UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CAIO VIKTOR DA SILVA AVILA

CONQUEST: UM FRAMEWORK PARA A CONSTRUÇÃO DE CHATBOTS DE IQA BASEADOS EM TEMPLATES SOBRE KNOWLEDGE GRAPHS

FORTALEZA

2020

CAIO VIKTOR DA SILVA AVILA

CONQUEST: UM FRAMEWORK PARA A CONSTRUÇÃO DE CHATBOTS DE IQA BASEADOS EM TEMPLATES SOBRE KNOWLEDGE GRAPHS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do 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 mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientadora: Prof.ª Dr.ª Vânia Maria Ponte Vidal

FORTALEZA

2020

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)

A972c Avila, Caio Viktor da Silva.
Conquest: um framework para a construção de chatbots de IQA baseados em templates sobre knowledges graphs / Caio Viktor da Silva Avila. – 2020.
113 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2020.
Orientação: Profa. Dra. Vânia Maria Ponte Vidal.

1. Interactive Question Answering. 2. ChatBot. 3. Knowledge Graph. 4. Linked Data. I. Título.
CDD 005

CAIO VIKTOR DA SILVA AVILA

CONQUEST: UM FRAMEWORK PARA A CONSTRUÇÃO DE CHATBOTS DE IQA BASEADOS EM TEMPLATES SOBRE KNOWLEDGE GRAPHS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do 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 mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em: 10 de Fevereiro de 2020.

BANCA EXAMINADORA

Prof.ª Dr.ª Vânia Maria Ponte Vidal (Orientadora)
Universidade Federal do Ceará (UFC)

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.ª Dr.ª Vládia Célia Monteiro Pinheiro
Universidade de Fortaleza (UNIFOR)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Vó, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai e mãe, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

À Prof.ª Dr.ª Vânia Maria Ponte Vidal por me orientar durante minha pesquisa e a produção de minha dissertação de mestrado. Prof.ª, sem sua confiança e conselhos esta conquista não seria possível.

Ao Doutorando em Ciência da Computação, José Wellington Franco da Silva, pelo apoio, suporte e dicas durante o programa de mestrado. Wellington, agradeço por sempre ter achado tempo para me ajudar.

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).

Aos colegas de laboratório Amanda Drielly Pires Venceslau e Túlio Vidal Rolim pelas indicações de artigos e pelo apoio durante o curso.

Aos meus pais, avós e irmãos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Aos Professores Doutores José Maria Da Silva Monteiro Filho e Angelo Roncalli Alencar Brayner por seus conselhos e dicas.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

E à Fundação Cearense de Apoio ao Desenvolvimento (Funcap), na pessoa do Presidente Tarcísio Haroldo Cavalcante Pequeno pelo financiamento da pesquisa de mestrado via bolsa de estudos.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

Nesta dissertação é apresentado *CONQUEST*, um *framework* que automatiza grande parte do processo de construção de *chatbots* para a tarefa de *Template-based Interactive Question Answering* (TBIQA) sobre *Knowledge Graphs* (KG) de Domínio Fechado. Os *chatbots* produzidos por *CONQUEST* são capazes de responder questões emitidas em linguagem natural, utilizando a conversação para solucionar os possíveis problemas da ambiguidade e da falta de informação necessária para a formulação da resposta. Para a interpretação da questão, o sistema realiza o processo de classificação de sua intenção para um dos padrões definidos pelo desenvolvedor à priore. Para isto, *CONQUEST* conta com um mecanismo flexível de classificação de questões baseado em aprendizado de máquina (*Machine learning*, ML) que utiliza tanto *features textuais*, quanto *features semânticas* e que se adapta ao uso, aprendendo novas maneiras como uma mesma questão pode ser realizada. Este mecanismo de classificação torna um *chatbot* capaz de tratar o problema da variabilidade linguística. Além disso, um *chatbot* produzido por CONQUEST utiliza o *feedback* do usuário no seu treinamento, adaptando-se ao uso. Como principais contribuições deste trabalho, temos: (1) Uma nova arquitetura para *chatbots* de *Template-Based Interactive Question Answering* (TBIQA) e (2) Uma ferramenta para a criação automática de *chatbots* de TBIQA para *Knowledge Graph* (KG)s. Por fim, com *CONQUEST*, o desenvolvedor investe seu tempo apenas na criação dos padrões de questões suportados pelo sistema, deixando o controle de mensagens, processamento de linguagem natural, interpretação da questão, consulta às fontes de dados e geração das respostas para o *framework*.

Palavras-chave: Interactive Question Answering. ChatBot. Knowledge Graph. Linked Data.

ABSTRACT

In this dissertation *CONQUEST* is presented, a *framework* that automates much of the construction process of *chatbots* for the task of *Template-based Interactive Question Answering* (TBIQA) for Closed Domain *Knowledge Graphs* (KG). The *chatbots* produced by *CONQUEST* are able to answer questions issued in natural language, using conversation to solve the possible problems of ambiguity and the lack of information needed to formulate the answer. For the interpretation of the question, the system performs the process of classifying the user intention to one of the templates defined by the developer a priori. For this, *CONQUEST* has a flexible question classification mechanism based on *Machine learning* (ML) that uses both *textual features* and *semantic features* and that adapts to use, learning new ways how the same question can be realized. This classification mechanism makes a *chatbot* capable of dealing with the problem of linguistic variability. In addition, a *chatbot* produced by *CONQUEST* uses the user *feedback* in its training, adapting to the use. As main contributions of this work, we have: (1) A new architecture for *chatbots* for gls TBIQA and (2) A tool for the automatic building of *chatbots* for TBIQA for KGs. Finally, with *CONQUEST*, the developer invests his time only in creating the question templates supported by the system, leaving the control of messages, natural language processing, question interpretation, querying data sources and generating responses for the *framework* deal with it.

Keywords: Interactive Question Answering. ChatBot. Knowledge Graph. Linked Data.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

TBIQA	*Template-Based Interactive Question Answering*
KG	*Knowledge Graph*
QA	*Question Answering*
NLQI	*Natural Language Query Interfaces*
CONQUEST	*Chatbot ONtology QUESTion*
QAI	*Question Answering Itens*
EKG	*Enterprise Knowledge Graph*
OWL	*Web Ontology Language*
SPARQL	*SPARQL Protocol and RDF Query Language*
QC	*Questões de Competência*
LN	Linguagem Natural
TBQA	*Template-Based Question Answering*
IQA	*Interactive Question Answering*
DS	*Dialog Systems*
IM	*Instant Messenger*
NLP	*Natural Language Processing*
ML	*Machine Learning*
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
SW	*Semantic Web*
WWW	*World Wide Web*
HTML	*Hypertext Markup Language*
RDF	*Resource Description Framework*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*
URI	*Universal Resource Identifier*
LD	*Linked Data*
RDFS	*Resource Description Framework Schema*
IR	*Information Retrieval*
NLI	*Natural Language Interface*

IoT	*Internet of Things*
SE	*Search Engines*
RDB	*Relational Databases*
PLN	*Processamento de Linguagem Natural*
POSTAG	*Part-Of-Speech Tagging*
NER	*Named Entity Recognition*
ASR	*Automatic Speech Recognition*
NLU	*Natural Language Understanding*
NLG	*Natural Language Generation*
SQL	*Structured Query Language*
CA	*Conversational Agent*
KIR	*Keyword-Based Information Retrieval*
AIML	*Artificial Intelligence Markup Language*
BC	*Business ChatBots*
KB	*Knowledge Base*
SQA	*Semantics Question Answering*
AQE	*Automatic Query Expansion*
QPs	*Questões de Pesquisa*
GQM	*Goal-Question-Metric*
EOF	*End of File*
KBQA	*Knowledge Base Question Answering*
CV	*Context Variables*
QP	*Question Pattern*
SP	*SPARQL query Pattern*
RP	*Response Pattern*
CVec	*Context Vector*
RV	*Returned Variables*
QV	*Question Vector*
VS	*Vector Sentence*
OOV	*Out of Vocabulary*
OMS	Organização Mundial da Saúde
MLP	*Multilayer Perceptron*

SUMÁRIO

1 INTRODUÇÃO

Recentemente, o uso de tecnologias do *Linked Data* para a tarefa de integração de fontes de dados em ambientes empresariais vem se popularizando, tornando possível a integração de fontes de diferentes setores, origens, formatos e vocabulários em uma representação única, uniforme e semanticamente integrada (FRISCHMUTH *et al.*, 2012). Nesta abordagem as fontes são disponibilizados através de uma visão semanticamente integrada. Esta visão resultante é então chamada de um *Enterprise Knowledge Graph* (EKG) (Grafos de Conhecimento Empresariais, em tradução livre) (GOMEZ-PEREZ *et al.*, 2017). Um EKG pode ser representado por um vocabulário comum definido por uma ontologia em *Web Ontology Language* (OWL), o que permite que as múltiplas fontes heterogêneas sejam acessadas simultaneamente de maneira transparente através de consultas na linguagem de consultas *SPARQL Protocol and RDF Query Language* (SPARQL) (HEATH; BIZER, 2011; GUI-YANG *et al.*, 2014).

Ainda no contexto empresarial, uma abordagem para a construção de EKGs que vem se consolidando é o uso de *Questões de Competência* (QC) para guiar o processo de construção da ontologia (REN *et al.*, 2014). Nesta abordagem, os especialistas de domínio (pessoas que entendem sobre o domínio da empresa) elencam um conjunto de questões que esperam que a ontologia final seja capaz de responder. Por sua vez, o engenheiro do conhecimento (especialista nas tecnologias de ontologias e KG) utiliza estas questões para identificar os conceitos importantes que devem ser representados no modelo final, além de como estes relacionam-se entre si (FERNANDES *et al.*, 2011).

Deste modo, uma questão de competência pode ser encarada como um padrão de consulta que pode vir a ser realizado com frequência sobre o EKG construído. Portanto, é importante que os especialistas de domínio sejam capazes de realizar estas consultas sobre o EKG de maneira simplificada.

Contudo, a construção de consultas SPARQL é uma tarefa desafiadora. A construção destas consultas exigem conhecimentos técnicos sobre as regras de construção da linguagem SPARQL, além de conhecimento sobre o esquema da ontologia sendo consultada, o que limita o acesso aos dados por parte de colaboradores não técnicos. Deste modo, surge o desafio de como disponibilizar um mecanismo de acesso aos dados que permita que usuários não técnicos recuperem do KG as informações necessárias para a realização de suas atividades.

Uma possível solução para tal desafio é o desenvolvimento de sistemas de QA. Segundo (DIEFENBACH *et al.*, 2018), sistemas de QA sobre KGs tem como objetivo achar no

KG a informação solicitada via Linguagem Natural (LN) pelo usuário. Isto é realizado através da tradução da questão Q em LN para uma consulta Q' em SPARQL que a represente. No enanto, ainda existem limitações sobre os tipos de questões que sistemas de QA atualmente são capazes de responder. Segundo (DIEFENBACH *et al.*, 2018), sistemas de QA de domínio aberto possuem o grande desafio de interpretar corretamente de maneira não ambígua uma questão.

No entanto, tendo em vista o contexto de ambientes empresariais, não é esperado que sejam realizadas questões arbitrárias quais quer sobre o EKG. Neste contexto, é esperado que em geral apenas um conjunto limitado e padronizado de questões relacionadas às atividades do cotidiano da empresa sejam realizadas. Em particular, as questões de competência levantadas durante o processo de construção da ontologia podem servir como uma boa base para a identificação deste conjunto de padrões de consultas.

Neste contexto, ao encarar as QCs como padrões de questões a serem respondidos, surge a oportunidade do uso de sistemas de *Template-Based Question Answering* (TBQA). Um sistema de TBQA recebe uma questão Q em LN e a mapeia para uma padrão de consulta já conhecido Q' escrito em SPARQL que a represente, executando a consulta Q' sobre o EKG e retornando a resposta ao usuário (DIEFENBACH *et al.*, 2018). Nestes sistemas, cada padrão conhecido contém "slots" que são preenchidos com parâmetros fornecidos pelo usuário em tempo de consulta. Estes parâmetros podem atuar como valores para filtros ou propriedades e classes para atender a necessidade específica de cada consulta. Este tipo de sistema provê um mecanismo de consulta de manuseio fácil, natural e acessível a uma ampla gama de usuários (KAUFMANN; BERNSTEIN, 2007). Além disso, sistemas de TBQA possuem a vantagem de reduzirem a tarefa complexa de interpretação de questões em LN para uma tarefa mais simples de classificação de intenção. Isto torna o custo de produção e execução destes sistemas bem mais barato do que o de sistemas de QA capazes de responder quaisquer questões.

