <http://dbpedia.org/ontology/City>.}
Resposta	Yes

– **Why**: *"Why did the emergence of disco stop Brown's success on the R&B charts?"*

Vale a pena ressaltar, que nosso conjunto de dados tem alguns problemas na estrutura gramatical das questões geradas. Como podemos observar no exemplo *"Who did University of North Carolina at Chapel Hill have helped?"*, entretanto, isso não inviabiliza o seu uso na nossa técnica devido a presença de palavras-chaves que auxiliam na realização do *matching*.

5.4 Disponibilidade e Evolução do *ExQuestions*

O conjunto de dados *ExQuestions* está disponível em um repositório aberto na plataforma GitHub [1] sob os termos da licença *Attribution 4.0 International (CC BY 4.0)* [2]. Como um URL permanente, também fornecemos nosso conjunto de dados através da plataforma Zedono[3].

O *ExQuestions* é distribuído como um despejo de dados (*dump*) no formato ***JSON*** do conjunto de dados KBQA, sendo que esses dados estão divididos em conjunto de treinamento

[1] https://github.com/jwellingtonfranco/ExQuestions
[2] https://creativecommons.org/licenses/by/4.0/
[3] https://zenodo.org/record/4947611#.YMeo6KhKjIU

e conjunto de teste.

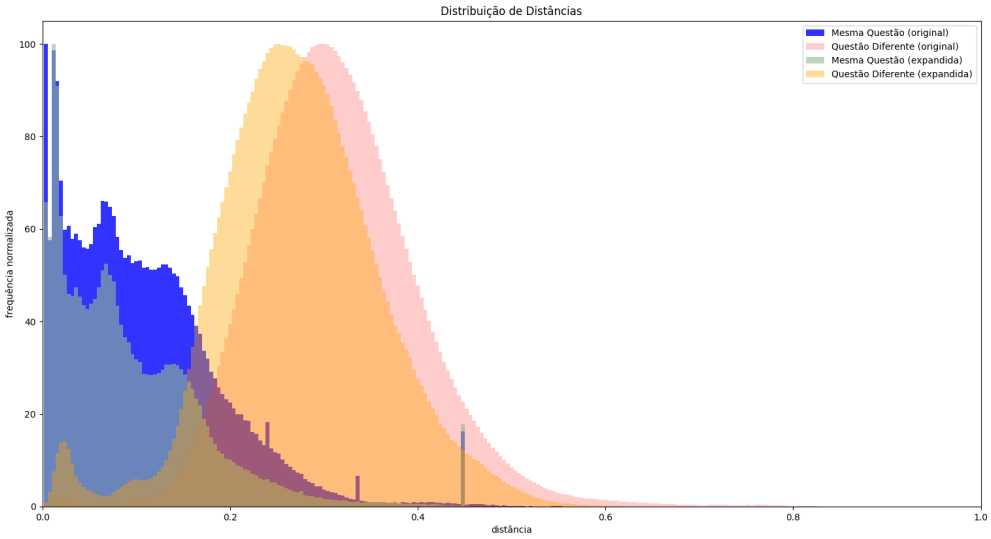
Em relação à evolução do *ExQuestions* como recurso de pesquisa, esse é um aspecto garantido pela equipe do grupo de pesquisa ARIDA da Universidade Federal do Ceará, que se encontra bem sedimentado academicamente e do qual o autor desta tese é membro. Observe que *ExQuestions* é um conjunto de dados crucial para o trabalho de várias pesquisas em andamento, incluindo aquelas desenvolvidas no ARIDA, além desta tese de doutorado. Portanto, espera-se que esse recurso seja mantido e atualizado com regularidade. Planejamos ter um ciclo de lançamento de seis meses, atualizando periodicamente o conjunto de dados com base em sugestões de melhorias e correções apontadas pela comunidade acadêmica.

6 EXPERIMENTOS

Neste capítulo, descrevemos a metodologia utilizada nos experimentos, bem como os resultados encontrados, estes divididos em duas partes: na primeira parte, validamos nosso conjunto de dados gerados (*ExQuestions*) e nosso método de expansão com base em conhecimento de senso comum; na segunda parte, descrevemos a metodologia usada para validar o nosso protótipo de sistema de QA baseado em *templates*.

6.1 Experimentos Relacionados ao *ExQuestions*

Figura 18 – Distribuições de distância entre questões considerando questões originais e expandidas do conjunto de dados *ExQuestions*.



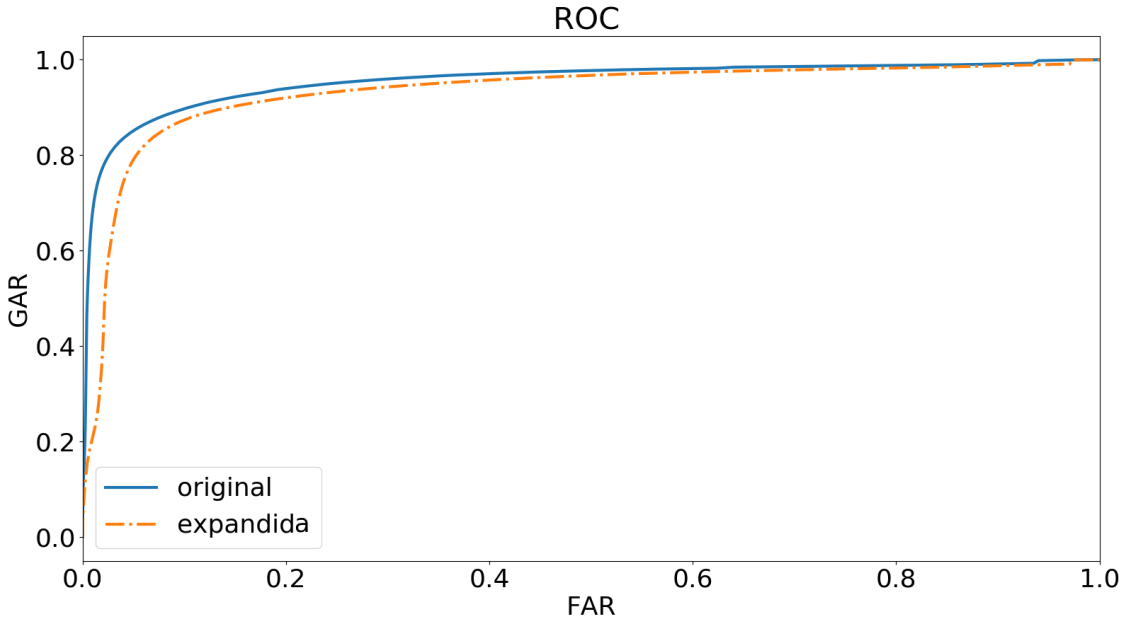
Fonte: Elaborado pelo autor (2021).

Inicialmente, avaliamos as questões geradas pelo nosso *framework* de forma automática. Para tanto, escolhemos comparar todas as questões expandidas com todas as outras não expandidas usando um modelo de *embedding* de texto pronto para uso, o *fastText* (MIKOLOV *et al.*, 2018). Adotamos o *fastText* por ele já ter um modelo pré-treinado com o *corpus* da Wikipédia.

A partir dessa avaliação, identificamos a necessidade de produzir os *embeddings* para todas as questões e todos os modelos no conjunto de dados usando um modelo pré-treinado. Em seguida, produzimos resultados sobre o quão bem essas questões são representadas e discernidas. Como esperado, os experimentos preliminares confirmaram que o modelo *fastText*

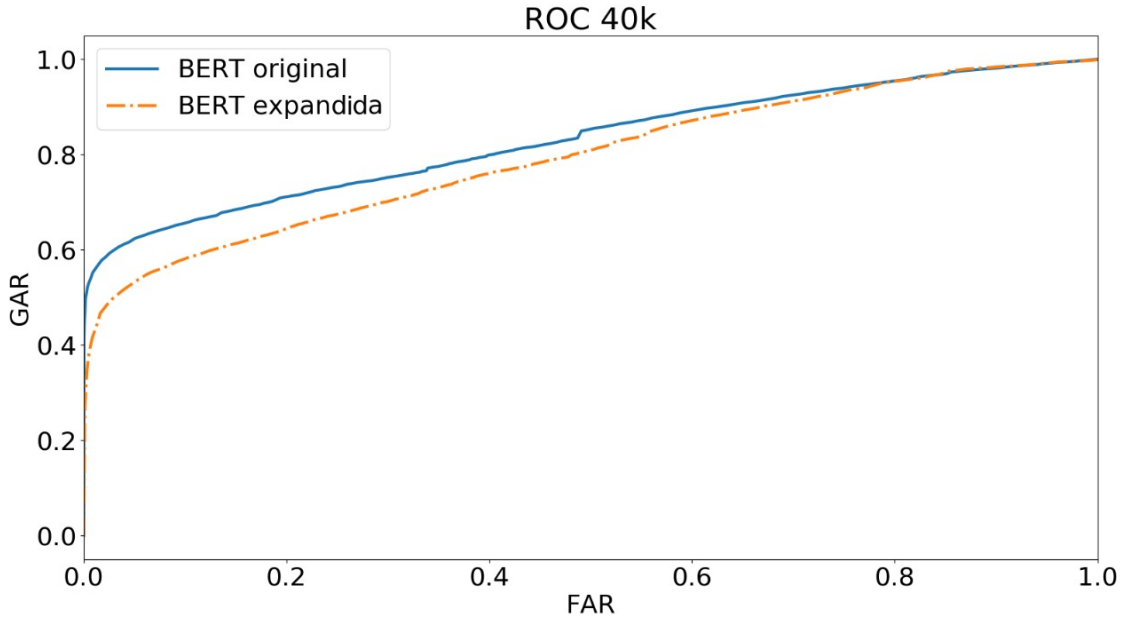
pré-treinado não funcionaria bem para modelos em SPARQL. Note-se que existe um desequilíbrio considerável entre os casos positivos e negativos, pois a maioria dos pares de *embeddings* são formados por modelos associados a diferentes questões. Portanto, consideramos mais adequado

Figura 19 – Curvas ROC usando um modelo *fastText* (MIKOLOV *et al.*, 2018) pré-treinado sobre as questões originais e expandidas presentes no conjunto de dados *ExQuestions*.