No entanto, sistemas de TBQA podem esbarrar em alguns problemas, tais como, (1) classificação inconclusiva da questão para um dos padrões; (2) ausência de parâmetros obrigatórios na questão emitida pelo usuário. Uma possível solução para esses problemas é o uso do diálogo para a desambiguação de intenção e solicitação de parâmetros, gerando assim sistemas de TBIQA. *Interactive Question Answering* (IQA) é um campo de pesquisa que tem emergido na interseção entre sistemas QA e *Dialog Systems* (DS), e que permite ao usuário achar as respostas para suas perguntas em uma maneira interativa, permitindo ao sistema realizar perguntas ao usuário. Hoje em dia sistemas conversacionais são popularmente conhecidos como

chatbots. *Chatbots* são agentes computacionais que servem como interface de usuário em LN para provedores de dados e serviços (DALE, 2016). Os *chatbots* podem ser disponibilizados por meio de aplicativos de *Instant Messenger* (IM) (e.g., *Telegram*, *Facebook Messenger*, *Skype*, etc.).

O processo de construção de um sistema de IQA pode variar grandemente dependo do seu domínio, ferramentas já existentes e propósito (KONSTANTINOVA; ORASAN, 2013). Neste trabalho, é proposto um *workflow* padrão para o processo de criação de sistemas de TBIQA para EKGs:

1. Construção dos padrões de questões suportados pelo sistema;
2. Construção do mecanismo de processamento de linguagem natural;
3. Construção do mecanismo de classificação de uma questão para um dos padrões;
4. Definição do fluxo de interação do sistema.
5. Construção do mecanismo de consulta às fontes de dados;
6. Construção da interface de interação com o usuário;

Neste trabalho, apresentamos *CONQUEST*, um *framework* para a criação de *chatbots* para a tarefa de TBIQA sobre EKGs representados por uma ontologia de domínio fechado. *CONQUEST* automatiza grande parte do *workflow* de desenvolvimento deste tipo de sistema, tratando automaticamente dos passos 2-6. Para a implementação deste passos, *CONQUEST* reusa bibliotecas de *Natural Language Processing* (*NLP*) muito utilizadas em ambientes industriais, aliado ao uso de um mecanismo de um mecanismo baseado em *Machine Learning* (ML) para realizar a classificação das questões de entrada para os padrões conhecidos, recorrendo ao diálogo de clarificação para solucionar classificações inconclusivas e solicitar parâmetros ausentes obrigatórios ao padrão. Além disso, ao confirmar a clarificação de uma questão, *CONQUEST* utiliza o padrão de questão como um novo exemplo para o seu treinamento, adaptando-se ao uso. Ademais, *CONQUEST* lida com o acesso ao *chatbot* ao reusar serviços de IM (e.g., *Telegram, Facebook Mesenger, Skype*, etc.) como seus canais de acesso, além de disponibilizar o acesso ao serviço através de uma *Application Programming Interface* (*API*) REST acessível através de requisições *Hypertext Transfer Protocol* (HTTP), tudo a partir da execução de uma única instância do *chatbot*. Assim, apenas a tarefa de construção dos padrões de questões suportados é delegada ao desenvolvedor. Como principais contribuições deste trabalho, temos:

1. Uma nova arquitetura para *chatbots* de TBIQA;
2. Uma ferramenta para a criação automática de *chatbots* de TBIQA para KGs.

O restante desta dissertação é estruturado da maneira a seguir. O Capítulo 2 apresenta a fundamentação teórica sobre os principais temas necessários para a compreensão desta dissertação. O Capítulo 3 apresenta uma revisão bibliográfica contendo os principais trabalhos presentes na literatura de sistemas de QA e IQA. O Capítulo 4 apresenta o *framework CONQUEST*. O Capítulo 5 apresenta a implementação do *chatbot MediBot* com o uso do *framework CONQUEST* como um estudo de caso. O Capítulo 6 apresenta os principais resultados alcançados neste trabalho. Por fim, o Capítulo 7 apresenta as conclusões e os trabalhos futuros com relação a esta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentado um breve apanhado sobre temas envolvidos no desenvolvimento deste trabalho. A Seção 2.1 apresenta os principais conceitos envolvidos no contexto das tecnologias Semânticas. Por sua vez, a Seção 2.2 aborda as tecnologias envolvidas no desenvolvimento de Interfaces de Linguagem Natural.

2.1 Tecnologias Semânticas

Nesta seção é apresentada uma breve introdução às tecnologias semânticas envolvidas no processo de desenvolvimento deste trabalhos. A Seção 2.1.1 apresenta uma visão geral sobre *Semantic Web* e *Linked Data*. A Seção 2.1.2 apresenta os conceitos de ontologias utilizadas para descrever os dados trabalhados. A Seção 2.1.3 apresenta o conceito de *Knowledge Graphs*. Por fim, a Seção 2.1.4 apresenta os conceitos de *Semantic Search e Semantic Query*.

2.1.1 Web Semântica e Dados Conectados

A Web Semântica ou *Semantic Web* (*SW*) é uma extensão da *Web* tradicional (*Web* de documentos), buscando permitir que humanos e computadores trabalhem em cooperação (W3C, 2009). A *SW* é uma iniciativa para adicionar semântica e significado entendível por máquina às informações existentes na *Web* tradicional sendo proposta pelo fundador da *World Wide Web* (*WWW*), *Tim Berners-Lee* em (BERNERS-LEE *et al.*, 2001).

A principal diferença entre a *Web* de documentos e a *SW* é que enquanto na primeira a informação está armazenada no formato de páginas *Hypertext Markup Language* (*HTML*)[1], enquanto na segunda os dados são estruturados em grafos *Resource Description Framework* (*RDF*)[2] estruturados segundo um esquema taxonômico, denominado Ontologia (BERNERS-LEE *et al.*, 2001). As principais diferenças entre *HTML* e Ontologias são que a primeira é uma linguagem de marcação utilizada para estruturar como a informação será formata para apresentação ao usuário, já a segunda estrutura os dados no formato de triplas que representam os fatos de uma maneira semanticamente compreensível e processável por máquinas.

O *RDF* é baseada em *Extensible Markup Language* (*XML*) com objetivo de promover uma padronização para descrever semanticamente dados na *Web* proposto pela *World Wide Web*

[1] https://www.w3.org/html/
[2] https://www.w3.org/TR/rdf-concepts/

Consortium (*W3C*)(GROUP, 2004). Diferentemente de outras linguagens, o *RDF* é capaz de descrever um objeto à nível semântico, possibilitando que um algoritmo consuma esses dados e compreenda o significado do objeto. Nele podemos definir que um dado objeto do mundo real possui determinada propriedade, definindo *links* entre objetos reais. Em *RDF*, os dados são escritos no formato de triplas: **sujeito, predicado e objeto**. Além disso, o *RDF* trata todas as informações na Web como recursos, onde cada um destes recursos possui um identificador único chamado *Universal Resource Identifier* (*URI*) [3].

O **sujeito** de um tripla geralmente representa um objeto real, como um indivíduo, uma instituição ou um local. O **objeto** pode representar um objeto real ou um literal, *e.g.*, um valor numérico, um texto (*string*) ou uma data. O **predicado** faz o papel de relacionar um **sujeito** à seu **objeto**. Tanto o objeto, quanto o sujeito são representados através de uma *URI*, que os identifica de forma única em toda a Web (GROUP, 2004).

Por sua vez, o termo *Linked Data* (*LD*) refere-se ao conjunto das melhores práticas para publicar e conectar dados estruturados na Web de Dados (BIZER *et al.*, 2011). As tecnologias do *LD* são utilizadas para a criação de *links* entre dados de diferentes fontes. Estas fontes podem ser tão diversas quanto banco de dados mantidos por duas organizações em diferentes localizações geográficas, ou simplesmente, sistemas heterogêneos dentro de uma mesma organização que historicamente não são interoperáveis no nível dos dados(BIZER *et al.*, 2011).

2.1.2 Ontologias

Segundo (GRUBER, 1993), uma Ontologia é uma especificação explícita de uma conceitualização formal. Em outras palavras, uma Ontologia é uma representação estruturada de conceitos existentes em um domínio de conhecimento específico de maneira inequívoca e interpretável por máquinas. No campo da *SW*, uma Ontologia pode ser implementa em linguagens como *Resource Description Framework Schema* (*RDFS*)[4] ou OWL[5], seguindo o padrão *RDF*.

No campo da *SW*, Ontologias OWL são utilizadas para representar um conhecimento complexo acerca de um determinado domínio. Uma Ontologia OWL consiste de uma representação taxonômica de classes e propriedades. Esta representação modela o domínio como uma hierarquia de classes e propriedades, onde conceitos do mundo real são representandos por

[3] https://www.w3.org/TR/uri-clarification/
[4] https://www.w3.org/TR/rdf-schema/
[5] https://www.w3.org/OWL/

classes, enquanto estas classes relacionam-se entre si através de propriedades.

OWL é uma linguagem baseada em lógica computacional, deste modo, o conhecimento expresso em OWL pode ser explorado por programas de computadores, *e.g.*, para verificar a consistência do conhecimento ou para tornar explícito o conhecimento implícito nos dados (W3, 2013).

Uma das características mais marcantes no uso de Ontologias para a representação dos dados é a capacidade da inferência de informações implícitas. Linguagens como OWL e *RDFS* possuem um conjunto de *Inferece Rules* (Regras de Inferência) que permitem o descobrimento de novas informações de maneira automática, através da execução de algoritmos chamados Raciocinadores (*Reasoners*) (BERNERS-LEE *et al.*, 2001). Estes *Reasoners* utilizam um conjunto de regras específicas para cada linguagem para descobrir conhecimento implícito baseado nas informações já conhecidas na Ontologia, e.g., caso a ontologia defina a existências de duas classes, **Classe A** e **Classe B**, onde **Classe B** é uma subclasse de **Classe A**, o *Reasoner* inferirá que todo recurso do tipo **Classe B** também é um recurso do tipo **Classe A**. Ontologias *RDFS* possuem um conjunto de regras de inferência muito reduzido, mas computacionalmente barato, enquanto ontologias OWL possuem conjuntos muito mais complexos, mas mais caro computacionalmente.

Por conta do alto custo computacional do processo de inferência, OWL é divido em subconjuntos de regimes de inferências ou perfis da linguagem, onde cada perfil possui um diferente poder de inferência e custo computacional[6].

2.1.3 Grafos de Conhecimento

O termo KG ou Grafo de Conhecimento foi popularizado pelo *Google* em uma publicação em seu *blog* oficial intitulada "*Introducing the Knowledge Graph: things, not strings*"[7]. A empresa adotou o uso da tecnologia para aprimorar os resultados do seu motor de busca, adicionando-o pequenas "caixas" com informações estruturadas sobre o termo sendo buscando, além de sugerir entidades relacionadas a busca ligadas no KG. Inspiradas pelo *Google*, outras empresas passaram a investir na pesquisa e no uso dos KGs, e.g., *Graph Search* do *Facebook*[8], o *Satori* da *Microsoft* (MICROSOFT, 2019), etc.

Um KG é um conjunto de dados estruturados que é compatível com o modelo de

[6] https://www.w3.org/TR/sparql11-entailment/#OWL2-RDFBS-Profiles
[7] http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html
[8] https://www.facebook.com/graphsearcher/

dados *RDF* e tem uma ontologia OWL como seu esquema (PAN *et al.*, 2017). Um KG não é necessariamente ligado a grafos de conhecimento externos; contudo, este tipo de informação pode ser útil para prover informações contextuais. Um ponto importante sobre KGs é que estes são esperados serem confiáveis, de alta qualidade, alta acessibilidade e serem capazes de prover um serviço de informação orientado ao usuário final (PAN *et al.*, 2017). A Tabela 1 apresenta as diferenças entre *datasets* puramente *RDF*, *datasets* de *LD* e KGs.

Tabela 1 – RDF dataset vs Linked Data vs Knowledge Graph

Features	**Datasets RDF puros**	**Linked Data**	**Knowledge Graph**
Legibilidade de máquina	S	S	S
Legibilidade humana	NN	NN	S
Distribuição de dados	N	S	NN
Ligação entre dataset	B	S	S
Integração de dados	NN	NN	S
Consistência dos dados	NN	NN	S
Confiabilidade	NN	NN	S
Alta qualidade	NN	NN	S

Fonte: (PAN *et al.*, 2017)
Legenda: S = SIM ; N = NÃO ; NN = NÃO NECESSARIAMENTE ; B = BAIXO.