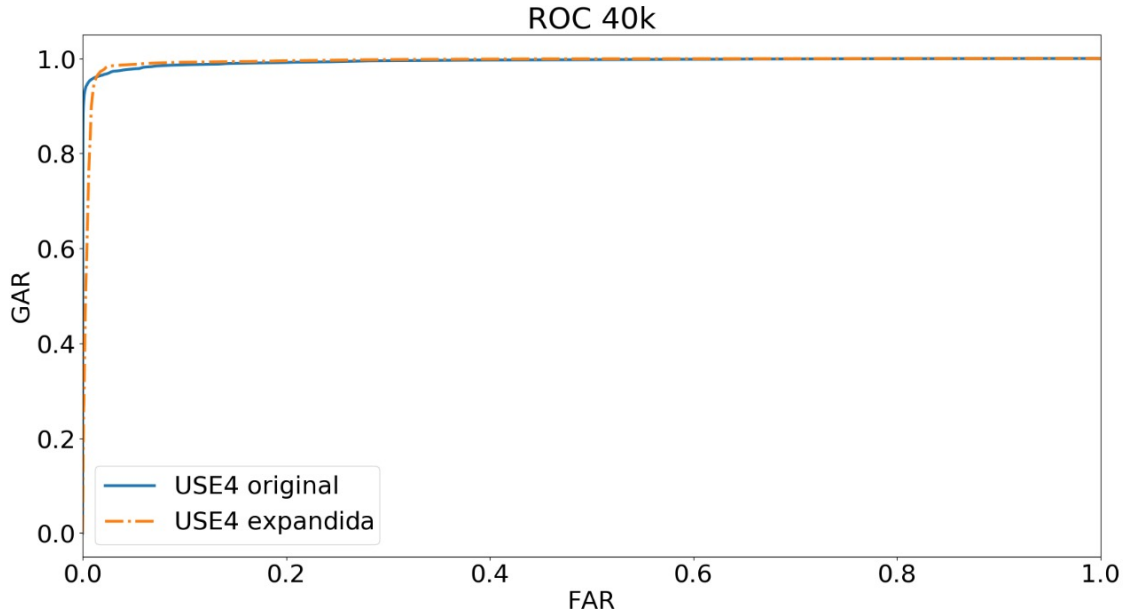
Fonte: Elaborado pelo autor (2021).

Figura 20 – Curvas ROC usando um modelo *BERT* (REIMERS; GUREVYCH, 2019) pré-treinado sobre as questões originais e expandidas presentes no conjunto de dados *ExQuestions*.



Fonte: Elaborado pelo autor (2021).

Figura 21 – Curvas ROC usando um modelo *USE4* (CER *et al.*, 2018) pré-treinado sobre as questões originais e expandidas presentes no conjunto de dados *ExQuestions*.



Fonte: Elaborado pelo autor (2021).

descartarmos métricas convencionais, como a de precisão, cujos resultados tenderiam a expressar apenas a capacidade do modelo de descartar pares negativos, o que poderia enviesar a análise subsequente.

Dito isso, com o objetivo de avaliar como o uso de uma base de conhecimento de senso comum no processo de expansão afeta a semelhança entre questões equivalentes, avaliamos as curvas de distribuição de distâncias considerando questões *"originais"* e questões expandidas. Isso nos permitiu observar como o *fastText* afeta cada um desses dois tipos de questão. Existem centenas de milhares de perguntas no conjunto de dados, por isso é impraticável calcular uma matriz de distância — há limitações de espaço de memória, que seria ocupada com mais de 1 TB de dados.

Consequentemente, essa limitação foi contornada calculando um histograma com resolução de 500 intervalos homogêneos para ambos os casos. Esse processo foi executado em 32 e 12 horas para os conjuntos de dados originais e expandidos, respectivamente. Como pode ser visto na Figura 18, as questões originais têm um desempenho um pouco melhor do que as expandidas, uma vez que as curvas positivas e negativas estão um pouco melhor separadas. Essas curvas são mais próximas para questões expandidas, além de ligeiramente deslocadas para a esquerda, ou seja, exibem distâncias menores para o caso negativo.

Recorremos à curva ROC (*Receiver Operating Characteristic*) para descrever de forma mais detalhada o desempenho do modelo para a classificação das questões originais

e expandidas. Trata-se de uma ferramenta amplamente utilizada na pesquisa em medicina (MCCLISH, 1989) e que descreve a variação da acurácia do modelo (em termos de *Genuine Acceptance Rate*, taxa de aceitação de positivos genuínos) em função do erro admitido (em termos de *False Acceptance Rate*, taxa de aceitação entre questões que deveriam ser tidas como distintas). Note-se que a curva ROC é necessariamente monótona, i.e., o valor de GAR cresce à medida que o valor de FAR aumenta como efeito do classificador se tornar menos criterioso para admitir que duas questões são equivalentes. Assim, a curva ROC é construída mudando o limiar de distância (*threshold*) usado para considerar que dois *embeddings* estão representando a mesma questão, partindo do zero, onde o ponto de operação do modelo não admite erro de classificação algum e o valor de GAR é o menor possível, até um valor que cause com que todos pares possíveis de questões sejam considerados idênticos (logo GAR e FAR são máximos, i.e., iguais a 1). Por fim, vale salientar que a curva ROC ideal apresenta $GAR = 1$ quando $FAR = 0$.

Conforme mostrado na Figura 19, esses resultados são expressivos por terem sido obtidos por um modelo *fastText* pré-treinado, mesmo que tenha sido treinado sobre o mesmo *corpus*. Constatando que as curvas são suficientemente próximas, seria razoável admitir que o comportamento do desempenho do modelo é semelhante nos casos de questões originais e expandidas.

Considerando um erro abaixo de $0,05$ em termos de Taxa de Falsa Aceitação (FAR), tal abordagem tem um desempenho semelhante para as perguntas originais e expandidas, apresentando Taxas de Aceitação Genuína (GAR) de $0,8524$ e $0,7961$, respectivamente. A diferença de GAR é de apenas de $5,6\%$. Consequentemente, nossos resultados experimentais fornecem (1) um resultado inicial usando vetores de *embbedings* para representação de questões; e (2) evidência de que nosso método de expansão produz distribuições e desempenhos semelhantes para as questões originais e expandidas.

Nós também decidimos abordar modelos de linguagem que usam aprendizado profundo para gerar vetores de *embeddings*. Recorremos ao *Universal Frase Encoder, versão 4* (USE4) (CER *et al.*, 2018) e *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN *et al.*, 2019) pelos excelentes resultados que esses modelos forneceram para outras aplicações (TENNEY *et al.*, 2019).

6.1.1 Comparação entre Técnicas de Representação

Curiosamente, como mostrado nas Figuras 19 e 20, o BERT não pode fornecer resultados aceitáveis mesmo quando comparado a uma abordagem mais simples como *fastText*. Apesar de o modelo BERT ter sido treinado sobre o mesmo *corpus*, ele apresentou desempenho muito ruim na captura de nuances semânticas encontradas no texto. Nossa hipótese é de que essa limitação surge da complexidade encontrada na maioria das sentenças em nosso conjunto de dados - com base no fato de que muitas sentenças são mapeadas em um pequeno subconjunto do espaço vetorial, ou seja, apenas em alguns *clusters*, devido à estratégia simplista de representação de sentenças usada no modelo. Acreditamos que modelos mais sofisticados baseados em BERT (REIMERS; GUREVYCH, 2019) podem superar essa limitação e superar o BERT "simples", que apresentou o pior desempenho.

Para fins de comparação, também geramos histogramas adicionais usando os dados dos histogramas resultantes do modelo USE4. Fizemos três recortes: um mais reduzido, com 2.000 questões (vide Figura 22); um intermediário, com 10.000 questões (vide Figura 23); e outro com 40.000 questões (vide Figura 24). Note-se que as curvas sugerem que há uma boa similaridade entre as questões positivas (mesma questão, porém incluindo expansões) e separabilidade entre questões negativas (distintas) nessas amostras de dados.

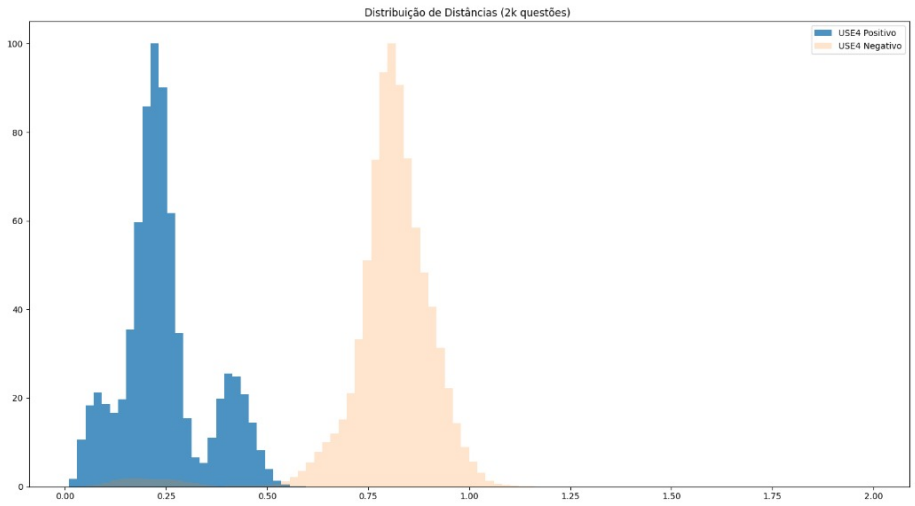
Por fim, conforme mostrado nas Figuras 19, 20 e 21, USE4 apresentou os melhores resultados, superando BERT e *fastText* por uma margem significativa. Considerando FAR abaixo de $0,05$, USE4 tem um desempenho notavelmente bom tanto para questões originais quanto expandidas, apresentando valores $0,981$ e $0,988$ para GAR, respectivamente.