Como um tipo específico de KG, os EKGs são grafos de conhecimento construídos em ambientes empresarias com o propósito específico da construção de um sistema para o serviço de acesso integrado às diversas e heterogêneas fontes de dados existentes na empresa (GUI-YANG *et al.*, 2014). Segundo (PAN *et al.*, 2017), diferentemente dos KGs encontrados na *SW* através do *LD*, um EKG é privado, centralizado, de domínio específico, provavelmente possuindo uma ontologia mais expressiva e possui a aquisição de dados mais simplificada. O aspecto privado se deve ao fato dos dados só estarem disponíveis no ambiente da empresa, diferentemente dos dados disponíveis publicamente na *SW*. O aspecto centralizado decorre do fato de um EKG integrar as diferentes fonte de dados de uma empresa em uma única, enquanto no ambiente da *SW* a informações está espalhada por uma variedade de KGs disponíveis na *Web* de dados, isto também impacta na aquisição dos dados. Por fim, por um EKG tratar apenas sobre dados específicos de uma empresa, este possui um domínio específico e bem conhecido, o que permite a criação de uma ontologia mais rica e expressiva sobre este. A Tabela 2 resume as diferenças entre KGs utilizados para buscas na *Web* e EKGs.

Tabela 2 – Knowledge Graph para Web vs. Knowledge Graph para empresas

Features	**KG para pesquisa na Web**	**KG para empresas**
Fonte de dados	Distribuído	Centralizado
Abertura de dados	Aberto ao público	Privado
Tamanho dos dados	Enorme	Grande
Aquisição de dados	Mais difíceis	Mais fácil
Qualidade dos dados	Baixa	Alta
Linguagem de ontologia	Simples	Provavelmente será mais expressivo
Conhecimento	Conhecimento Genêrico	Conhecimento específico do domínio

Fonte: (PAN *et al.*, 2017)

2.1.4 Busca e Consulta Semântica

O volume de dados criados e replicados em ambientes computacionais vem crescendo em um nível sem precedentes. Segundo (GANTZ; REINSEL, 2011), ainda em 2010, o volume de dados existentes já quebrava a barreira dos *zettabytes* (2^{70} Bytes). Tal cenário está intimamente relacionado ao campo do *Big Data*, onde *datasets* enormes, em geral compreendidos por uma quantidade massiva de dados não estruturados, precisam ser analisados em tempo real (CHEN *et al.*, 2012). Extrair informações úteis deste mar de dados é uma tarefa desafiadora, exigindo tempo, esforço e habilidades técnicas e analíticas. Tradicionalmente, nestes casos, Motores de Busca (*Search Engines*) são utilizados como ferramentas de *Information Retrieval* (*IR*). Os motores de busca são ferramentas que recebem como entrada uma sentença em LN, produzindo como saída um conjunto de trechos que sejam relevantes para a questão da entrada. Tais ferramentas realizam buscas léxicas, ou seja, buscam nos documentos consultando por palavras-chave e similares extraídas da entrada. Hoje em dia os motores de buscas estão cada vez mais presentes na vida cotidiana das pessoas, tendo como alguns exemplos o Google[9], o Yahoo[10] e o Bing[11].

Contudo, tais ferramentas limitam-se a tarefa de busca e comparação textual, não levando em conta o real significado e a intenção por trás da consulta. Deste modo, o resultado gerado consiste de um conjunto de trechos relevantes para a consulta, deixando para o usuário encontrar a resposta específica para sua questão. No entanto, não há a garantias de que os trechos retornados pela busca de fato contenham a resposta para a questão inicial do usuário.

Em oposição a ideia de consulta léxica executada por motores de busca, a Busca Semântica (*Semantic Search*) denota uma busca com significado. Nesta o motor de busca tenta entender o significado geral da consulta, procurando melhorar a precisão da pesquisa,

[9] https://www.google.com/
[10] https://br.search.yahoo.com/
[11] https://www.bing.com/

compreendendo a intenção do pesquisador e o significado contextual dos termos conforme eles aparecem no espaço de dados "pesquisável", seja na *Web* ou em um sistema fechado. Os sistemas de busca semântica consideram vários pontos, incluindo contexto de pesquisa, localização, intenção, variação de palavras, sinônimos, consultas generalizadas e especializadas, correspondência de conceito e consultas de linguagem natural para fornecer resultados de pesquisa relevantes.

Muito próxima ao conceito de busca semântica, a Consulta Semântica (*Semantic Query*) possibilita a recuperação de informações tanto explícitas, quanto implícitas. Neste tipo de consulta, informações são implicitamente derivadas baseadas em informações sintáticas, semânticas e estruturais contidas nos dados. Para a inferência de novas informações, são utilizados axiomas como regras de inferências. Tais axiomas são definidos em uma linguagem formal, e descrevem como um nova informação pode ser inferida a partir de informações já conhecidas (*e.g.*, "se Caio é um homem, então Caio também é uma pessoa"). As consultas semânticas atuam sobre KGs o que permite que a consulta processe os relacionamentos reais entre informações e infira as respostas da rede de dados. Isso contrasta com a busca semântica, que usa a semântica (a ciência do significado) no texto não estruturado para produzir um melhor resultado de pesquisa.

2.2 Interfaces de Linguagem Natural

Nesta seção são apresentados os principais conceitos acerca das Interface de Linguagem Natural (*Natural Language Interfaces*) e suas tecnologias. A Seção 2.2 apresenta uma breve introdução sobre as *Natural Language Interfaces* (NLIs). A Seção 2.2.2 apresenta uma introdução ao campo de pesquisa de *Natural Language Processing* (NLP). A Seção 2.2.3 apresenta o conceito das *Natural Language Query Interfaces* (NLQIs). A Seção 2.2.4 apresenta uma breve introdução ao campo de pesquisa de Dialog Systems (DS). A Seção 2.2.5 apresenta os sistemas de *Keyword-Based Information Retrieval*. A Seção 2.2.6 apresenta a tecnologia emergente dos *Chatbots*. A Seção 2.2.7 apresenta os sistemas de *Question Answering* (QA). Por fim, a Seção 2.2.8 apresenta o campo de pesquisa de sistemas de *Interactive Question Answering* (IQA).

2.2.1 Uma Introdução às Interface de Linguagem Natural

A Linguagem Natural (LN), ou *Natural Language* (NL) em inglês, é um dos meios de comunicação mais eficientes e utilizados por humanos, seja por linguagem falada ou por texto. Tendo isto em vista, o uso da LN na interação homem-máquina apresenta-se como uma opção natural e atrativa. As *Natural Language Interface* (NLI), ou Interfaces de Linguagem Natural no português, são um tipo de interface de comunicação homem-máquina realizada por meio do uso de sentenças e frases, como as utilizadas por humanos em conversas do dia-a-dia, que agem como comandos para sistemas computacionais (HENDRIX, 1982).

Nos anos recentes, o uso de NLIs, vem tornando-se cada vez mais popular. Um dos principais fatores para o atual aumento da popularidade destas é a ampla adoção de dispositivos móveis como *Smartphones*, *Smartwatchs*, dispositivos vestíveis e de *Internet of Things* (*IoT*). O uso de tais dispositivos com pouco ou nenhum espaço de tela disponível apresenta um grande desafio para o projeto de interfaces que permitam a interação com o usuário. Portanto, o uso de NLIs por voz, como assistentes pessoais, tais como *Apple Siri* (INC, 2018), *Microsoft Cortana* (MICROSOFT, 2018b), *Google Assistant* (GOOGLE, 2018a) vem atuando como um grande agente popularizador destas tecnologias.

Outra área na qual o uso das NLIs vem se mostrando bastante popular é a área de *IR*. Em *IR*, é buscado desenvolver sistemas computacionais que possuam a capacidade de obter recursos computacionais, tais como documentos e páginas Web, relevantes para dada tarefa (CROFT *et al.*, 2010). Como um exemplo para o uso mais comum de NLIs em *IR*, temos os amplamente utilizados *Search Engines* (*SE*) da *Web*, tais como o *Google Search* (GOOGLE, 2018b) e *Microsoft Bing* (MICROSOFT, 2018a). Um *SE* recebe como entrada uma sentença em LN que especifica a informação solicita pelo usuário, gerando como saída um conjunto de páginas que contenham informações relevantes para a solicitação do usuário. Um outro exemplo do uso de NLIs em *IR* é o desenvolvimento de interfaces de consulta para fontes de dados estruturados, tais como *Relational Databases* (*RDB*)s e KGs (HENDRIX *et al.*, 1978).

2.2.2 Processamento de Linguagem Natural

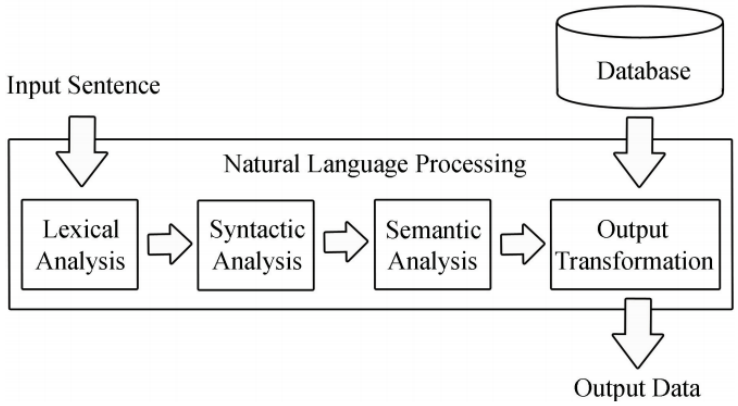
NLP ou *Processamento de Linguagem Natural* (*PLN*) é um sub-campo da inteligência artificial que lida com a interação homem-máquina via linguagem natural. *NLP* explora como computadores podem ser utilizados para entender e manipular linguagem natural por texto ou

por voz (CHOWDHURY, 2003).

Os principais tipos de sistemas *NLP* são: os Baseados em Regras (*Ruled-Based*) e os Baseados em ML. O primeiro é composto por sistemas com regras escritas à mão, estas podendo ser implementadas por meio de expressões regulares, gramáticas ou heurísticas para o tratamento de sentenças. Tais sistemas necessitam de um grande esforço para o seu desenvolvimento, dependendo que o desenvolvedor explore as diferentes possibilidades de entrada. Tais sistemas são limitados quanto a variações linguísticas, o que os torna limitados para um amplo uso (MAYO, 2018).

O segundo utiliza técnicas de ML para automatizar o processo de aprendizado de regras e aumentar a flexibilidade do sistema. Tais métodos utilizam um *corpus* (no plural *corpora*), conjuntos de exemplos de textos do mundo real, como fontes para aprendizado. O *corpus* em LN é convertido para uma representação matricial (*feature matrix*) que é usada como entrada para o modelo de ML que aprende a probabilidade de uma *feature* (termo da entrada) pertencer a um contexto ou interpretação. As principais vantagens dessas abordagens é que estas exigem menos intervenção do desenvolvedor e produzem modelos robustos para entradas não familiares, o que os torna indicados para tratar problemas de variações linguísticas. No entanto, estes modelos necessitam da existência de um *corpus* grande e de boa qualidade para o treino de um bom modelo. (MANNING *et al.*, 1999).

Figura 1 – Passos do Processamento de Linguagem Natural



Fonte: (CERI *et al.*, 2013).

Segundo (TAPSAI *et al.*, 2016) o *workflow* de *NLP* (Figura 1) pode ser dividido nas seguintes etapas:

1. **Análise Léxica (*Lexical Analysis*):** Divide a entrada em segmentos (*tokens*) atômicos para serem processados nas etapas posteriores;

2. **Análise Sintática (*Syntactic Analysis*):** Checa se os *tokens* seguem uma estrutura aceitável pelo sistema. Nesta etapa, algumas técnicas podem ser aplicadas para aplicar regras gramaticais a um grupo de palavras e derivar o significado destas. Segundo (GARBADE, 2018), algumas técnicas envolvidas nesta etapa são:
    a) **Quebra de Sentença (*Sentence Breaking*):** Envolve colocar limites de sentenças em um grande pedaço de texto;
    b) **Lematização (*Lemmatization*):** Isso implica reduzir as várias formas flexionadas de uma palavra em uma única forma para facilitar a análise. Este é o processo complexo e caro computacionalmente, exigindo o uso de regras escritas à mão para cada palavra. Tendo isto em vista, a *Stemização* pode atuar como um substituto para a Lematização;
    c) ***Stemização* (*Stemming*):** Envolve cortar as palavras flexionadas em sua forma raiz;
    d) **Segmentação de Palavras (*Word Segmentation*):** Envolve dividir um grande pedaço de texto contínuo em unidades distintas;
    e) **Segmentação Morfológica (*Morphological Segmentation*):** Envolve dividir palavras em unidades individuais chamadas morfemas;
    f) **Rotulação de Parte da Fala (*Part-Of-Speech Tagging (POSTAG)*):** Envolve identificar a parte do discurso de cada palavra;
    g) **Análise Gramatical (*Parsing*):** Envolve realizar análises gramaticais da sentença fornecida.
3. **Análise Semântica (*Semantic Analysis*):** Interpreta o significado da sentença dada como entrada. Este passo pode utilizar uma ontologia para interpretar e representar o significado interpretado. Segundo (GARBADE, 2018), algumas técnicas envolvidas nesta etapa são:
    a) **Reconhecimento de Entidades Nomeadas (*Named Entity Recognition* (*NER*)):** Envolve determinar as partes de um texto que podem ser identificadas e categorizadas em grupos predefinidos. Exemplos de tais grupos incluem nomes de pessoas e nomes de lugares. Alguns exemplos de plataformas de *NER* são: GATE (SHEFFIELD, 1995) e OpenNLP (FOUNDATION, 2004);
    b) **Desambiguação do sentido da palavra (*Word Sense Disambiguation*):** Envolve dar significado a uma palavra com base no contexto.
4. **Processo de Transformação da Saída (*Output Transformation Process*):** Converte a interpretação encontrada na etapa anterior para uma ação a ser executada. Tal ação pode

ser uma consulta um KG, um comando executado em uma *API*, etc.