As curvas ROC para as questões originais e as parafraseadas são muito semelhantes e se cruzam em aproximadamente GAR = $0,96$ quando FAR é cerca de $0,01$ (consultar Figura 21). Trata-se de um resultado muito promissor considerando que o modelo USE4 é usado sem ser retreinado com o nosso *corpus*. Isso, em conjunto com o bom desempenho também mostrado pelo modelo *fastText*, reforça a hipótese de que nossa estratégia de paráfrase beneficia responder perguntas sobre bases de conhecimento à luz de como modelos de representação de frases do estado-da-arte podem ser usados para recuperar respostas de modelos de perguntas com base em uma pergunta escrita em linguagem natural.

6.1.2 Ameaças à validade

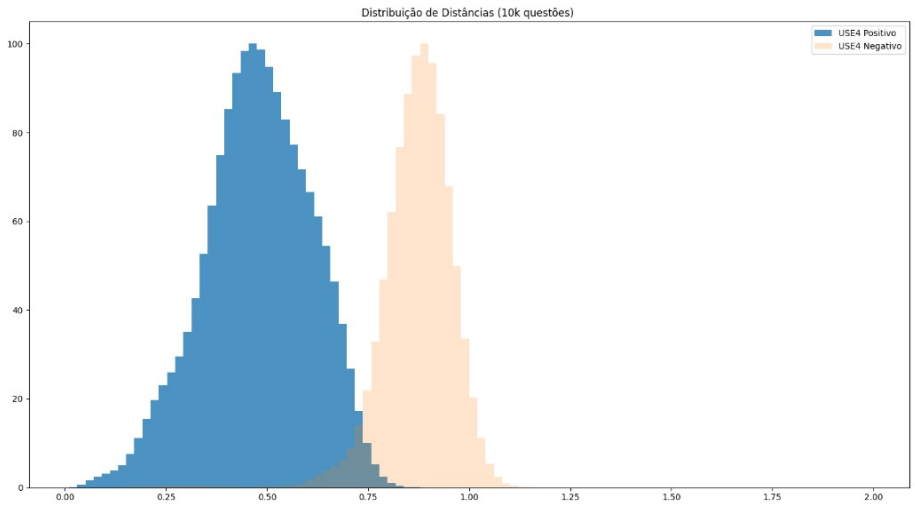
Mesmo trabalhando com uma metodologia rígida, identificamos algumas ameaças à validade durante a construção do nosso conjunto de dados. Acreditamos que *ExQuestions* ainda é relevante nesse contexto. Além disso, o conjunto de dados irá evoluir conforme novas versões forem lançadas. Eis as principais ameaças à validade:

Figura 22 – Distribuições de distância entre questões considerando o USE4 para 2.000 questões.



Fonte: Elaborado pelo autor (2022).

Figura 23 – Distribuições de distância entre questões considerando o USE4 para 10.000 questões.



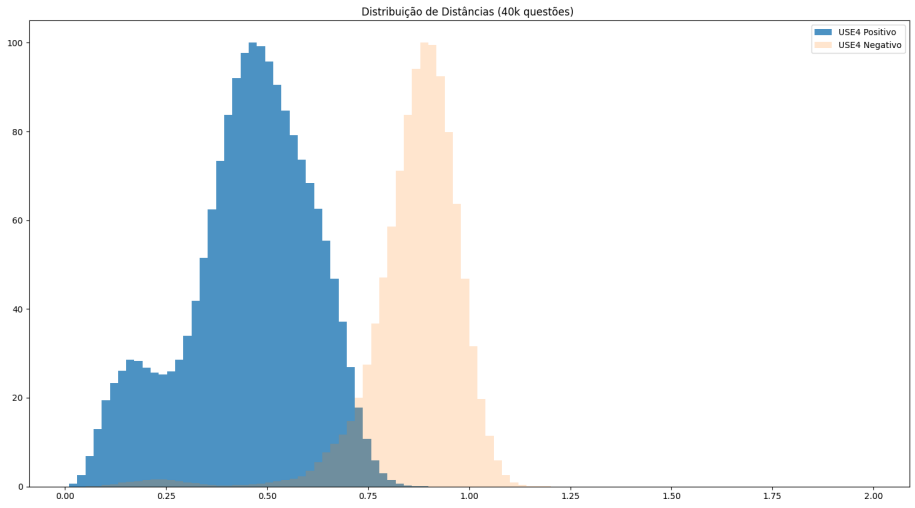
Fonte: Elaborado pelo autor (2022).

– Sentenças redundantes podem ocorrer durante a geração da pergunta, produzindo como resultado sentenças com pequenas diferenças em suas estruturas. No entanto, isso não é necessariamente uma limitação. Os sistemas de QA devem capturar melhor as diferenças de como os usuários colocam suas perguntas;
– Existe uma possível baixa variedade em diferentes tipos de pergunta - isso ocorre, por exemplo, com algumas perguntas *Who*. Assim, percebemos a necessidade de atualizar o algoritmo de geração de questões em linguagem natural para produzir resultados mais balanceados, com maior diversidade nos tipos de pergunta gerados; e
– Nosso método depende muito do KG e da base de conhecimento de senso comum. Consequentemente, a qualidade do conjunto de dados gerado provavelmente diminuirá quando o KG ou a base de conhecimento de senso comum contiverem inconsistências, vieses ou falhas. Não abordamos esse problema porque ele está além do escopo do trabalho.

6.2 Experimentos Relacionados ao QALD-9

Como forma de validação adicional do nosso algoritmo de expansão, aplicamo-lo sobre as questões que estão presentes no QALD-9. Isso serve para ilustrar que o método de expansão proposto é viável e exequível sobre esse conjunto de dados. É importante notar que essa edição do QALD disponibilizou um conjunto de questões limitado a apenas 150 amostras, sendo

Figura 24 – Distribuições de distância entre questões considerando o USE4 para 40.000 questões.

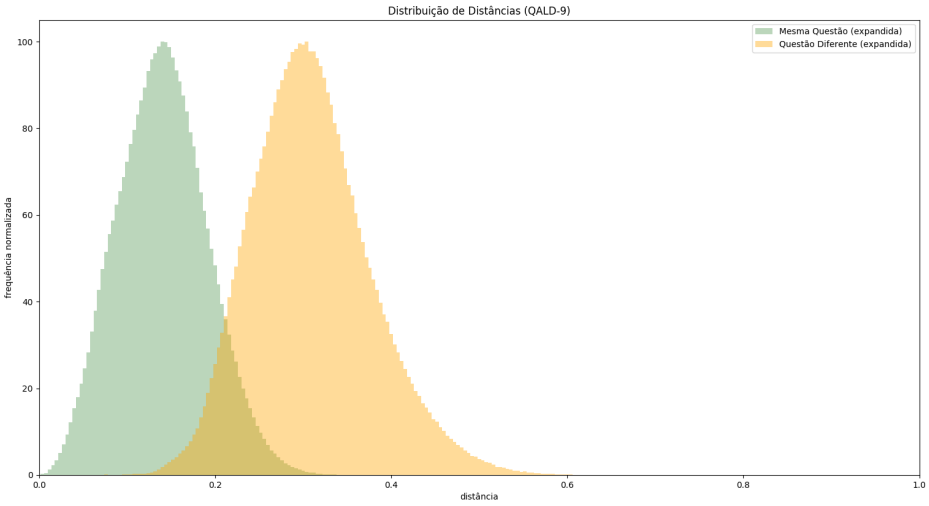


Fonte: Elaborado pelo autor (2022).

cada questão única. Isso posto, trata-se de um cenário típico para aplicação de um algoritmo de expansão que vise aprimorar os resultados alcançáveis por um sistema de QA.

Tal como no experimento anterior, calculamos um histograma para ambos os casos. Como pode ser visto na Figura 25, as questões originais têm um desempenho um pouco melhor do que as expandidas. Essas curvas são mais próximas para questões expandidas, mostrando que a questão expandida e a questão original estão bem próximas. Assim, consideramos razoável assumir que o resultado do experimento contribui para reforçar a hipótese de que um número maior de questões diferentes é obtido por meio do nosso método de expansão, de modo que as questões expandidas pertencem ao mesmo campo semântico do significado das questões originais.

Figura 25 – Distribuições de distância entre questões do conjunto de dados QALD-9. Note-se que isso só é possível devido à expansão, pois cada questão do QALD-9 é única.



Fonte: Elaborado pelo autor (2021).

6.3 Experimentos Relativos ao Sistema de QA

Nossos experimentos relativos ao protótipo de sistema de QA baseado em *templates* se dividiram em duas partes. A primeira diz respeito ao método de indexação utilizado para recuperar os *templates*. Na segunda parte, usaremos para explicar uma avaliação inicial que realizamos para validar o protótipo.

6.3.1 Avaliação do método de indexação dos Templates

Um dos gargalos dos sistemas de QA baseado em *templates* é a recuperação do *template*. Como forma de mitigar esse problema, nós usamos técnicas de busca baseadas em ANN para indexar os *templates* na etapa de QM. Dentre os algoritmos existentes no estado-da-arte escolhemos os seguintes: FLANN (MUJA; LOWE, 2009; MUJA; LOWE, 2014), ANNOY (AUMÜLLER *et al.*, 2020), NGT (IWASAKI; MIYAZAKI, 2018). Escolhemos estes algoritmos para avaliar e definir qual será o método de indexação a ser usado no nosso sistema, devido a eles terem bibliotecas de fácil acesso e manipulação, além de apresentaram bons resultados para outras áreas (MUJA; LOWE, 2009; AUMÜLLER *et al.*, 2020; FRANCO *et al.*, 2020a).

Usamos uma metologia bem simples e efetiva para avaliar os algoritmos de indexação. A ideia é aplicar sobre um conjunto dados textuais no formato de *embeddings* as técnicas de indexação e realizar as consultas usando uma métrica de similaridade entre os vetores. Utilizamos as seguintes similaridade: Cosseno (SOHN, 2001), Euclidiana (DANIELSSON, 1980), Manhattan (KONG *et al.*, 2012) e Hamming (NOROUZI *et al.*, 2012). O conjunto dados textuais usado será o mesmo utilizado em 6.1. Outro fator importante ressaltar é que também vamos utilizar a métrica *TOP-k* para recuperação da resposta. Como mencionado em diversos trabalhos de sistemas de QA (SILVA *et al.*, 2020; ABUALIGAH, 2019; DIEFENBACH *et al.*, 2018).

Com o propósito de prover uma comparação adequada das técnicas de indexação, foram usadas suas implementações na linguagem *Python* e priorizando o uso da parametrização padrão de cada algoritmo tanto na etapa de construção do seu respectivo índice, quanto na etapa de busca. É importante salientar que os resultados da indexação são independentes do limiar escolhido, visto que os vetores reportados pelo índice são aqueles considerados mais próximos de acordo com o critério utilizado no algoritmo de busca. Depois de realizar essa comparação, escolhemos as melhores métricas de similaridade para cada abordagem e comparamos para escolher a melhor técnica. Para o algoritmo NGT, a métrica de dissimilaridade que apresentou os melhores resultados foi a distância do cosseno. Na Tabela 10, temos os resultados da NGT para o *TOP-50*.

No algoritmo ANNOY, à métrica de similaridade que apresentou os melhores resultados foi também à cosseno. Na Tabela 11, temos os resultados da ANNOY para o TOP-50.

No algoritmo *FLANN*, à métrica de similaridade que apresentou os melhores resul-

Tabela 10 – O Top-k referente a NGT utilizando a distância cosseno.

TOP-K	Acurácia
1	0,37
2	0,49
3	0,60
4	0,65
5	0,70
10	0,81
25	0,89
50	0,93

Tabela 11 – O Top-k referente a ANNOY utilizando a distância cosseno.

TOP-K	Acurácia
1	0,63
2	0,71
3	0,76
4	0,79
5	0,83
10	0,89
25	0,94
50	0,98

tados foi a euclidiana. Na Tabela 12, temos os resultados da *FLANN* para o TOP-50. Temos o melhor resultado no algoritmo *FLANN*, devido há esse excelente resultado escolhemos essa técnica para o nosso protótipo de QA.

Tabela 12 – O Top-k referente a *FLANN* utilizando a distância euclidiana.

TOP-K	Acurácia
1	0,68
2	0,78
3	0,83
4	0,86
5	0,88
10	0,94
25	0,98
50	0,99

Como falado anteriormente, a técnica *FLANN* utiliza múltiplas *kd-trees* para lidar com alta dimensionalidade e conjuntos de dados muito grandes e na construção dos índices. Com isso, um parâmetro de entrada importante é a definição do número n de árvores geradas. Para isso geramos testes com $n \in 4, 8, 16, 32, 64$. A Tabelas 13 sumariza os resultados obtidos comparando-se uma única questão original com as demais armazenadas no índice, *i.e.*, todas as centenas de milhares de questões expandidas. Como esperado, o tempo de construção do índice

aumenta com a quantidade de árvores usadas nessa estrutura, podendo chegar até 1 minuto com *FLANN*, no pior caso. Já o tempo de busca não apresenta diferença relevante: cada pesquisa pelos k vizinhos toma menos de 10 milissegundos.

Tabela 13 – Acurácias (*Acc.*) resultantes da pesquisa pelo *TOP-k* referente ao método *FLANN* em que cada questão original foi pesquisada contra todas as questões expandidas. Foram utilizadas 4, 8, 16, 32 e 64 *kd-trees* em cada índice.

TOP-K	**Acc.** n=4	**Acc.** n=8	**Acc.** n=16	**Acc.** n=32	**Acc.** n=64
1	0,61	0,61	0,63	0,64	0,64
2	0,68	0,70	0,71	0,71	0,72
3	0,70	0,72	0,74	0,75	0,75
4	0,71	0,73	0,75	0,76	0,76
5	0,72	0,74	0,76	0,77	0,77
10	0,73	0,75	0,77	0,79	0,79
25	0,74	0,76	0,79	0,80	0,80
50	0,76	0,79	0,81	0,83	0,83

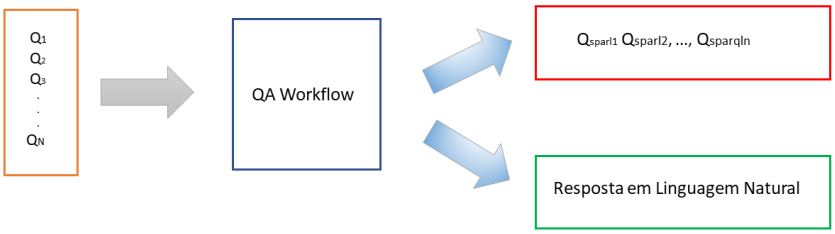
6.3.2 Avaliação inicial do Protótipo de Sistema de QA

Na nossa tese, passamos por todas as etapas de criação de um *framework* para sistema de QA baseado em *templates*. A última parte desse *framework* é protótipo do sistema de QA. Na capítulo 4, explicamos detalhadamente como funciona a etapa *online*.

Para validar essa etapa, desenvolvemos um sistema de QA no formato API, na qual o temos como entrada um conjunto de questões em linguagem natural e como resposta um conjunto de consultas em SPARQL referente a resposta dessa entrada. Na Figura 26 temos um modelo de como funciona a nossa API.

Na Figura 27, apresentamos a execução de um exemplo para a questão *Was chemban Vinod Jose born to Maliackal Chemban Jose and Annies on 24 May 1976, in Angamaly as their old age boy?*.

Figura 26 – Modelo referente ao *workflow* da API.



Fonte: Elaborado pelo autor (2021).

Figura 27 – Exemplo da utilização da API do sistema de QA

```
In [2]: model = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
        fn2 = "indexDasQuestoesLN.csv.USE4"
        db_path = "baseDeTemplates.db"

In [3]: Q = "Was chemban Vinod Jose born to Maliackal Chemban Jose and Annies on 24 May 1976 , in Angamaly as their old age boy?"
        questoes = [Q]
        respostas = workflow(model, fn2, db_path, questoes)

In [4]:
sparql = SPARQLWrapper("https://dbpedia.org/sparql")
sparql.setReturnFormat(JSON)
for r in respostas:
    print("Questão: ", r['questao'])
    for q in r['questoesSPARQL']:
        print(f"----------{q['template_id']}--------------")
        print(q['SPARQL'])
        sparql.setQuery(q['SPARQL'])
        sparqlResult = sparql.queryAndConvert()
        print("Respostas: ")
        if 'boolean' in sparqlResult:
            print(sparqlResult["boolean"])
        elif 'results' in sparqlResult:
            print(sparqlResult["results"])
        else:
            print("...")

Questão:  Was chemban Vinod Jose born to Maliackal Chemban Jose and Annies on 24 May 1976 , in Angamaly as their old age boy?
----------788847--------------
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

ASK{
        <http://dbpedia.org/resource/Chemban_Vinod_Jose> <http://dbpedia.org/ontology/residence> <http://dbpedia.org/resource/A
ngamaly>.
<http://dbpedia.org/resource/Chemban_Vinod_Jose> rdf:type <http://dbpedia.org/ontology/Person>.
<http://dbpedia.org/resource/Angamaly> rdf:type <http://dbpedia.org/ontology/Place>.

}

Respostas:
False
```

Fonte: Elaborado pelo autor (2021).

7 CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho, propusemos uma estratégia completa para criação de *templates* para QA sobre KBs de forma eficiente e com qualidade, de modo que possa ser usada em virtualmente qualquer sistema de QA que adote a abordagem baseada em *templates*. Nossa estratégia tem como diferenciais: o uso do *Open IE* para seleção do conjunto inicial das triplas; o uso de base de senso comum para aumentar a variabilidade das questões em linguagem natural; a criação de conjunto de dados para KBQA; e a metodologia de avaliação dos resultados. Buscamos assim traçar diretrizes acerca desses problemas, onde nossos experimentos e análises trouxeram novos conhecimentos e forneceram insumos para traçar hipóteses sobre as questões de pesquisa propostas na tese:

QP1 O uso do conhecimento de senso comum pode aumentar a variabilidade das questões de um TBQA geradas com o uso de técnicas de paráfrase?

Foram realizados diversos experimentos com estratégias diferentes para gerar *embeddings* das sentenças. Concluímos que houve um crescimento na quantidade de questões sem comprometer o sentido das questões, i.e., o uso de uma base de conhecimento de senso comum no nosso método de paráfrase melhorou a variabilidade das questões geradas;

QP2 É possível extrair automaticamente, a partir de texto, relações úteis para responder questões factuais usando a semântica de um KG?

Graças ao uso do Open IE na nossa estrategia, houve uma melhora significativa na qualidade das relações extraídas, isso só foi possível devido ao uso da semântica presente no KG que por consequência garantiu um melhor qualidade na construção do nossos *templates*.

QP3 Como enriquecer semanticamente um *corpus* de questões associadas aos *templates* de um TBQA?

Com o uso de conhecimento de senso comum conseguimos enriquecer as questões associadas aos *templates*, como mostrado nos experimentos, e consequentemente melhorar a variabilidade das questões geradas e ter resultados melhores na etapa de *matching* entre a entrada e a pergunta associada ao *template*; e

QP4 Como representar e recuperar, de forma eficiente, os *templates* mais adequados para responder uma questão a partir de um grande volume de *templates*?

Em nosso sistema de QA, desenvolvemos um algoritmo de indexação baseado em ANN

para busca do *template*. A vantagem dessa abordagem é que sempre conseguimos produzir uma resposta. Os testes realizados demostraram que o método de busca é muito eficiente e fornecem um bom *baseline* para evoluir com trabalhos futuros.

7.1 Contribuições do Trabalho

Além das questões de pesquisa, listamos a seguir as principais contribuições da nossa tese de doutorado:

- A realização de uma ampla revisão bibliográfica;
- O projeto e desenvolvimento um *framework* completo para sistema de QA baseado em TBQA sobre KBQA;
- A criação e validação do *ExQuestions*. Até onde sabemos, é o primeiro conjunto de dados factual gerado com auxílio de uma base de conhecimento de senso comum para KBQA;
- A proposta de algoritmo de indexação baseado em ANN baseado na similaridade entre as perguntas em linguagem natural; e
- Um sistema QA baseado em TBQA sobre KBQA.