Segundo (REITER; DALE, 2000), um sistema de *NLP* pode ser dividido em três principais módulos:

- **Reconhecimento de Fala (*Speech Recognition*):** Também chamado Reconhecimento de Fala Automático, ou *Automatic Speech Recognition* (*ASR*), é um sub-campo de linguística computacional que pesquisa o reconhecimento de linguagem falada e sua tradução para texto. Este módulo é opcional na maioria dos casos, sendo necessário apenas em sistemas que utilizem LN via voz. Onde nestes casos também há a necessidade de acoplar um módulo de síntese de linguagem natural ao módulo *Natural Language Understanding* (*NLU*) que o permita apresentar resposta via fala sintetizada. Em (JIANG, 2005) pode ser encontrado uma apanhado com as diversas técnicas de implementação de sistemas *ASR*s;
- **Entendimento de Linguagem Natural (*NLU*):** Lida com a tarefa de compreensão e entendimento de texto em LN (ALLEN, 1995);
- **Geração de Linguagem Natural (*Natural Language Generation* (*NLG*)):** Lida com a tarefa de geração de texto em LN que represente uma interpretação de máquina de uma resposta.

As técnicas de *NLP* são extremamente versáteis e aplicáveis à diversas tarefas, tais como a classificação de texto, modelagem de sequência de palavras, descoberta de significado de texto, tarefas de *Sequence-to-Sequence*, e em DS (MAYO, 2018).

2.2.3 Interfaces de Consulta em Linguagem Natural

NLQI ou Interface de Consulta em Linguagem Natural (em tradução livre) é um tipo de NLI que tem como objetivo consultar bases de dados e recuperar informações com o uso de consultas em LN, ao invés de recorrer ao uso de linguagens de consultas formais (e.g., *Structured Query Language* (*SQL*), SPARQL, etc). O uso de uma *NLQI* permite a recuperação de informação à usuários leigos de uma maneira simples, rápida e natural sem que haja a necessidade do aprendizado de linguagens de consulta formais, tornando assim a curva de aprendizado próxima a zero (HENDRIX *et al.*, 1978).

Segundo (HENDRIX *et al.*, 1978), o uso de *NLQI*s para o acesso aos dados estruturados traz inúmeros benefícios, tais como:

- Evita a necessidade de o usuário possuir conhecimentos sobre linguagens formais de consulta, tais como *SQL* e SPARQL. O que possibilita o acesso à informação por usuários

não técnicos;

- Torna transparente a estrutura interna de armazenamento dos dados. Assim, o usuário não precisa entender como os dados estão estruturados e armazenados internamente, liberando-o para forca-se em tarefas mais abstratas e relevantes para a sua atividade;
- Utilizam um modo de interação mais natural para o usuário. Dispensam o uso estruturas de *menus* e telas potencialmente complexas. Isto pode facilitar a interação de usuários com pouca familiaridade com tecnologia;
- Quando utilizando linguagem natural por voz, reduz a necessidade de recursos visuais. O que pode ser essencial para portadores de deficiências visuais ou em dispositivos com espaço de tela reduzido.

Entretanto, o uso de LN como linguagem de consulta implica no surgimento de alguns problemas inerentes da mesma, tais como a ambiguidade da consulta, a variabilidade linguística, alto custo de processamento e necessidade de adaptação manual da ferramenta ao domínio específico de uso (KAUFMANN; BERNSTEIN, 2007).

2.2.4 Sistemas Conversacionais

DS, *Conversational Agent* (*CA*) ou Agentes Conversacionais (em tradução livre) são sistemas computacionais que tem como objetivo conversar com seres humanos via LN de uma maneira coerente. Um DS pode realizar uma ampla variedade de tarefas, desde simular um humano durante conversas do cotidiano, até realizar tarefas mais específicas como automatização de processos e suporte a tomada de decisões (CASSELL *et al.*, 2000).

Um DS também pode ser utilizado para responder perguntas do usuário sobre um determinado domínio, atuando assim como uma *NLQI*. A principal diferença entre uma *NLQI* tradicional e um DS dar-se pelo tipo da interação com o usuário. Enquanto uma *NLQI* atua apenas como uma ferramenta de perguntas e respostas, geralmente de maneira direta e curta, um DS busca interagir com o usuário de maneira mais imersiva e longa. Deste modo, em um DS há a necessidade da criação de um fluxo de conversação coerente com múltiplas perguntas e respostas que levem em consideração o contexto e o histórico da conversa. Por conta de tais aspectos, a construção de um DS para *NLQI* requer uma maior esforço, quando comparado a sistemas de *NLQI* tradicionais.

2.2.5 Sistemas de Recuperação de Informação Baseados em Palavras-chaves

Os *Keyword-Based Information Retrieval* (*KIR*) ou Sistemas de Recuperação de Informação Baseados em Palavras-Chave (em tradução livre), muitas vezes chamados apenas de *Informations Retrieval Systems*, são um tipo específico de NLQI. Estes sistemas são responsáveis por obter recursos como a informação relevante para a consulta do usuário sobre uma coleção de recursos de informação. Os recursos podem estar disponíveis em documentos, como na web, também como metadados e bancos de dados de textos, imagens e sons. Um *KIR* recebe como entrada um conjunto de *keywords*, ou palavras-chave, em LN e busca recursos que são relevantes para esta entrada(CERI *et al.*, 2013).

Um sistema *KIR* costumeiramente atua sobre bases não estruturadas, como coleções de textos, tendo como objetivo retornar um subconjunto de textos ou trechos que contenham informações relevantes sobre os termos de entrada. Um ponto diferencial quanto aos demais sistemas de *NLQI* é que um *KIR* não tem como objetivo primário responder diretamente a questão emitida pelo usuário, mas sim oferecer informação que o ajude a tomar decisões ou resolver problemas. Deste modo, os resultados retornados por um sistema *KIR* podem não ter relação direta com a resposta esperada para a pergunta, podendo ter diferentes graus de relevância, dai surge a necessidade de calcular a relevância de cada um dos recursos (em geral documentos) e ordená-los. Pelo fato destes sistemas geralmente não apresentarem respostas sucintas, estes exigem mais esforço interpretativo do usuário para que as informações retornadas sejam úteis. Além disso, estes sistemas não levam em consideração a semântica da consulta emitida, limitando-se ao processamento e a busca de *keywords*.

Segundo (CERI *et al.*, 2013), a operação de sistemas de *KIR* é dividida em seis estágios:

1. **Entrada do Usuário:** O usuário especifica sua consulta em texto por *keywords*;
2. **Refinamento de Entrada:** O sistema refina a entrada, aplicando técnicas de *NLP*, tais como, *tokenization* e *stemization*;
3. **Representação em Nível de Sistema:** O sistema aplica transformações para representar a consulta em LN para uma consulta formada por um conjunto de operações em nível de sistema;
4. **Recuperação de Recursos:** A consulta é executada sobre a base com o uso de um índice para recuperar os documentos relevantes;
5. **Ranqueamento de Documentos:** Os documentos selecionados são ranqueados de acordo

com sua relevância para a necessidade do usuário;

6. **Retorno e Feedback:** O sistema retorna os documentos para o usuário que pode selecioná-los e oferecer *feedback* para o sistema sobre a relevância das informações apresentadas.

2.2.6 Chatbots

ChatterBot ou simplesmente *Chatbot*, é um agente computacional que possui a capacidade de se engajar em diálogos em LN como um humano. Enquanto alguns *chatbots* são projetados com o único objetivo de simular humanos, como em conversas banais, existem também os que são orientados a objetivos. Exemplos destes objetivos são a automatização de tarefas anteriormente realizadas por atendentes humanos, *e.g.*, serviços de atendimento ao consumidor, vendedores online, envio de mensagens e notícias (DALE, 2016).

Segundo (BRADEŠKO; MLADENIĆ, 2012), a evolução dos *chatbots* tem contado com um grande número de tecnologias, dentre estas, a seguintes podem ser ressaltadas:

- **Casamento de Padrões (*Pattern Matching*):** Abordagem adotada mais comum. Nesta abordagem o desenvolvedor define um padrão de entrada esperado pelo *chatbot* e um padrão de saída a ser respondido. O padrão pode ser definido com o uso de expressões regulares (*Regex*);
- ***Parsing*:** Nesta abordagem o texto de entrada original é segmentado em *tokens*, que são então estruturados como uma árvore que representa a estrutura léxica da sentença de entrada. Esta abordagem pode ser implementada com o uso de interpretadores gramaticais que avaliam se a entrada é gramaticalmente aceita pela linguagem, seguindo o uso dos termos e sua ordem correta;
- **Modelos de Cadeia de Markov (*Markov Chain*):** Esta abordagem vê uma conversa como uma máquina de estados, onde a probabilidade do próximo estado (resposta do *chatbot*) depende somente do estado atual (pergunta do usuário). São então utilizados modelos estatísticos de *Markov* para calcular a probabilidade de uma saída;
- **Ontologias ou Redes Semânticas:** Utiliza Ontologias para estruturar hierarquicamente o conhecimento sobre determinado um domínio, contendo seus conceitos e relações. É então tentado mapear a entrada do usuário para termos existentes na Ontologia, para assim construir uma interpretação da pergunta estruturada como um grafo. Isto permite o uso de técnicas de busca em grafo para a inferência de fragmentos faltosos na pergunta e, por fim, a computação da resposta;

- **AIML:** A *Artificial Intelligence Markup Language* (*AIML*) é uma linguagem de marcação baseada em *XML* para a criação de conversas em LN para *chatbots*. *AIML* usa uma abordagem semelhante ao *pattern matching*, onde é definido um padrão de entrada esperado junto ao seu padrão de saída correspondente. O diferencial de *AIML* está no fato desta ser uma linguagem de padrões (padrão de pergunta-resposta) estruturados, o que garante reuso e interoperabilidade. O grande trunfo de *AIML* esta na sua capacidade de auto-referência onde, um padrão pode chamar a si mesmo e passar uma nova entrada como parâmetro;
- ***ChatScript*:** Almeja ser o sucessor do *AIML*, incorporando características de ontologias, como conceitos, variáveis e fatos ao mecanismo de gerência de diálogo do *AIML*. O seu código está disponível no *github* (CHATSCRIPTNLP, 2018);
- **RiveScript**: Assim como *AIML* e *ChatScript*, é uma linguagem de *script* para criação de *chatbots*. *RiveScript* vem se popularizando bastante no processo de criação de *chatbots*, tendo implementações para diversas linguagens como *Go*, *Java*, *JavaScript*, *Perl* ou *Python*. Um dos pontos fortes de *RiveScript* é sua simplicidade, sendo implementada em texto simples seguindo alguns padrões, não havendo a necessidade de uso de *XML* e afins (PETHERBRIDGE, 2019).

Existem diversos tipos e classificações de *chatbots*, ainda não havendo consenso nessa área. A seguir são apresentadas uma categorização de *chatbots* de acordo com seus objetivos.

2.2.6.1 Business ChatBot

Os *Business ChatBots* (*BC*) são um tipo específico de *chatbot*. A principal característica de um *BC* é o fato destes serem orientados a uma tarefa específica, ou seja, o *chatbot* utiliza o diálogo com o objetivo final de realizar alguma tarefa, assim o diálogo em LN atua como um meio para alcançar um objetivo final.

Esse tipo de *chatbot* tem se tornado particularmente popular, principalmente devido a grande adoção de serviços de IM (e.g., *FaceBook Messenger, Telegram, WhatsApp, Skype, Slack*). Apenas no *Facebook*, em 2017 foram registrados 100 mil *chatbots* ativos no *messenger*, com mais de 2 bilhões de mensagens trocadas por mês. Segundo (MAGAZINE, 2018), os principais setores a adotar a tecnologia foram: vendas online (34%), cuidados de saúde (27%), telecomunicações (25%), bancos e finanças (20%).