7.2 Publicações

Nesta seção, detalhamos os principais trabalhos publicados e as submissões realizadas durante o doutoramento. Dividimo-la em duas partes: a primeira diz respeito aos trabalhos que têm relação direta com as contribuições presentes na tese; a segunda parte já elenca os trabalhos desenvolvidos de forma correlata com o tema da tese.

7.2.1 Publicações e Submissões da Tese

Na Tabela 14, estão dispostas as publicações já ocorridas a partir da tese. Esses trabalhos são relativos à atividade de revisão bibliográfica.

Tabela 14 – Publicações decorrentes da tese.

Tipo do Trabalho	**Referência**	**Qualis**	**Status**
Periódico	(SILVA *et al.*, 2020)	A2	Publicado
Conferência	(FRANCO *et al.*, 2020b)	B2	Publicado
Conferência	(FRANCO *et al.*, 2022)	B1	Publicado
Periódico	(FRANCO *et al.*,)	A1	Submetido

São inicialmente quatro trabalhos em conferências e periódicos de bastante relevância. Vale a pena ressaltar que até a defesa da tese, esperamos ter a resposta da submissão do trabalho (FRANCO *et al.*,), para o periódico *Language Resources and Evaluation (LREV)*.

7.2.2 Publicações e Submissões Periféricas

Na Tabela 15, constam as publicações periféricas realizadas durante o período do doutorado em que o candidato é co-autor e o tema se relaciona diretamente com os aspectos subjacentes aos estudos realizados no desenvolvimento da tese.

Tabela 15 – Publicações periféricas realizadas durante o doutorado.

Tipo do Trabalho	**Referência**	**Qualis**
Periódico	(SILVA *et al.*, 2021)	B2
Periódico	(AVILA *et al.*, 2021)	B3
Conferência	(MOTA *et al.*, 2021)	B3
Conferência	(SÁ *et al.*, 2021)	B2
Conferência	(AVILA, 2020)	B1
Conferência	(ARRUDA *et al.*, 2019)	B2
Conferência	(AVILA *et al.*, 2019)	B2
Conferência	(AVILA *et al.*, 2019b)	B2
Conferência	(AVILA *et al.*, 2019a)	B1
Conferência	(SALES *et al.*, 2018)	A1
Conferência	(FRANCO *et al.*, 2018a)	B3
Conferência	(AVILA *et al.*, 2018)	B1
Conferência	(FRANCO *et al.*, 2018b)	B2

São treze publicações periféricas em veículos bem sedimentados, o que pode ser atestado pelo Qualis dessas publicações, o qual considera o extrato anterior à mudança prevista para 2022. Por fim, note-se que são publicados quatro artigos por ano nesse período, o que sugere evolução das temáticas no contexto do grupo e também a importância do tema de pesquisa.

7.3 Trabalhos Futuros

Durante o trabalho de construção da tese surgiram diversas possibilidades de pesquisas futuras. Dentro de cada parte do *framework* encontramos uma diversidade de possibilidades. Dentre os trabalhos futuros, elencamos:

– Explorar técnicas de aprendizado de máquina profundo, como GPT-2 (RADFORD *et al.*, 2019), para geração das questões em linguagem natural utilizando conhecimento de senso comum;

- Realizar uma avaliação extrínseca do nosso conjunto de dados, o *ExQuestions*, com um serviço de *crowd-sourcing*;
- Realizar uma avaliação extrínseca no nosso protótipo de sistema de QA, comparando com os principais encontrados na literatura;
- Explorar técnicas linguísticas mais aprofundadas para evoluir o nosso *framework* para questões complexas;
- Explorar o *framework* em outros *corpora* para considerando diferentes idiomas para gerar versões multilíngues do *ExQuestions*;
- Explorar a criação de *embeddings* de grafos, para realização do *matching* diretamente em cima da consulta SPARQL; e
- Por fim, pesquisar e desenvolver técnicas de mitigação da alta dependência de memória no nosso algoritmo de indexação.

REFERÊNCIAS

ABUALIGAH, L. M. Q. **Feature selection and enhanced krill herd algorithm for text document clustering**. [*S. l.*]: Springer, 2019.

ABUJABAL, A.; ROY, R. S.; YAHYA, M.; WEIKUM, G. Comqa: A community-sourced dataset for complex factoid question answering with paraphrase clusters. **arXiv preprint arXiv:1809.09528**, 2018.

ABUJABAL, A.; YAHYA, M.; RIEDEWALD, M.; WEIKUM, G. Automated template generation for question answering over knowledge graphs. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Proceedings of the 26th international conference on world wide web**. [*S. l.*], 2017. p. 1191–1200.

AGHAEBRAHIMIAN, A.; JURČÍČEK, F. Open-domain factoid question answering via knowledge graph search. In: **Proceedings of the Workshop on Human-Computer Question Answering**. [*S. l.*: *s. n.*], 2016. p. 22–28.

ANACLETO, J.; GODOI, M. de Souza; CARVALHO, A. de; LIEBERMAN, H. A common sense-based on-line assistant for training employees. **Human Computer Interaction-INTERACT 2007**, Springer, p. 243–254, 2007.

ARRUDA, N.; ALCÂNTARA, J.; VIDAL, V.; BRAYNER, A.; CASANOVA, M.; PEQUENO, V.; FRANCO, W. A fuzzy approach for data quality assessment of linked datasets. In: SCITEPRESS. **International Conference on Enterprise Information Systems**. [*S. l.*], 2019. v. 1, p. 399–406.

AUMÜLLER, M.; BERNHARDSSON, E.; FAITHFULL, A. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. **Information Systems**, Elsevier, v. 87, p. 101374, 2020.

AVILA, C.; CALIXTO, A.; ROLIM, T.; FRANCO, W.; VENCESLAU, A.; VIDAL, V.; PEQUENO, V.; MOURA, F. Medibot: An ontology based chatbot for portuguese speakers drug's users. In: SCITEPRESS. **International Conference on Enterprise Information Systems**. [*S. l.*], 2019. v. 1, p. 25–36.

AVILA, C. V.; ROLIM, T. V.; CRUZ, M. M. L. da; PIRES, A.; FRANCO, J. W.; VIDAL, V. Um linked data mashup de dados de execuções financeiras e indicadores educacionais no ensino básico. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [*S. l.*: *s. n.*], 2018. v. 29, n. 1, p. 1911.

AVILA, C. V. d. S. **Conquest: um framework para a construção de chatbots de IQA baseados em templates sobre knowledges graphs**. Dissertação (Mestrado) – Departamento de Computação, Universidade Federal do Ceará, 7 2020.

AVILA, C. V. S.; FRANCO, W.; VENCESLAU, A. D.; ROLIM, T. V.; MP, V. Medibot: An ontology-based chatbot to retrieve drug information and compare its prices. 2021.

AVILA, C. V. S.; MAIA, G.; FRANCO, W.; ROLIM, T. V.; FRANCO, A. O.; VIDAL, V. M. Ontoval: A tool for ontology evaluation by domain specialists. 2019.

AVILA, C. V. S.; ROLIM, T. V.; SILVA, J. W. F. da; VIDAL, V. M. P. Medibot: Um chatbot para consulta de riscos e informações sobre medicamentos. In: SBC. **Anais Estendidos do XIX Simpósio Brasileiro de Computação Aplicada à Saúde**. [*S. l.*], 2019. p. 1–6.

BAADER, F.; CALVANESE, D.; MCGUINNESS, D.; PATEL-SCHNEIDER, P.; NARDI, D. **The description logic handbook: Theory, implementation and applications**. [*S. l.*]: Cambridge university press, 2003.

BAKER, C.; FILLMORE, C.; LOWE, J. The Berkeley Framenet Project. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1**. [*S. l.*], 1998. p. 86–90.

BAKHSHI, M.; NEMATBAKHSH, M.; MOHSENZADEH, M.; RAHMANI, A. M. Data-driven construction of sparql queries by approximate question graph alignment in question answering over knowledge graphs. **Expert Systems with Applications**, Elsevier, v. 146, p. 113205, 2020.

BANKO, M.; CAFARELLA, M. J.; SODERLAND, S.; BROADHEAD, M.; ETZIONI, O. Open Information Extraction from the Web. In: **IJCAI**. [*S. l.*: *s. n.*], 2007. v. 7, p. 2670–2676.

BAO, J.; DUAN, N.; YAN, Z.; ZHOU, M.; ZHAO, T. Constraint-based question answering with knowledge graph. In: **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers**. [*S. l.*: *s. n.*], 2016. p. 2503–2514.

BAST, H.; HAUSSMANN, E. More accurate question answering on freebase. In: ACM. **Proceedings of the 24th ACM International on Conference on Information and Knowledge Management**. [*S. l.*], 2015. p. 1431–1440.

BERANT, J.; CHOU, A.; FROSTIG, R.; LIANG, P. Semantic parsing on freebase from question-answer pairs. In: **Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing**. [*S. l.*: *s. n.*], 2013. p. 1533–1544.

BIZER, C.; LEHMANN, J.; KOBILAROV, G.; AUER, S.; BECKER, C.; CYGANIAK, R.; HELLMANN, S. Dbpedia-a crystallization point for the web of data. **Web Semantics: science, services and agents on the world wide web**, Elsevier, v. 7, n. 3, p. 154–165, 2009.

BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **arXiv preprint arXiv:1607.04606**, 2016.

BOLLACKER, K.; EVANS, C.; PARITOSH, P.; STURGE, T.; TAYLOR, J. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM. **Proceedings of the 2008 ACM SIGMOD international conference on Management of data**. [*S. l.*], 2008. p. 1247–1250.

BORDES, A.; CHOPRA, S.; WESTON, J. Question answering with subgraph embeddings. **arXiv preprint arXiv:1406.3676**, 2014.

BORDES, A.; USUNIER, N.; CHOPRA, S.; WESTON, J. Large-scale simple question answering with memory networks. **arXiv preprint arXiv:1506.02075**, 2015.