Dentro da comunidade de desenvolvimento existem inúmeras discussões sobre

diferentes tipos de classificações para um *BC*. Neste trabalho foi optado por seguir a classificação adotada pela equipe da IBM, uma das primeiras empresas a investir em grande escala na área de inteligência artificial (IBM, 2017):

- ***Support Chatbots*:** Possuem um conhecimento profundo sobre um domínio restrito. Estes *chatbots* tem o intuito de dar suporte ao usuário durante a realização de tarefas. Alguns pontos necessários para um *support chatbot* são uma personalidade que se adéque ao seu propósito, capacidades de conversação multi-turno (interações longas consecutivas) e consciência do contexto. Este tipo de *BC* deve ser capaz de dar suporte ao usuário durante a realização de grandes processos de um negócio, além de ser capaz de responder uma ampla faixa de perguntas sobre o domínio. A facilidade de navegação e execução de tarefas devem ser vistas como prioridade no processo de desenvolvimento. Sendo o uso e reconhecimento de fala um aspecto opcional;
- **Skills Chatbots:** Possuem um conhecimento raso (*shallow*) sobre um conjunto de domínios restritos. Estes *chatbot* possuem um conjunto de comandos (*skills* ou habilidades) que buscam facilitar a vida do usuário. Este tipo de *chatbot* atua como uma interface de controle, podendo executar ações de maneira rápida. Logo, geralmente tais *chatbots* possuem interações de um único turno, não havendo a necessidade do engajamento em grandes conversas, assim não requerem uma grande consciência de contexto. Alguns pontos necessários para um *skill chatbot* são: integração com diversos objetos e sistemas inteligentes, interação por voz e possuírem comandos rápidos;
- ***Assistant Chatbots*:** Possuem um conhecimento raso sobre um domínio amplo. Estes *chatbots* tem o intuito de atuarem como assistentes pessoais do usuário, tendo a capacidade de conversar sobre diversos tópicos, e no futuro atuar como intermediários entre o usuário e outros *chatbots* de *support* e *skill* a para realização de tarefas. Alguns pontos necessários para um *assistant chatbot* são: conhecimento sobre uma ampla variedade de tópicos e ter boas habilidades de conversação; A ampla variedade de questões utilizadas para o treinamento do *chatbot* são de extrema importância durante o seu desenvolvimento; além disso, o comando por voz é um bom aspecto para este tipo de *chatbots*.

2.2.6.2 Persona ChatBot

Os *Persona ChatBots* são agentes computacionais que possuem a capacidade de simular uma conversação em linguagem natural como um humano. Mais do que somente

simular a conversação, um *Persona Chatbots* também é capaz de simular uma personalidade humana, possuindo história, gostos, interesses, habilidades e outras características próprias de um ser humano. Por possuírem uma "personalidade real", daí surge seu nome "Persona", que na psicologia significa "personalidade que o indivíduo apresenta aos outros como real". Este é o tipo de *chatbot* que mais faz jus ao "chat" em seu nome, por ter como principal objetivo conversar da maneira mais humana possível. Como mostrado em (ABDUL-KADER; WOODS, 2015), os *Persona chatbots* podem adotar os chamados *Languages Tricks* (Truques de linguagem em português) para simular erros, como soletração errada, costumeiramente feitos por humanos.

Não existe um padrão para o processo executado por um *Persona chatbots*, existindo uma ampla variedade de técnicas que vão desde uso de simples "*if*s" e "*else*s" ou expressões regulares como gatilhos para a emissão de repostas pré-programadas; uso de *scripts* de conversas inteiras codificadas em linguagens como AIML e *ChatScript*; até o uso de ML e complexas arquiteturas híbridas.

Alguns exemplos clássicos deste tipo de *chatbots* são o *Eliza* (SHAH *et al.*, 2016), posteriormente evoluído para *Elizabeth* (SHAWAR; ATWELL, 2002), conhecido como o primeiro *chatbot*, e projetado para atuar como um psicólogo. Um outro exemplo bastante popular é o *ALICE*, um dos primeiros *chatbots* a usar *AIML* (SHAWAR; ATWELL, 2002). Uma lista contendo os *chatbots* campeões das ultimas edições do prêmio *Loebner* pode ser visto em (BRADEŠKO; MLADENIĆ, 2012). O prêmio *Loebner* é uma instância do "*Jogo da Imitação*", proposto por A. M. Turing, tendo início em 1996 até os dias de hoje.

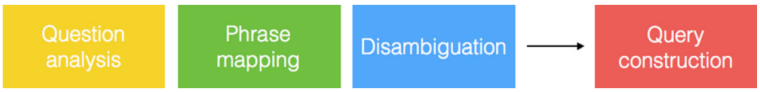
2.2.7 Sistemas de Question Answering (QA)

De acordo com (HIRSCHMAN; GAIZAUSKAS, 2001), os Sistemas de Question Answering (QA) são sistemas computacionais que são capazes de responder questões realizadas em LN. Um sistema de QA tem como objetivo interpretar uma pergunta feita por um humano e responde-la de maneria natural e sucinta, dispensando a necessidade do uso de linguagens de consulta formais como *SQL* e SPARQL. Este tipo de sistema possui uma curva de aprendizado suave, o que expande o acesso à informação para usuário não-técnicos. Um sistema de QA pode atuar sobre uma grande variedade de fontes de dados, tais como um *corpus*, *RDB*s, KGs e outros tipos de *Knowledge Base* (KB)s atuando como uma camada de acesso aos dados de maneira rápida e prática. Segundo (DIEFENBACH *et al.*, 2018), sistemas de QA sobre bases de conhecimento (QA_KB, *Question Answering over Knowledge Base*) tem como objetivo achar no

KB a informação solicitada via LN pelo usuário. Isto é realizado através da tradução da questão Q em LN para uma consulta Q' em SPARQL que a represente. Sistemas de QA_KB também são conhecidos como sistemas de *Semantics Question Answering* (SQA) (HÖFFNER *et al.*, 2017).

As principais diferenças entre um sistema de QA e um sistema de *KIR* são que enquanto um *KIR* é baseado na recuperação de documentos que contenham as *keywords* de entrada, um sistema de QA busca interpretar a questão da entrada e retornar sua resposta sucintamente (ALLAM; HAGGAG, 2012). Além disso, dependendo da base utilizada, tal como um EKG como um KB, um QA pode responder questões as quais suas respostas não estão explicitamente presentes na base.

Figura 2 – Tarefas no processamento de QA



Fonte: (DIEFENBACH *et al.*, 2018).

Segundo o *survey* apresentado por (DIEFENBACH *et al.*, 2018), os principais sistemas de QA construídos até então seguem os seguintes passos (Figura 2):

1. **Análise de questão (*Question Analysis*):** Nesta etapa, a questão do usuário é analisada com base em características puramente sintáticas, realizando a segmentação da sentença, reconhecimento de entidades nomeadas (*NER*) e a criação da árvore de dependência da frase;
2. **Mapeamento de frases (*Phrase Mapping*):** Nesta etapa, as frases (uma ou mais palavras da entrada) são mapeadas para recursos da base de conhecimento, tais como instâncias, classes ou propriedades;
3. **Desambiguação (*Desambiguation*):** Nesta etapa, são solucionadas possíveis ambiguidades das entapas anteriores, onde no passo 1 há a possibilidade de haver ambiguidade na segmentação ou na dependência entre segmentos, já no passo 2 uma mesma frase pode ser mapeada para múltiplos recursos;
4. **Construção de consulta (*Query Construction*):** Nesta etapa, é realizada a construção de uma consulta em uma linguagem formal, e.g., SPARQL, que represente a interpretação da questão. Esta etapa pode ser realizadas das seguintes formas:
   - **Baseada nas informações da análise de questão:** É criada uma árvore de análise sintática (*parse tree*) que é utilizada para a dedução da consulta SPARQL;
   - **Baseada em *parser* semântico:** O sistema une as regras sintáticas da *parser tree*

às regras semânticas, tais como hierarquia de classes, domínio e contra-domínio de propriedades;

– **Baseada em padrões (*Template-Based*):** O sistema tenta classificar a questão do usuário para padrões em linguagens formais de consultas, possuindo *slots* a serem preenchidos por parâmetros informados pelo usuário.

Contudo, embora sistemas de QA sejam capazes de abstrair a complexidade da linguagem de consulta, estes ainda necessitam que o usuário possua conhecimentos sobre o vocabulário da ontologia sendo consultada. Um dos principais desafios para o uso de sistemas de SQA por usuários leigos no domínio é o chamado desafio da lacuna léxica (*Lexical Gap*). Lacuna léxica, também conhecido como lacuna de vocabulário (*Vocabulary Gap*), refere-se aos casos onde o vocabulário utilizado pelo usuário e o vocabulário usado para formalmente representar os dados podem ser substancialmente diferentes (HAKIMOV *et al.*, 2015). Isto pode ocorrer por conta de erros ortográficos e gramaticais, pelo uso de sinônimos ou quando o nível de abstração utilizado pelo usuário difere daquele modelado na fonte.

De acordo com (HÖFFNER *et al.*, 2017), algumas estratégias utilizadas para tratar tais desafios, em geral, na etapa de mapeamento da frase, são:

– **Normalização de *Strings* e Funções de Similaridade:** Nesta abordagem, são utilizadas funções de normalização, tais como remoção de *stopwords*, conversão de texto para caixa baixa (*lower case*), stemização[12] (reduzir uma palavra ao seu radical), remoção de acentuação e caracteres especiais, etc. Tais técnicas já são suficientes para tratar os casos mais simples de erros de digitação e diferenças de tempos verbais.

No entanto, caso a simples normalização não seja suficiente, o uso de funções de similaridade pode mostra-se uma alternativa. As funções de similaridade realizam comparações que vão além se dois termos são iguais. Alguns exemplos são o quanto uma palavra é semelhante a outra na quantidade de edições necessárias para igualá-las, ou o quanto duas palavras soam semelhantes, ou se apenas trechos das palavras são iguais.

Contudo, a aplicação de funções de similaridades traz um maior custo computacional dependendo de sua complexidade. Além disso, tais estratégias são capazes de solucionar apenas casos onde os termos utilizados pelo usuário se assemelham de alguma maneira aos codificados na Ontologia. Deste modo, tal abordagem não é efetiva quanto ao uso de sinônimos e diferentes níveis de abstrações;

[12] https://pt.wikipedia.org/wiki/Stemiza%C3%A7%C3%A3o

– **Expansão de Consulta Automática (*Automatic Query Expansion* (*AQE*)):** Nesta abordagem, são utilizados rótulos adicionais para identificar os termos e conceitos presentes na Ontologia, deste modo expandindo a base de conhecimento. Para tal, *databases* léxicos, tais como o *WordNet* podem ser utilizados. *WordNet* possui uma larga base de palavras e seus sinônimos, além de relações de hiperônimo (palavras mais genéricas) e hipônimo (palavras mais específicas), enriquecendo assim os aspectos linguísticos da base de conhecimento (MILLER, 1995). O uso desta técnica permite um considerável aumento na taxa de *recall* do sistema (quantidade de questões que o sistema consegue responder).
  Contudo, o seu uso pode levar o sistema a cometer erros, por conta do mapeamento incorreto de termos, levando assim a diminuição da taxa de precisão (quantidade de respostas corretas dadas pelo sistema). Isto ocorre, pois, duas palavras podem ser sinônimos apenas em contextos específicos ou em um determinado contexto a substituição de uma palavra por um conceito mais genérico (ou vice-versa) induz a inserção ou remoção de elementos no conjunto de resposta. Deste modo, fica claro que o contexto do emprego de uma palavra é de suma importância na tarefa de mapeamento;
– **Biblioteca de Padrões:** Nesta abordagem, são definidos padrões que representam conceitualmente uma propriedade. Padrões podem ser definidos para identificar o uso de uma propriedade tratando a diferença de posições em seus argumentos, ou até mesmo abstraindo relações complexas codificadas por caminhos de propriedades.
  Entretanto, tal estratégia depende da criação da biblioteca de padrões. Embora hajam esforços para a construção de grandes bibliotecas (FADER *et al.*, 2013) e ferramentas de geração automática (GERBER; NGOMO, 2012), estas ainda limitam-se a padrões mais simples. Deste modo, não é raro ser necessária a construção manual da biblioteca de padrões para aplicações mais especializadas;
– **Uso de Inferências:** Nesta abordagem, um *corpus* de questões respondidas ou padrões de questões já conhecidas podem ser utilizados para inferir respostas para novas questões. Toda via, esta abordagem é dependente da existência de tal *corpus*, limitando assim sua aplicabilidade;
– **Abordagens compostas:** Nesta abordagem, soluções híbridas são criadas. Tais soluções utilizam mais de uma das abordagens anteriores, buscando amenizar seus pontos negativos.