BUCHHOLZ, S.; DAELEMANS, W. Complex answers: a case study using a www question answering system. **Natural language engineering**, Cambridge University Press, v. 7, n. 4, p. 301–323, 2001.

BUZAABA, H.; AMAGASA, T. A modular approach for efficient simple question answering over knowledge base. In: SPRINGER. **International Conference on Database and Expert Systems Applications**. [*S. l.*], 2019. p. 237–246.

CABRIO, E.; COJAN, J.; APROSIO, A. P.; MAGNINI, B.; LAVELLI, A.; GANDON, F. Qakis: an open domain qa system based on relational patterns. In: **International Semantic Web Conference, ISWC 2012**. [*S. l.*: *s. n.*], 2012.

CAI, Q.; YATES, A. Large-scale semantic parsing via schema matching and lexicon extension. In: **Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [*S. l.*: *s. n.*], 2013. p. 423–433.

CER, D.; YANG, Y.; KONG, S.-y.; HUA, N.; LIMTIACO, N.; JOHN, R. S.; CONSTANT, N.; GUAJARDO-CÉSPEDES, M.; YUAN, S.; TAR, C. *et al.* Universal sentence encoder. **arXiv preprint arXiv:1803.11175**, 2018.

CHRISTENSEN, J.; SODERLAND, S.; ETZIONI, O. An analysis of open information extraction based on semantic role labeling. In: **Proceedings of the sixth international conference on Knowledge capture**. [*S. l.*: *s. n.*], 2011. p. 113–120.

CHRISTMANN, P.; ROY, R. S.; ABUJABAL, A.; SINGH, J.; WEIKUM, G. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In: **Proceedings of the 28th ACM International Conference on Information and Knowledge Management**. [*S. l.*: *s. n.*], 2019. p. 729–738.

CIMIANO, P.; LOPEZ, V.; UNGER, C.; CABRIO, E.; NGOMO, A.-C. N.; WALTER, S. Multilingual question answering over linked data (qald-3): Lab overview. In: SPRINGER. **International Conference of the Cross-Language Evaluation Forum for European Languages**. [*S. l.*], 2013. p. 321–332.

CIMIANO, P.; MINOCK, M. Natural language interfaces: what is the problem?–a data-driven quantitative analysis. In: SPRINGER. **International Conference on Application of Natural Language to Information Systems**. [*S. l.*], 2009. p. 192–206.

COHEN, W. W.; RAVIKUMAR, P.; FIENBERG, S. E. *et al.* A comparison of string distance metrics for name-matching tasks. In: **IIWeb**. [*S. l.*: *s. n.*], 2003. v. 2003, p. 73–78.

COLLINS, A. M.; LOFTUS, E. F. A spreading-activation theory of semantic processing. **Psychological review**, American Psychological Association, v. 82, n. 6, p. 407, 1975.

CUI, W.; XIAO, Y.; WANG, H.; SONG, Y.; HWANG, S.-w.; WANG, W. Kbqa: learning question answering over qa corpora and knowledge bases. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 10, n. 5, p. 565–576, 2017.

DAI, Z.; LI, L.; XU, W. Cfo: Conditional focused neural question answering with large-scale knowledge bases. **arXiv preprint arXiv:1606.01994**, 2016.

DAIBER, J.; JAKOB, M.; HOKAMP, C.; MENDES, P. N. Improving efficiency and accuracy in multilingual entity extraction. In: **Proceedings of the 9th International Conference on Semantic Systems**. [*S. l.*: *s. n.*], 2013. p. 121–124.

DAMLJANOVIC, D.; AGATONOVIC, M.; CUNNINGHAM, H. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In: SPRINGER. **Extended Semantic Web Conference**. [*S. l.*], 2010. p. 106–120.

DANIELSSON, P.-E. Euclidean distance mapping. **Computer Graphics and image processing**, Elsevier, v. 14, n. 3, p. 227–248, 1980.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)**. Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: https://doi.org/10.18653/v1/n19-1423.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DIEFENBACH, D.; LOPEZ, V.; SINGH, K.; MARET, P. Core techniques of question answering systems over knowledge bases: a survey. **Knowledge and Information systems**, Springer, v. 55, n. 3, p. 529–569, 2018.

DIEFENBACH, D.; TANON, T.; SINGH, K.; MARET, P. Question answering benchmarks for wikidata. In: **ISWC 2017**. [*S. l.*: *s. n.*], 2017.

DUBEY, M.; BANERJEE, D.; ABDELKAWI, A.; LEHMANN, J. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In: SPRINGER. **International semantic web conference**. [*S. l.*], 2019. p. 69–78.

DUBEY, M.; DASGUPTA, S.; SHARMA, A.; HÖFFNER, K.; LEHMANN, J. Asknow: A framework for natural language query formalization in sparql. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2016. p. 300–316.

EGONMWAN, E.; CHALI, Y. Transformer and seq2seq model for paraphrase generation. In: **Proceedings of the 3rd Workshop on Neural Generation and Translation**. [*S. l.*: *s. n.*], 2019. p. 249–255.

FADER, A.; ZETTLEMOYER, L.; ETZIONI, O. Paraphrase-driven learning for open question answering. In: **Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [*S. l.*: *s. n.*], 2013. v. 1, p. 1608–1618.

FERRÁNDEZ, O.; IZQUIERDO, R.; FERRÁNDEZ, S.; VICEDO, J. L. Addressing ontology-based question answering with collections of user queries. **Information Processing & Management**, Elsevier, v. 45, n. 2, p. 175–188, 2009.

FRANCO, A. d. O. da R.; SILVA, J. W. F. da; PINHEIRO, V. C. M.; MAIA, J. G. R.; GOMES, F. A. de C.; CASTRO, M. F. de. Analyzing actions in play-by-forum rpg. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. [*S. l.*], 2018. p. 180–190.

FRANCO, A. O.; ROLIM, T. V.; SANTOS, A. M.; SILVA, J. W.; VIDAL, V. M.; GOMES, F. A.; CASTRO, M. F.; MAIA, J. G. An ontology for role playing games. **Proceedings of SBGames**, p. 615–618, 2018.

FRANCO, A. O.; SOARES, F. F.; NETO, A. V. L.; MACÊDO, J. A. de; REGO, P. A.; GOMES, F. A.; MAIA, J. G. Vehicle re-identification by deep feature embedding and approximate nearest neighbors. In: IEEE. **2020 International Joint Conference on Neural Networks (IJCNN)**. [*S. l.*], 2020. p. 1–8.

FRANCO, W.; AVILA, C. V. S.; OLIVEIRA, A.; MAIA, G.; BRAYNER, A.; VIDAL, V. M. P.; CARVALHO, F.; PEQUENO, V. M. Ontology-based question answering systems over knowledge bases: A survey. In: **ICEIS (1)**. [*S. l.*: *s. n.*], 2020. p. 532–539.

FRANCO, W.; FRANCO, A. O.; AVILA, C. V.; CABRAL, L.; MAIA, G.; PINHEIRO, V.; VIDAL, V.; MACHADO, J. Exquestions: An expanded factual corpus for question answering over knowledge graphs. In: IEEE. **2022 IEEE 16th International Conference on Semantic Computing (ICSC)**. [*S. l.*], 2022. p. 235–242.

FRANCO, W.; OLIVEIRA, A.; VIKTOR, C.; MAYRON, M.; MAIA, G.; PINHEIRO; VLáDIA; VIDAL, V. A commonsense-based framework for expanding factual questions over knowledge graphs.

GE, T.; HE, K.; KE, Q.; SUN, J. Optimized product quantization for approximate nearest neighbor search. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [*S. l.*: *s. n.*], 2013. p. 2946–2953.

GOLUB, D.; HE, X. Character-level question answering with attention. **arXiv preprint arXiv:1604.00727**, 2016.

HAARSLEV, V.; MÖLLER, R. Racer system description. In: SPRINGER. **International Joint Conference on Automated Reasoning**. [*S. l.*], 2001. p. 701–705.

HAARSLEV, V.; MÖLLER, R.; WESSEL, M. Querying the semantic web with racer + nrql. In: **Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany**. [*S. l.*: *s. n.*], 2004. v. 24.

HANNUN, A.; CASE, C.; CASPER, J.; CATANZARO, B.; DIAMOS, G.; ELSEN, E.; PRENGER, R.; SATHEESH, S.; SENGUPTA, S.; COATES, A. *et al.* Deep speech: Scaling up end-to-end speech recognition. **arXiv preprint arXiv:1412.5567**, 2014.

HASAN, S. A.; LIU, B.; LIU, J.; QADIR, A.; LEE, K.; DATLA, V.; PRAKASH, A.; FARRI, O. Neural clinical paraphrase generation with attention. In: **Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)**. [*S. l.*: *s. n.*], 2016. p. 42–53.

HAVASI, C.; SPEER, R.; ALONSO, J. ConceptNet 3: A Flexible, Multilingual Semantic Network for Common Sense Knowledge. In: **Recent Advances in Natural Language Processing**. [*S. l.*: *s. n.*], 2007. p. 27–29.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **Proceedings of the IEEE international conference on computer vision**. [*S. l.*: *s. n.*], 2015. p. 1026–1034.

HERZOG, O.; ROLLINGER, C.-R. **Text understanding in LILOG: integrating computational linguistics and artificial intelligence: final report on the IBM Germany LILOG-Project**. [*S. l.*]: Springer, 1991.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

HÖFFNER, K.; WALTER, S.; MARX, E.; USBECK, R.; LEHMANN, J.; NGOMO, A.-C. N. Survey on challenges of question answering in the semantic web. **Semantic Web**, IOS Press, v. 8, n. 6, p. 895–920, 2017.

HOGAN, A.; BLOMQVIST, E.; COCHEZ, M.; D'AMATO, C.; MELO, G. de; GUTIÉRREZ, C.; KIRRANE, S.; GAYO, J. E. L.; NAVIGLI, R.; NEUMAIER, S.; NGOMO, A.-C. N.; POLLERES, A.; RASHID, S. M.; RULA, A.; SCHMELZEISEN, L.; SEQUEDA, J. F.; STAAB, S.; ZIMMERMANN, A. **Knowledge Graphs**. Morgan & Claypool, 2021. (Synthesis Lectures on Data, Semantics, and Knowledge, 22). ISBN 9781636392363. Disponível em: https://kgbook.org/.