2.2.8 Sistemas de Interactive Question Answering (IQA)

Interactive Question Answering (IQA) é um campo de pesquisa que tem emergido na interseção entre sistemas de QA e DS, e que permite ao usuário achar as respostas para suas perguntas em uma maneira interativa, permitindo ao sistema realizar perguntas ao usuário (KONSTANTINOVA; ORASAN, 2013). Assim como um sistema de QA, um sistema de IQA tem como objetivo interpretar e compreender uma pergunta feita por um humano e respondê-la de maneria natural e sucinta, dispensando a necessidade do uso de linguagens formais de consulta como *SQL* e SPARQL. A principal diferença entre um sistema de QA e um de IQA reside no fato dos últimos possuírem a capacidade de manter iterações longas com o usuário, respondendo várias perguntas relacionadas em sequência, enquanto o primeiro limita-se a iterações curtas de perguntas e respostas independentes.

Durante as pesquisas deste trabalho, não foi possível identificar um padrão de como um sistema de IQA deva ser construído ou quais funcionalidades um sistema deste tipo deva dar suporte através de sua abordagem interativa. A seguir são destacados os principais recursos (funcionalidades) suportados pelos sistemas de IQA disponíveis na literatura:

- **Evolução através de *Feedbacks*:** Permitindo que o usuário opine sobre a utilidade das informações apresentadas, o que pode levar a adaptação e evolução do sistema;
- **Diálogo de Clarificação:** O sistema realiza perguntas ao usuário para solucionar possíveis **ambiguidades** na interpretação da intenção do usuário. Além disso, a clarificação também pode atuar como um mecanismo de **gerenciamento de restrições**, permitindo ao usuário ajustar o nível de restrição sobre uma consulta, clarificando o escopo da questão. Um dos casos mais comuns apresenta-se quando a consulta interpretada possui um conjunto de retorno vazio ou muito amplo, neste caso o sistema permite que o usuário relaxe ou reforce as restrições;
- **Suporte à Perguntas Complementares:** O usuário pode realizar novas perguntas que usam informações sobre perguntas ou resultados anteriores como base. Podendo, por exemplo, pedir mais informações sobre um elemento específico de uma lista apresentada como resposta. Os problemas mais comuns durante a interpretação de perguntas complementares são:
  - ***Anaphora:*** Também chamado de problema de correferência. A Correferência trata-se no campo da linguística da ocorrência de uma ou mais expressões que referem-se à mesma pessoa ou objeto. Este problema apresenta-se quando uma expressão ou

pergunta é referente implicitamente à um termo anteriormente citado (MITKOV, 1999). Por exemplo:

```
Q1: Quem os carabineiros prenderam?
A1: Lorenzo Tinnirello.
Q2: Quando ele foi preso?
A2: No sábado.
```

Neste exemplo, o problema de correferência consiste na necessidade de relacionar o pronome "*ele*" de *Q2* ao indivíduo apresentado em *A1*. Neste caso, "*ele*" é um "*anaphoric pronoun*".

Outro exemplo pode ser visto na frase "O presidente dos EUA, *George W. Bush* Chegou a Nova Deli hoje. *Bush* disse que estava muito feliz com a visita.". Neste exemplo, "*Bush*" na segunda sentença é referente ao mesmo "*George W. Bush*" na sentença anterior. Neste caso, "*Bush*" é um "*anaphor noun*";

- ***Ellipses:*** Na linguística *ellipsis* (plural *ellipses*, literalmente significa o sinal de reticências "...") indicam a omissão do restante da frase. Este problema apresenta-se quando uma expressão ou pergunta é incompleta (DALRYMPLE *et al.*, 1991). Por exemplo:

```
Q1: Quem os carabineiros prenderam?
A1: Lorenzo Tinnirello.
Q2: Quando ele foi preso?
A2: No sábado.
Q3: Onde?
A3: Próximo a palermo.
```

Neste exemplo, o problema de omissão consiste na necessidade de relacionar a informação "onde" com o contexto ao qual ela pertence, este sendo a prisão de *Lorenzo*.

- **Sugestão de Informação:** O sistema pode sugerir informações não solicitadas que possam ser de interesse do usuário, dando a opção de receber novas informações ou não. Tal recurso pode aumentar o engajamento do usuário ou antecipar informações que viriam a

ser solicitadas no futuro, diminuindo o esforço do usuário e aumentando sua satisfação. Por exemplo:

```
Q1: Quem os carabineiros prenderam?
A1: Lorenzo Tinnirello.
Você gostaria de saber quando e onde?
Q2: Sim.
A2: No sábado, próximo a palermo.
```

– **Suporte à Questões Complexas:** Questões complexas são perguntas que podem ser decompostas em múltiplas perguntas factuais. Os sistemas de IQA podem fragmentar questões complexas e responder cada fragmento separadamente. Por exemplo:

```
Q1: Quem os carabineiros prenderam e quando?
A1: Lorenzo Tinnirello.
A2: No sábado.
```

Além deste cenário, questões complexas podem necessitar parâmetros para sua resolução. Onde, estes parâmetros pode estar ausentes ou implícitos na questão original. Deste modo, o sistema pode recorrer ao uso do diálogo interativo para solicitar ou esclarecer estes. Por exemplo:

```
Q1: Quais prisões foram registradas?
A1: Em que dia?
Q2: Neste sábado.
A2: Lorenzo Tinnirello foi preso no sábado.
```

Segundo (HICKL *et al.*, 2006), sistemas de IQA possuem três desafios quando comparados a sistemas de QA tradicionais:

1. Sistemas IQA devem sempre buscar prever as possíveis futuras interações com o usuário, tendo como base o histórico de seu processo de busca. Para assim, tomar ações que mantenham o engajamento do usuário, *e.g.*, sugerir informações que possam ser de seu

interesse ou que são constantemente buscadas em conjunto com as já realizadas durante a interação;

2. Sistemas de IQA devem possuir a capacidade de responder perguntas complexas que precisam de várias outras perguntas mais simples para serem respondidas;
3. Por fim, sistemas de IQA devem levar em consideração o contexto do usuário durante a pergunta, tendo conhecimento sobre o que o usuário "sabe" e "quer saber", para evitar informações redundantes e irrelevantes.

3 REVISÃO BIBLIOGRÁFICA

Nesta seção, é apresenta uma revisão da literatura acerca das temáticas de sistemas de QA e IQA. Esta revisão atuou como ponto de partida desta pesquisa, ajudando a identificar os desafios e oportunidades existentes nos referidos campos de pesquisa. Além disso, esta revisão pode atuar como um resumo da evolução e do atual estado-da-arte para os leitores não familiarizados com os temas tratados neste trabalho. O restante desta Seção é estruturado da maneira que se segue: A Seção 3.1 apresenta a metodologia seguida para a realização desta revisão bibliográfica; a Seção 3.2 apresenta os principais sistemas de QA tradicionais encontrados durante esta revisão; a Seção 3.3 apresenta os principais sistemas de IQA encontrados durante esta revisão; a Seção 3.4 apresenta as principais técnicas utilizadas para avaliar sistemas de QA e IQA; por fim, a Seção 3.5 apresenta discurções sobre os resultados desta revisão, além de uma breve comparação entre os trabalhos publicados e este atual.

3.1 Metodologia de Pesquisa

Uma Revisão Sistemática é um estudo secundário com objetivo de identificar, avaliar e analisar diversos estudos primários, i.e., estudo que gera resultado, acerca de determinado tópico, área de estudo ou fenômeno seguindo uma metodologia rigorosa (MALLETT *et al.*, 2012). Um elemento crítico ao realizar uma revisão sistemática é o desenvolvimento de um protocolo. O protocolo especifica todos os passos realizados durante a revisão e é o elemento que que determina o grau de confiabilidade do estudo. No protocolo são especificados, por exemplo, as seguintes metodologias: (I) Como encontrar estudos relevantes? (II) Quais estudos estão ou não relacionados com a temática? (III) Quais estudos devem ser analisados? Dentre outros.

O protocolo é iniciado definindo-se as *Questões de Pesquisa* (*QPs*). Em seguida, a estratégia para busca manual e automática dos artigos primários é definida, de modo que a maior quantidade de artigos primários seja encontrada. Após encontrar uma gama de estudos, é apresentado um método para seleção dos estudos encontrados, levando-se em consideração critérios para inclusão e exclusão de tais estudos. Também são definidos critérios para avaliar a qualidade dos estudos, levando em consideração suas procedências. Finalmente, é apresentada uma estratégia para selecionar o que deve ser extraído de cada estudo, a fim de realizar uma discussão sobre eles (MALLETT *et al.*, 2012).

3.1.1 Questões de Pesquisa

Para a definição das questões de pesquisa abordadas nesta revisão sistemática, foi utilizado o método *Goal-Question-Metric* (*GQM*) (CALDIERA; ROMBACH, 1994). Este método define que para encontrar questões de pesquisas, devem ser especificados *Purpose, Issue, Object e Viewpoint* (PIOV); propósito, assunto, objeto e ponto de vista, respectivamente:

- ***Purpose:*** Analisar as principais características;
- ***Issue:*** Interfaces de consulta via linguagem natural sobre grafos de conhecimento;
- ***Object:*** Sistemas de QA e IQA;
- ***Viewpoint:*** Do ponto de vista dos recursos de interação com o usuário e apresentação dos resultados.

Assim, foram definidas as seguintes questões de pesquisa:

- ***QP1:*** O sistema permite a interação multi-turno com o usuário?
- ***QP2:*** Se o sistema permite a interação multi-turno, a quais recursos interativos ele dá suporte (com relação a Seção 2.2.8)?
- ***QP3:*** Qual o tipo de fonte consultada pelo sistema?
- ***QP4:*** O sistema apresenta uma resposta concisa ao usuário?
- ***QP5:*** Qual o principal desafio dos sistemas existentes?

3.1.2 Estratégia de Busca

Nesta etapa, é definida qual metodologia utilizada para encontrar os artigos discutidos nesta revisão sistemática. Primeiramente, foi realizada uma etapa de busca com as strings "*Question Answering*", "*Interactive Question Answering*", "*Natural Language Query Interface*" e "Information Retrieval Systems" com o objetivo de conhecer os principais artigos primários e secundários sobre o tema.

A busca foi realizada concatenando os termos de pesquisa por predicados OR e realizadas sobre as bases de dados. A busca teve o escopo de publicações a partir do ano de 2000 até o ano de 2019, limitando-se a publicações com o idioma inglês. Após o processo de recuperação da publicações, foi realizado o processo de exclusão de duplicatas manualmente. As bases consultadas foram:

- *IEEExplore*[1]

[1] http://ieeexplore.ieee.org

– *ACM Digital Library*[2]
– *Science Direct*[3]

3.1.3 Estratégia de Seleção

Os critérios que foram julgados relevantes para a realização desta revisão foram:

– **I1:** O artigo descreve uma abordagem para interfaces de consulta em linguagem natural.

Os critérios que foram julgados a exclusão de publicações desta revisão foram:

– **E1:** O artigo foi publicado fora do período de 2000 até 2019;
– **E2:** O artigo não é escrito no idioma inglês;
– **E3:** O artigo trata de uma ferramenta de *KIR* tradicional;
– **E4:** A publicação não está disponível gratuitamente na *Web*.

3.1.4 Critérios de Avaliação de Qualidade

Após o processo de busca e seleção dos trabalhos, um total de 23 publicações foram avaliadas. Para medir a qualidade da publicação foram utilizadas as seguintes métricas de qualidade:

– **M1:** O artigo apresenta a arquitetura da abordagem de maneira completa e clara?
– **M2:** O artigo apresenta um exemplo ou estudo de caso do uso da abordagem?
– **M3:** O artigo disponibiliza um link para download da ferramenta ou do código fonte?

3.1.5 Estratégia de Extração

Durante a leitura de cada artigo foram extraídas as informações acerca das Questões de Pesquisa apresentadas na Seção 3.1.1 e sobre as Métricas de Qualidade definidas em 3.1.4. Estas informações foram então armazenadas em uma planilha *Excel*, onde artigos sobre um mesmo trabalho foram agrupados em uma única linha. Esta planilha foi então utilizada para a avaliação dos resultados da revisão.

[2] http://dl.acm.org/
[3] http://www.sciencedirect.com/

3.2 Principais Sistemas de *Question Aswering*

Em (KWOK *et al.*, 2001), os autores apresentam *MULDER*, um sistema de QA baseado em técnicas de *IR* para a *Web*. O sistema usa técnicas de *NLP* e múltiplos motores de busca para obter seus resultados. Além disso, o sistema utiliza métricas de confiabilidade de respostas para garantir a qualidade da saída do sistema. Segundo seus autores, *MULDER* foi o primeiro sistema QA de propósito geral disponível na *Web*, alcançando resultados até três vezes superiores a sistemas semelhantes da época. Contudo, apesar se poder ter sido classificado como um sistema QA, *MULDER* retorna uma lista de resultados (dotados de seu grau de certeza) contendo trechos que podem conter a resposta para a pergunta emitida pelo usuário. Deste modo, *MULDER* tem a falta concisão como um de seus pontos negativos.

Em (BRILL *et al.*, 2002), os autores apresentam *AskMSR*, um sistema de QA baseado em *IR*s tradicionais com o objetivo de evitar respostas erradas. Ao invés de focar em técnicas avançadas para aumentar o número de cenários em que o sistema é capaz de responder perguntas factuais, o sistema foca em técnicas para reduzir o número de respostas incorretas. O sistema utiliza técnicas de reformulação de consultas para gerar variações válidas da consulta original e realiza o processo de busca por trechos que combinem com o padrão buscado através de um conjunto de documentos. Contudo, o sistema é capaz apenas de responder questões mais simples e curtas, não tratando consultas complexas que dependam de respostas de sub-consultas.

Em (LOPEZ *et al.*, 2007), os autores apresentam *AquaLog*, um sistema de QA portátil baseado em ontologias. O *AquaLog* é capaz de responder consultas em linguagem natural através da junção de técnicas de *NLP* e tecnologias da web semântica, sendo capaz de consultar um ou mais KBs. Além disso, o sistema também possui capacidades de aprendizado, possibilitando seu aprimoramento com o tempo de uso. O processo de *feedback* para o aprendizado é realizado através da seleção de menus na interface gráfica.

Em (BORDES *et al.*, 2014), os autores apresentam um modelo de *embedding* de palavras construído a partir de bases de conhecimento constituídas por questões(exemplos) em LN e as representações estruturadas de suas respostas. Além disso, para cada exemplo, são armazenados pares de questões alternativas equivalentes. Um *embedding* de palavras é uma representação vetorial de palavras para serem usadas como entrada em modelos de ML (GOYAL; FERRARA, 2018). O sistema gera a representação vetorial da questão do usuário e busca a questão candidata mais próxima de acordo com esta representação, utilizando métricas simples de distâncias espaciais. O sistema possui a desvantagem de necessitar que um grande *dataset*

estruturado de perguntas e respostas seja construído de antemão.

Em (RANI *et al.*, 2014), os autores apresentam uma abordagem híbrida que alia o uso de ontologias e lógica *fuzzy* para tratar o problema de recuperação de informação na *Web* de dados. O sistema utiliza ontologias *fuzzy* para representar o conhecimento e lidar com aspectos de incerteza. Para o uso de lógica *fuzzy* é proposta a utilização de uma escala *fuzzy* de dois níveis. O primeiro nivel, o *fuzzy* tipo 1, é responsável pela relação de pertencimento entre os documentos e palavras. Enquanto o segundo nível, o *fuzzy* tipo 2, é responsável pela relação de pertencimento quando há incerteza por meio do uso de sinônimos. Para tratar o problema de detecção da semântica correta de uma palavra é utilizado o conceito de co-clusterização, onde cada *cluster* representa uma interpretação semântica da palavra, esta podendo estar presente em mais de um *cluster* simultaneamente. Como desvantagens da abordagem, destaca-se o auto custo computacional das regras e *clusters fuzzy*, o que torna sua aplicação em bases de grande porte um desafio.

Em (SAHA *et al.*, 2016), os autores apresentam *ATHENA*, um sistema baseado em ontologias OWL para a realização de consultas em LN sobre *RDB*s. *ATHENA* utiliza ontologias de domínio específico como meio de representação semântica do conhecimento sobre um determinado domínio, descrevendo suas suas entidades e os relacionamentos entre as mesmas. O uso de ontologias para a interpretação de linguagem natural provê uma base semântica mais forte para desambiguação de intenção do usuário, quando comparado ao uso do esquema do banco de dados. Esta representação ontológica é construída automaticamente, tendo como base o esquema da fonte *RDB*. Como desvantagens, a representação ontológica produzida segue um conjunto de heurísticas já predefinidas, o que não pode garantir a qualidade final da ontologia produzida.

Outros exemplos de sistemas de QA podem ser vistos no *workshop TREC Tracks*[4]. Um *workshop* do *TREC* consiste em um conjunto de trilhas, áreas de foco, em que determinadas tarefas de recuperação são definidas. Estas trilhas funcionam como incubadoras para novas áreas de pesquisa.

3.3 Principais Sistemas de *Interactive Question Aswering*

Em (FLIEDNER, 2006), os autores apresentam um sistema de IQA para o idioma Alemão. O sistema apresentado trata-se da extensão de um sistema de QA já existente, o qual foi

[4] https://trec.nist.gov/tracks.html

aprimorado com o objetivo de tratar os problemas de *"anaphora"* (ou problema de correferência) e *"ellipses"*. O trabalho tem como objetivo garantir um diálogo coerente durante o uso de perguntas complementares. O sistema atua sobre um conjunto de documentos de texto não estruturados que são então pré-processados em uma etapa *offline* e armazenados em um formato estruturado de árvore de dependência junto de suas informações léxicas. Contudo, a ferramenta não considera outros aspectos semânticos além de relações super/sub-classes, *e.g.*, relações do tipo "*parte-de*" ou "*similar-ao*".

Em (KAUFMANN *et al.*, 2006), os autores apresentam *Querix*, uma interface de linguagem natural para a Web Semântica. O *Querix* atua como um IQA, tendo como objetivo prover uma interface acessível e de fácil uso para usuários leigos. *Querix* utiliza o diálogo com o usuário para realizar a clarificação da consulta, pedindo ao usuário para resolver possíveis ambiguidades. Quando a ferramenta encontra alguma possível ambiguidade ela apresenta um *menu* com uma lista de interpretações possíveis para a seleção. Contudo, apesar de usar uma ontologia no processo de interpretação da consulta, *Querix* não considera aspectos semânticos, e.g., subsunção de classes e outros tipos de relações.

Nos artigos (HARABAGIU *et al.*, 2005b; HARABAGIU *et al.*, 2005a; HICKL *et al.*, 2006), os autores apresentam *FERRET*, um sistema de IQA construído para atuar sobre uma coleção de documentos com o uso de uma técnica de interação até então nova, o *predictive questioning* (ou questionamento preditivo em tradução livre). *FERRET* busca tratar dois desafios: (1) Prever as possíveis interações futuras de acordo com processo de busca do usuário e (2) Responder questões complexas. A técnica *Predictive questioning* tenta prever novas possíveis informações relevantes para o usuário de acordo com sua pergunta original para então sugeri-las, tendo como objetivo manter o usuário engajado com o sistema. Para o tratamento de uma questão complexa, o sistema a decompõe em subquestões mais simples, respondo cada uma das subquestões individualmente e por fim combinando suas respostas. Como os demais sistemas que atuam sobre dados não estruturados *FERRET* precisa de uma etapa *offline* de preprocessamento e indexação dos documentos. Por fim, a interface de *FERRET* é semelhante a de um motor de buscas podendo apresentar uma quantidade potencialmente grande de respostas, tendo a falta de concisão como um de seus pontos negativos.

Em (OWDA *et al.*, 2007), os autores apresentam uma abordagem para a construção de interfaces de consulta em LN sobre *RDB*s. A abordagem apresentada combina técnicas de agentes conversacionais orientados a objetivos e arvores de conhecimento. O sistema tem como

objetivo prover a capacidade de desambiguação e conversação sobre o domínio para ambas, questões factuais e complexas. Para isto, é utilizada uma abordagem de casamento de padrões, onde a intenção da questão do usuário é classificada para um padrão de pergunta pré-programado pelo engenheiro de conhecimento. O sistema estrutura seu conhecimento acerca do domínio no formato de uma árvore de conhecimento. Esta árvore é utilizada pelo agente conversacional para guiar o fluxo da conversa. A árvore armazena o conjunto de questões suportadas pelo sistema de maneira hierárquica, contendo consultas *SQL* para serem executadas no *RDB* e *scripts* de diálogos para colher as informações necessárias para sua execução. Como limitações, o sistema necessita da criação manual de sua árvore de conhecimento. Além disso, o fluxo da conversação é restringindo pelos cenários modelados na árvore de conhecimento durante sua construção.

Em (QUARTERONI; MANANDHAR, 2006; QUARTERONI; MANANDHAR, 2007; QUARTERONI, 2008; QUARTERONI; MANANDHAR, 2009), os autores apresentam *YourQA*, um sistema de QA de domínio aberto com interface de diálogo baseada em *chatbots*. *YourQA* é capaz de resolver tanto questões factuais, quanto sobre definições e descrições. O sistema utiliza diálogos, implementados em *AIML*. O sistema tem como objetivo possibilitar o diálogo de clarrificação e dar suporte para perguntas de acompanhamento. O componente de QA de *YourQA* utiliza documentos da Web (páginas *HTML*) como sua base de conhecimento, recuperando as vinte páginas mais bem ranqueadas na pesquisa do *Google* para a questão do usuário. Em seguida, para cada página *Web* recuperada, analisa-as e extrai as respostas para a pergunta original, ranqueando-as por relevância. Como limitações da abordagem seguida por *YourQA*, temos: O sistema possui um fluxo de conversação limitado e rígido; o sistema não é capaz de tratar questões complexas, tais como duas questões ou mais realizadas como uma única (e.g. "*Quais são os estados do Brasil e suas capitais?*"), nestes casos o sistema limita-se a tratar apenas uma das subquestões; além disso, o sistema realiza o tratamento de perguntas complementares apenas relacionadas as perguntas, não levando em conta termos presentes em respostas. Por fim, o sistema apresenta a resposta como uma lista de trechos que contenham informações relevantes para a pergunta do usuário, pecando assim no aspecto de falta de concisão.

Em (DORNESCU; ORASAN, 2010), os autores propõem uma extensão do *framework* de QA, *QALL-ME*, para adicioná-lo recursos interativos com o objetivo de evitar respostas vazias. Para tal, os autores propõem o relaxamento de restrições através do *feedback* do usuário para a clarificação das restrições da consulta. Contudo, o sistema limita-se a utilizar

o diálogo de clarificação como mecanismo de relaxamento de restrições apenas quando uma consulta possui um conjunto de respostas vazio.

***F**eedback, **R**efinement and **E**xtended Vocabulary **A**ggregation* (*FREyA*) é uma *NLQI* para ontologias *RDF* independente de domínio. O *FREyA* utiliza técnicas de análise sintática e buscas baseadas em ontologias para interpretar e responder às perguntas do usuário. Além disso, o sistema utiliza o *feedback* do usuário como ferramenta de clarificação de ambiguidades e como mecanismo de reforço. O sistema utiliza a interação aliada a um mecanismo de aprendizado por reforço com o objetivo de solucionar possíveis ambiguidades que possam surgir durante a etapa de interpretação da consulta, aprendendo a solucioná-las automaticamente em situações futuras (DAMLJANOVIC *et al.*, 2010; DAMLJANOVIC *et al.*, 2011). Durante estágios iniciais de uso da ferramenta, *FREyA* pode exigir a interversão constante do usuário para a clarificação. Contundo, com o passar do tempo o sistema pode ajustar-se, aprendendo a solucioná-las automaticamente, podendo inclusive aprender conceitos novos não presentes explicitamente na ontologia.

OntBot, apresentado em (AL-ZUBAIDE; ISSA, 2011), é um *chatbot* baseado em ontologia. *OntBot* responde perguntas em LN sobre a ontologia de domínio em uma maneira conversacional, dando suporte a perguntas complementares. O sistema armazena a ontologia como um banco de dados relacional, gerando automaticamente o esquema da base relacional e os mapeamentos da ontologia para o novo meio de armazenamento. Com isto, o sistema gera automaticamente padrões de questões com o uso de *"rules"* pré-definidas que utilizam o esquema relacional para gerar padrões de perguntas em LN suportadas pelo sistema. Estas *rules* podem ser estendidas pelo desenvolvedor. Além disso, o sistema utiliza técnicas de *NLP* para normalizar a entrada que é então classificada para um dos padrões gerados. Por fim, são aplicadas técnicas de *NLG* para gerar a resposta a ser apresentada ao usuário. Como principal limitação, o sistema apresenta a necessidade da representação intermediária da fonte de conhecimento, além de necessitar da criação de regras construídas a mão para sua customização.

O *Context-aware Attention Network* (*CAN*), apresentado em (LI *et al.*, 2017), é um modelo *encoder-decoder* de ML projetado para executar tarefas de QA. *CAN* utiliza três diferentes *GRUs* (CHO *et al.*, 2014) aliadas a um mecanismo de dois níveis de atenção. Além disso, *CAN* possui a capacidade de gerar automaticamente perguntas de clarificação, caso as informações dadas como entrada não sejam suficientes para que o modelo alcance uma resposta, iniciando o processo interativo de clarificação. No entanto, o modelo está limitado ao uso de uma única questão suplementar, o que limita a interação a uma única rodada. Além disso, também

exige que todas as sentenças contenham carácteres de *End of File* (*EOF*), *e.g., "." e "?"*. Por fim, como os demais modelos de ML, *CAN* pode cair no problema da escassez de dados bem formatados e corretos para a geração da resposta.