HOGAN, A.; HARTH, A.; UMBRICH, J.; KINSELLA, S.; POLLERES, A.; DECKER, S. Searching and browsing linked data with swse: The semantic web search engine. **Web semantics: science, services and agents on the world wide web**, Elsevier, v. 9, n. 4, p. 365–401, 2011.

HUANG, X.; ZHANG, J.; LI, D.; LI, P. Knowledge graph embedding based question answering. In: **Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining**. [*S. l.*: *s. n.*], 2019. p. 105–113.

IWASAKI, M.; MIYAZAKI, D. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. **arXiv preprint arXiv:1810.07355**, 2018.

JAIN, S. Question answering over knowledge base using factual memory networks. In: **Proceedings of the NAACL Student Research Workshop**. [*S. l.*: *s. n.*], 2016. p. 109–115.

JAIN, S.; TANWAR, N.; BHARGAVA, S. **AUTOMATIC QUESTION GENERATION**. [*S. l.*], 2019.

JIANG, K.; WU, D.; JIANG, H. Freebaseqa: a new factoid qa data set matching trivia-style question-answer pairs with freebase. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. [*S. l.*: *s. n.*], 2019. p. 318–323.

JOHNSON, J.; DOUZE, M.; JÉGOU, H. Billion-scale similarity search with gpus. **IEEE Transactions on Big Data**, IEEE, v. 7, n. 3, p. 535–547, 2019.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; DOUZE, M.; JÉGOU, H.; MIKOLOV, T. Fasttext.zip: Compressing text classification models. **arXiv preprint arXiv:1612.03651**, 2016.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; NICKEL, M.; MIKOLOV, T. Fast linear model for knowledge graph embeddings. **arXiv preprint arXiv:1710.10881**, 2017.

JURAFSKY, D. en jh martin.(2009). **Speech and Language Processing: An Introduction to natural**, 2000.

KACUPAJ, E.; BANERJEE, B.; SINGH, K.; LEHMANN, J. Paraqa: A question answering dataset with paraphrase responses for single-turn conversation. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2021. p. 598–613.

KACUPAJ, E.; ZAFAR, H.; LEHMANN, J.; MALESHKOVA, M. Vquanda: Verbalization question answering dataset. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2020. p. 531–547.

KALAIVANI, S.; DURAISWAMY, K. Comparison of question answering systems based on ontology and semantic web in different environment. In: CITESEER. **Journal of Computer Science**. [*S. l.*], 2012.

KAUFMANN, E.; BERNSTEIN, A. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. **Web Semantics: Science, Services and Agents on the World Wide Web**, Elsevier, v. 8, n. 4, p. 377–393, 2010.

KEYSERS, D.; SCHÄRLI, N.; SCALES, N.; BUISMAN, H.; FURRER, D.; KASHUBIN, S.; MOMCHEV, N.; SINOPALNIKOV, D.; STAFINIAK, L.; TIHON, T. *et al.* Measuring compositional generalization: A comprehensive method on realistic data. **arXiv preprint arXiv:1912.09713**, p. 1–38, 2019.

KONG, W.; LI, W.-J.; GUO, M. Manhattan hashing for large-scale image retrieval. In: **Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval**. [*S. l.*: *s. n.*], 2012. p. 45–54.

KORABLINOV, V.; BRASLAVSKI, P. Rubq: a russian dataset for question answering over wikidata. In: SPRINGER. **International Semantic Web Conference**. [*S. l.*], 2020. p. 97–110.

LATIFI, M. **Using natural language processing for question answering in closed and open domains**. Tese (Doutorado) – Universitat Politècnica de Catalunya, Barcelona, 2018.

LATIFI, M.; HONTORIA, H. R.; SÀNCHEZ-MARRÈ, M. Scoqas: A semantic-based closed and open domain question answering system. **Procesamiento del Lenguaje Natural**, Sociedad Española para el Procesamiento del Lenguaje Natural, 2017.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LEHMANN, J.; FURCHE, T.; GRASSO, G.; NGOMO, A.-C. N.; SCHALLHART, C.; SELLERS, A.; UNGER, C.; BÜHMANN, L.; GERBER, D.; HÖFFNER, K. *et al.* Deqa: deep web extraction for question answering. In: SPRINGER. **International Semantic Web Conference**. [*S. l.*], 2012. p. 131–147.

LI, H.; BOLLEGALA, D.; MATSUO, Y.; ISHIZUKA, M. Using Graph Based Method to Improve Bootstrapping Relation Extraction. In: **Computational Linguistics and Intelligent Text Processing**. [*S. l.*]: Springer, 2011. p. 127–138.

LIANG, S.; STOCKINGER, K.; FARIAS, T. M. de; ANISIMOVA, M.; GIL, M. Querying knowledge graphs in natural language. **Journal of big data**, Springer, v. 8, n. 1, p. 1–23, 2021.

LIU, H.; SINGH, P. ConceptNet: A Practical Commonsense Reasoning Toolkit. **BT Technology Journal**, v. 22, n. 4, p. 211–226, 2004. Disponível em: http://publication.wilsonwong.me.

LIU, H.; SINGH, P. Conceptnet—a practical commonsense reasoning tool-kit. **BT technology journal**, Springer, v. 22, n. 4, p. 211–226, 2004.

LOPEZ, V.; UREN, V.; MOTTA, E.; PASIN, M. Aqualog: An ontology-driven question answering system for organizational semantic intranets. **Web Semantics: Science, Services and Agents on the World Wide Web**, Elsevier, v. 5, n. 2, p. 72–105, 2007.

LOPEZ, V.; UREN, V.; SABOU, M.; MOTTA, E. Is question answering fit for the semantic web?: a survey. **Semantic web**, IOS Press, v. 2, n. 2, p. 125–155, 2011.

LU, J.; SUN, X.; LI, B.; BO, L.; ZHANG, T. Beat: Considering question types for bug question answering via templates. **Knowledge-Based Systems**, Elsevier, v. 225, p. 107098, 2021.

LUKOVNIKOV, D.; FISCHER, A.; LEHMANN, J. Pretrained transformers for simple question answering over knowledge graphs. In: SPRINGER. **International Semantic Web Conference**. [*S. l.*], 2019. p. 470–486.

LUKOVNIKOV, D.; FISCHER, A.; LEHMANN, J.; AUER, S. Neural network-based question answering over knowledge graphs on word and character level. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Proceedings of the 26th international conference on World Wide Web**. [*S. l.*], 2017. p. 1211–1220.

MALKOV, Y. A.; YASHUNIN, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 42, n. 4, p. 824–836, 2018.

MANNING, C.; RAGHAVAN, P.; SCHÜTZE, H. Introduction to information retrieval. **Natural Language Engineering**, Cambridge university press, v. 16, n. 1, p. 100–103, 2010.

MARNEFFE, M.-C. D.; MANNING, C. D. The stanford typed dependencies representation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation**. [*S. l.*], 2008. p. 1–8.

MCCLISH, D. K. Analyzing a portion of the roc curve. **Medical decision making**, Sage Publications Sage CA: Thousand Oaks, CA, v. 9, n. 3, p. 190–195, 1989.

MCKEOWN, K. R. Paraphrasing questions using given and new information. **Comput. Linguist.**, MIT Press, Cambridge, MA, USA, v. 9, n. 1, p. 1–10, jan. 1983. ISSN 0891-2017.

MENDES, P. N.; JAKOB, M.; GARCÍA-SILVA, A.; BIZER, C. Dbpedia spotlight: shedding light on the web of documents. In: **Proceedings of the 7th international conference on semantic systems**. [*S. l.*: *s. n.*], 2011. p. 1–8.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIKOLOV, T.; GRAVE, E.; BOJANOWSKI, P.; PUHRSCH, C.; JOULIN, A. Advances in pre-training distributed word representations. In: **Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)**. [*S. l.*: *s. n.*], 2018.

MILLER, G. WordNet: a Lexical Database for English. **Communications of the ACM**, ACM, v. 38, n. 11, p. 39–41, 1995.

MITHUN, S.; KOSSEIM, L.; HAARSLEV, V. Resolving quantifier and number restriction to question owl ontologies. In: IEEE. **Third International Conference on Semantics, Knowledge and Grid (SKG 2007)**. [*S. l.*], 2007. p. 218–223.

MOHAMMED, S.; SHI, P.; LIN, J. Strong baselines for simple question answering over knowledge graphs with and without neural networks. **arXiv preprint arXiv:1712.01969**, 2017.

MOTA, A. A. X.; FRANCO, W.; MATTOS, C. L. C. Detecção de desinformação sobre covid-19 no twitter. In: SBC. **Anais do XIII Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana**. [*S. l.*], 2021. p. 172–181.

MUJA, M.; LOWE, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. **VISAPP (1)**, v. 2, n. 331-340, p. 2, 2009.

MUJA, M.; LOWE, D. G. Scalable nearest neighbor algorithms for high dimensional data. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 36, n. 11, p. 2227–2240, 2014.

NGOMO, N. 9th challenge on question answering over linked data (qald-9). **language**, v. 7, n. 1, 2018.

NIKLAUS, C.; CETTO, M.; FREITAS, A.; HANDSCHUH, S. A survey on open information extraction. **arXiv preprint arXiv:1806.05599**, 2018.

NOROUZI, M.; FLEET, D. J.; SALAKHUTDINOV, R. R. Hamming distance metric learning. In: **Advances in neural information processing systems**. [*S. l.*: *s. n.*], 2012. p. 1061–1069.

PAN, J. Z.; VETERE, G.; GOMEZ-PEREZ, J. M.; WU, H. **Exploiting linked data and knowledge graphs in large organisations**. [*S. l.*]: Springer, 2017.

PETROCHUK, M.; ZETTLEMOYER, L. Simplequestions nearly solved: A new upperbound and baseline approach. **arXiv preprint arXiv:1804.08798**, 2018.

PINHEIRO, V.; PEQUENO, T.; FURTADO, V.; FRANCO, W. InferenceNet.Br: Expression of Inferentialist Semantic Content of the Portuguese Language. In: **PROPOR**. [*S. l.*]: Springer, 2010. (Lecture Notes in Computer Science, v. 6001), p. 90–99. ISBN 978-3-642-12319-1.

PRAKASH, A.; HASAN, S. A.; LEE, K.; DATLA, V.; QADIR, A.; LIU, J.; FARRI, O. Neural paraphrase generation with stacked residual lstm networks. **arXiv preprint arXiv:1610.03098**, 2016.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. *et al.* Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, p. 9, 2019.

RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. Squad: 100,000+ questions for machine comprehension of text. **arXiv preprint arXiv:1606.05250**, 2016.

RAO, Y.; LU, J.; ZHOU, J. Attention-aware deep reinforcement learning for video face recognition. In: **Proceedings of the IEEE International Conference on Computer Vision**. [*S. l.*: *s. n.*], 2017. p. 3931–3940.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. **arXiv preprint arXiv:1908.10084**, 2019.

RUS, V.; CAI, Z.; GRAESSER, A. Question generation: Example of a multi-year evaluation campaign. **Proc WS on the QGSTEC**, 2008.

SÁ, I. C. de; MONTEIRO, J. M.; SILVA, J. W. F. da; MEDEIROS, L. M.; MOURAO, P. J. C.; CUNHA, L. C. C. da. Digital lighthouse: A platform for monitoring public groups in whatsapp. 2021.

SAHA, A.; PAHUJA, V.; KHAPRA, M.; SANKARANARAYANAN, K.; CHANDAR, S. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [*S. l.*: *s. n.*], 2018. v. 32, n. 1.

SAHA, S. *et al.* Open information extraction from conjunctive sentences. In: **Proceedings of the 27th International Conference on Computational Linguistics**. [*S. l.*: *s. n.*], 2018. p. 2288–2299.

SALES, J. E.; BARZEGAR, S.; FRANCO, W.; BERMEITINGER, B.; CUNHA, T.; DAVIS, B.; FREITAS, A.; HANDSCHUH, S. A multilingual test collection for the semantic search of entity categories. In: **Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)**. [*S. l.*: *s. n.*], 2018.

SEABORNE, A.; PRUD'HOMMEAUX, E. Sparql query language for rdf. w3c recommendation. **World Wide Web consortium, January**, v. 15, 2008.

SERBAN, I. V.; GARCÍA-DURÁN, A.; GULCEHRE, C.; AHN, S.; CHANDAR, S.; COURVILLE, A.; BENGIO, Y. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. **arXiv preprint arXiv:1603.06807**, 2016.

SHEKARPOUR, S.; MARX, E.; NGOMO, A.-C. N.; AUER, S. Sina: Semantic interpretation of user queries for question answering on interlinked data. **Journal of Web Semantics**, Elsevier, v. 30, p. 39–51, 2015.

SILVA, J. W. F. da; VENCESLAU, A. D. P.; SALES, J. E.; MAIA, J. G. R.; PINHEIRO, V. C. M.; VIDAL, V. M. P. A short survey on end-to-end simple question answering systems. **Artificial Intelligence Review**, Springer, p. 1–25, 2020.

SILVA, R. N.; MONTEIRO, J. M.; FRANCO, W. Farolkg: Um grafo de conhecimento sobre desinformação em mensagens do whatsapp. In: SBC. **Anais Estendidos do XXXVI Simpósio Brasileiro de Bancos de Dados**. [*S. l.*], 2021. p. 134–140.

SINGH, P.; LIN, T.; MUELLER, E.; LIM, G.; PERKINS, T.; ZHU, W. L. Open mind common sense: Knowledge acquisition from the general public. **On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE**, Springer, p. 1223 – 1237, 2002.

SOHN, M.-W. Distance and cosine measures of niche overlap. **Social Networks**, Elsevier, v. 23, n. 2, p. 141–165, 2001.

SPEER, R.; CHIN, J.; HAVASI, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [*S. l.*: *s. n.*], 2017. v. 31, n. 1.

SPEER, R.; HAVASI, C. Representing general relational knowledge in Conceptnet 5. In: **International Conference on Language Resources and Evaluation (LREC)**. [*S. l.*: *s. n.*], 2012. p. 79–86.

SU, Y.; SUN, H.; SADLER, B.; SRIVATSA, M.; GÜR, I.; YAN, Z.; YAN, X. On generating characteristic-rich question sets for qa evaluation. In: **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing**. [*S. l.*]: Association for Computational Linguistics, 2016. p. 562–572.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: a core of semantic knowledge. In: ACM. **Proceedings of the 16th international conference on World Wide Web**. [*S. l.*], 2007. p. 697–706.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: A large ontology from wikipedia and wordnet. **Web Semantics: Science, Services and Agents on the World Wide Web**, Elsevier, v. 6, n. 3, p. 203–217, 2008.

TABLAN, V.; DAMLJANOVIC, D.; BONTCHEVA, K. A natural language query interface to structured information. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2008. p. 361–375.

TALMOR, A.; BERANT, J. The web as a knowledge-base for answering complex questions. **arXiv preprint arXiv:1803.06643**, 2018.

TENNEY, I.; DAS, D.; PAVLICK, E. Bert rediscovers the classical nlp pipeline. **arXiv preprint arXiv:1905.05950**, 2019.

TRIVEDI, P.; MAHESHWARI, G.; DUBEY, M.; LEHMANN, J. Lc-quad: A corpus for complex question answering over knowledge graphs. In: SPRINGER. **International Semantic Web Conference**. [*S. l.*], 2017. p. 210–218.

TURE, F.; JOJIC, O. No need to pay attention: Simple recurrent neural networks work!(for answering"simple"questions). **arXiv preprint arXiv:1606.05029**, 2017.

UNGER, C.; BÜHMANN, L.; LEHMANN, J.; NGOMO, A.-C. N.; GERBER, D.; CIMIANO, P. Template-based question answering over rdf data. In: **Proceedings of the 21st international conference on World Wide Web**. [*S. l.*: *s. n.*], 2012. p. 639–648.

UNGER, C.; CIMIANO, P. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In: SPRINGER. **International conference on application of natural language to information systems**. [*S. l.*], 2011. p. 153–160.

VETULANI, Z.; MARIANI, J.; KUBIS, M. **Human Language Technology. Challenges for Computer Science and Linguistics**. [*S. l.*]: Springer, 2011.

WANG, C.; XIONG, M.; ZHOU, Q.; YU, Y. Panto: A portable natural language interface to ontologies. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2007. p. 473–487.

WILENSKY, R.; CHIN, D. N.; LURIA, M.; MARTIN, J.; MAYFIELD, J.; WU, D. The berkeley unix consultant project. **Computational Linguistics**, MIT Press, v. 14, n. 4, p. 35–84, 1988.

XU, K.; REDDY, S.; FENG, Y.; HUANG, S.; ZHAO, D. Question answering on freebase via relation extraction and textual evidence. **arXiv preprint arXiv:1603.00957**, 2016.

YAHYA, M.; BERBERICH, K.; ELBASSUONI, S.; RAMANATH, M.; TRESP, V.; WEIKUM, G. Natural language questions for the web of data. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 2012 Joint Conference on Empirical**

Methods in Natural Language Processing and Computational Natural Language Learning. [*S. l.*], 2012. p. 379–390.

YAHYA, M.; BERBERICH, K.; ELBASSUONI, S.; WEIKUM, G. Robust question answering over the web of linked data. In: ACM. **Proceedings of the 22nd ACM international conference on Conference on information & knowledge management**. [*S. l.*], 2013. p. 1107–1116.

YAO, X.; DURME, B. V. Information extraction over structured data: Question answering with freebase. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [*S. l.*: *s. n.*], 2014. v. 1, p. 956–966.

YIH, W.-t.; RICHARDSON, M.; MEEK, C.; CHANG, M.-W.; SUH, J. The value of semantic parse labeling for knowledge base question answering. In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. [*S. l.*: *s. n.*], 2016. p. 201–206.

YIN, W.; YU, M.; XIANG, B.; ZHOU, B.; SCHÜTZE, H. Simple question answering by attentive convolutional neural network. **arXiv preprint arXiv:1606.03391**, 2016.

YOUNG, T.; CAMBRIA, E.; CHATURVEDI, I.; ZHOU, H.; BISWAS, S.; HUANG, M. Augmenting end-to-end dialogue systems with commonsense knowledge. In: **Thirty-Second AAAI Conference on Artificial Intelligence**. [*S. l.*: *s. n.*], 2018.

ZAFAR, H.; NAPOLITANO, G.; LEHMANN, J. Formal query generation for question answering over knowledge bases. In: SPRINGER. **European semantic web conference**. [*S. l.*], 2018. p. 714–728.

ZHENG, W.; CHENG, H.; YU, J. X.; ZOU, L.; ZHAO, K. Interactive natural language question answering over knowledge graphs. **Information Sciences**, Elsevier, v. 481, p. 141–159, 2019.

ZHENG, W.; YU, J. X.; ZOU, L.; CHENG, H. Question answering over knowledge graphs: question understanding via template decomposition. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 11, n. 11, p. 1373–1386, 2018.

ZHENG, W.; ZOU, L.; LIAN, X.; YU, J. X.; SONG, S.; ZHAO, D. How to build templates for rdf question/answering: An uncertain graph similarity join approach. In: ACM. **Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data**. [*S. l.*], 2015. p. 1809–1824.

ZHU, S.; CHENG, X.; SU, S.; LANG, S. Knowledge-based question answering by jointly generating, copying and paraphrasing. In: ACM. **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. [*S. l.*], 2017. p. 2439–2442.

ZOU, L.; HUANG, R.; WANG, H.; YU, J. X.; HE, W.; ZHAO, D. Natural language question answering over rdf: a graph data driven approach. In: ACM. **Proceedings of the 2014 ACM SIGMOD international conference on Management of data**. [*S. l.*], 2014. p. 313–324.