(KUMAR; JOSHI, 2017) apresentam um sistema *seq2seq* para tratar perguntas complementares em ambientes de IQA e *Intelligent personal assistants* (IPAs). O sistema apresentado utiliza técnicas de aprendizado *sequence-to-sequence* (SUTSKEVER *et al.*, 2014) baseado em recuperação (KANNAN *et al.*, 2016), tendo como entrada uma pergunta complementar incompleta IQ1(sofrendo de *Anaphora* ou *Ellipses*) e o par de (Pergunta completa ($Q1$), Resposta($A1$)) anterior. O sistema produz como resultado uma pergunta completa Q_2 equivalente a IQ_1, tratando este problema como uma tarefa de tradução. Um dos principais pontos positivos do sistema apresentado é a sua capacidade de reuso, podendo ser adicionado em diversos outros sistemas como um módulo *plug-in*, tendo sido testado em uso com o IPA Sire da *Apple* e alcançando uma melhora no desempenho do sistema de 131.57% ao responder perguntas complementares incompletas. O sistema apresentado tem como principal limitação a necessidade de um *corpus* rotulado bem formatado. Contudo, o sistema atenua tal limitação com o uso de buscas em uma biblioteca de padrões, traduzindo automaticamente a pergunta de entrada (tanto na fase de treino, quanto na de execução) para um padrão que represente a estrutura geral da pergunta. Outra limitação do sistema é o fato de aparentemente atuar apenas com o último par de pergunta-resposta completos.

Em (ATHREYA *et al.*, 2018) é apresentado *DBpedia CHATBOT*, um *chatbot* multiplataforma capaz de responder perguntas factuais geralmente feitas na comunidade *DBpedia*. Além disso, o *DBpedia CHATBOT* é capaz de responder também perguntas do cotidiano. Para alcançar uma solução mais cômoda o *chatbot* é capaz de interagir com o usuário por meio links e botões para a sugestão de novas informações, além do recolhimento do *feedback* do usuário sobre a utilidade da informação apresentada. O sistema tem como sua principal desvantagem depender do casamento de padrões construídos manualmente, o que pode demonstrar-se insuficiente quando lidando com situações de mundo real e a variabilidade da LN. Como outra desvantagem, o sistema restringe-se a responder questões factuais simples, não tratando, por exemplo, questões como relacionamentos entre entidades, o que reduz o escopo de perguntas solucionáveis.

3.4 Métodos Para a Avaliação de Sistemas de QA e IQA

Por serem híbridos entre sistemas de QA e DS, os sistemas de IQA podem ser avaliados de acordos com dois tipos de métricas: (1) Com o uso de métricas quantitativas de recuperação de informação (VOORHEES, 2003); (2) Com o uso de métricas qualitativas de satisfação do usuário (WALKER *et al.*, 2000).

O primeiro tipo é herdado de sistemas de QA e tenta medir o quão bem o sistema é capaz de responder perguntas. Alguns exemplos de métricas quantitativas são:

- **Precisão (*Precision*):** Fração de instâncias relevantes entre as instâncias retornadas pelo sistema. De maneira intuitiva, esta métrica afere a taxa de respostas corretas dadas pelo sistema (LEARN, 2019b). A precisão é definida da seguinte forma:

$$\text{Prec(Q)} = \frac{\text{Número de respostas corretas}}{\text{Número de questões}} \tag{3.1}$$

- **Revocação (*Recall*):** Fração de instâncias relevantes retornadas sobre o total de instâncias relevantes (LEARN, 2019c). De maneira intuitiva, esta métrica afere a frequência em que o sistema consegue responder as questões. O *recall* é definido da seguinte forma:

$$\text{Rec(Q)} = \frac{\text{Número de respostas corretas}}{\text{Número de questões respondíveis}} \tag{3.2}$$

- ***F-measure* (*F1-score*):** Média harmônica da precisão e do *recall*. Esta métrica dá um maior peso para valores baixos, deste modo, caso a precisão esteja muito elevada, mas o *recall* esteja muito baixa (e vice-versa), o valor da *F-measure* será menor do que a média simples esperada (LEARN, 2019a). A *F-measure* é definida da seguinte forma:

$$\text{F-score(Q)} = 2 * \frac{\text{Prec(Q) * Rec(Q)}}{\text{Prec(Q) + Rec(Q)}} \tag{3.3}$$

No segundo tipo, para a avaliação qualitativa, podem ser aplicados formulários que avaliem o nível de satisfação do usuário. As questões do formulário podem variar para cada estudo, tendo como objetivo avaliar pontos críticos para cada projeto. Em (QUARTERONI; MANANDHAR, 2009), os autores propõem um questionário contendo oito questões genéricas com o objetivo de avaliar o grau de satisfação do usuário para sistemas de IQA. Além de formulários, os registros de *logs* de conversas ou de *feedback* podem ser utilizados para medir o nível de satisfação do usuário.

As principais abordagens de avaliação apresentadas na literatura foram: (1) testes automatizados de *gold standard* e (2) avaliação baseada em tarefas. Enquanto a primeira é costumeiramente empregada em sistemas de QA tradicionais, a segunda é costumeiramente utilizada em sistemas interativos como os DS.

Na primeira, são realizadas um conjunto de perguntas para as quais já são conhecidas as respostas corretas que serão comparadas com as respostas retornadas pelo sistema. Esta abordagem não pode avaliar critérios qualitativos por conta de seu caráter automatizado. Em (DAMLJANOVIC *et al.*, 2011), os autores optaram por utilizar *datasets* para *benchmark* de sistemas QA disponibilizados no desafio QALD[5]. Em (KUMAR; JOSHI, 2017), os autores utilizaram BLE U(PAPINENI *et al.*, 2002), um método para avaliação automática de *machine translate*. Em (KAUFMANN *et al.*, 2006) e (DAMLJANOVIC *et al.*, 2010), os autores optaram por usar os *datasets* Mooney Geoquery (TANG; MOONEY, 2001), contendo consultas geo-espaciais. Em (LI *et al.*, 2017), os autores apresentam, *ibAbI*, um *dataset* para sistemas IQA baseado no *dataset* de QA do *Facebook*, bAbI (WESTON *et al.*, 2015).

Na avaliação baseada em tarefas, participantes humanos são convidados para realizar tarefas de recuperação de informação com o uso do sistema avaliado (KONSTANTINOVA; ORASAN, 2013). No fim, é calculado o grau de êxito do participante ao recuperar as respostas corretas para cada tarefa. Tal avaliação pode também contar com a apresentação de um questionário de satisfação ou registrar o número de interações, tempo demorado e quantidade de vezes em que o sistema precisou ser reiniciado para avaliar os aspectos qualitativos do sistema. Os trabalhos (QUARTERONI; MANANDHAR, 2009; HARABAGIU *et al.*, 2005a) adotaram variações desta abordagem.

3.5 Discussões Sobre os Resultados

A Tabela 3 apresenta um resumo dos trabalhos sobre sistemas de QA e IQA apresentados nesta revisão. Nela são apresentadas informações sobre se o sistema possui suporte a iteração multi-turno, caso sim, seus recursos interativos disponibilizados; o tipo de fonte de conhecimento consulta; e se a resposta apresentada ao usuário é concisa ou não. A coluna "*Fontes*" representa a fonte de conhecimento utilizada pelo sistema, podendo esta ser:

- ***RDB*:** Utilizam bancos de dados relacionais como fonte de conhecimento.
- **IR:** Utilizam sistemas de *IR* tradicionais,e.g., *KIR*s, como fonte de conhecimento. Tais

[5] http://qald.aksw.org/

sistemas operam sobre coleções de documentos (textos não estruturados) buscando documentos ou trechos que sejam relevantes para a pergunta do usuário;

- ***RDF*:** Utilizam bases *RDF* como fonte de conhecimento. Estes sistemas traduzem uma consulta Q em LN para uma consulta Q' em SPARQL que a represente;
- **ML:** Utilizam modelos de ML treinados sobre uma fonte de dados para responder perguntas. A fonte utilizada para treinamento pode ser ambas, estruturada ou não, podendo trabalhar sobre conjuntos de perguntas e respostas (conjunto rotulado) ou textos não estruturados (conjunto não rotulado).

Os recursos suportados pelos sistemas fazem referência aos apresentados na Seção 2.2.8, onde cada coluna significa:

- **Conciso:** Indica se o sistema apresenta a resposta direta para questão do usuário ao invés de trechos que a contenham;
- **Feedback:** O sistema aprende com o *feedback* retornado pelo usuário. Nos casos de sistemas de QA tradicionais, isto é feito pela seleção de opções em menus;
- **Clarificação:** A possibilidade do sistema efetuar a realização de diálogo de clarificação para solucionar possíveis ambiguidades na interpretação da questão;
- **Complemento:** A possibilidade do sistema tratar perguntas complementares (lidando com os problemas de *Anaphora* e *Ellipses*), relacionando-as com o histórico da conversa;
- **Recomendação:** A possibilidade do sistema recomendar novas informações (diferentes das questionadas) adicionais que possam ser de interesse do usuário;
- **Complexas:** A possibilidade do sistema responder perguntas complexas, podendo dividir estas em múltiplas questões factuais respondidas individualmente, ou podendo solicitar parâmetros ausentes para a resolução da questão.

Assim como afirmado em (KONSTANTINOVA; ORASAN, 2013), nesta revisão também não foi possível identificar uma solução padrão para este tipo de sistema. Estes sistemas podem variar em uma ampla gama de aspectos, tais como o tipo de fonte de conhecimento utilizada e os recursos disponibilizados. Como é possível visualizar na Tabela 3, cada sistema tenta priorizar um aspecto diferente dos recursos interativos. Nenhum dos sistemas analisados tentou tratar todas as oportunidades disponíveis para sistemas de IQA, onde a maioria destes sistemas focou em um ou dois recursos interativos.

O diálogo de clarificação foi o recurso mais explorado dentre aqueles listados. Isto pode ser por que sistemas de IQA serem tratados como evoluções de sistemas de QA tradicionais.

Tabela 3 – Resumo de trabalhos de *Natural Language Query Interface*

Trabalhos	Interativo	Base	Conciso	Feedback	Clarificação	Complementação	Recomendação	Complexas
(KWOK *et al.*, 2001)	N	IR	N	N	N	N	N	N
(BRILL *et al.*, 2002)	N	IR	N	N	N	N	N	N
(FLIEDNER, 2006)	S	IR	S	N	N	S	N	S
(KAUFMANN *et al.*, 2006)	S	RDF	S	N	S	N	N	N
(HARABAGIU *et al.*, 2005b; HARABAGIU *et al.*, 2005a; HICKL *et al.*, 2006)	S	IR	N	N	N	N	S	S
(LOPEZ *et al.*, 2007)	N	RDF	S	S	N	N	N	N
(OWDA *et al.*, 2007)	S	RDB	S	N	S	S	N	S
(QUARTERONI; MANANDHAR, 2006; QUARTERONI; MANANDHAR, 2007; QUARTERONI, 2008; QUARTERONI; MANANDHAR, 2009)	S	IR	N	N	S	S	N	N
(DORNESCU; ORASAN, 2010)	S	RDF	S	N	S	N	N	N
(DAMLJANOVIC *et al.*, 2010; DAMLJANOVIC *et al.*, 2011)	S	RDF	S	S	S	N	N	S
(AL-ZUBAIDE; ISSA, 2011)	S	RDF	S	N	N	S	N	S
(BORDES *et al.*, 2014)	N	ML	S	N	N	N	N	S
(RANI *et al.*, 2014)	N	RDF	S	N	N	N	N	N
(SAHA *et al.*, 2016)	N	RDB + RDF	S	N	N	N	N	S
(LI *et al.*, 2017)	S	ML	S	N	S	N	N	N
(KUMAR; JOSHI, 2017)	S	ML	N	N	N	S	N	N
(ATHREYA *et al.*, 2018)	S	RDF	S	S	N	N	S	N

Deste modo, os sistemas só recorrem ao uso da interação quando o sistema não é capaz de responder a questão inicialmente proposta, recorrendo ao diálogo de clarificação.

Por outro lado, a adaptação com base no *feedback* e a recomendação de novas informações foram os recursos menos explorados nos trabalhos explorados. Estes recursos, recursos estão intimamente ligados aos DS, onde o sistema passa a possuir uma maior consciência contextual (lembrando e adaptando-se ao fluxo da conversação) e uma maior proatividade (buscando antecipar-se e manter o diálogo). Esta negligência reforça a visão de que sistemas de IQA são apenas uma evolução dos QAs tradicionais, o que limitar o rico potencial que este tipo de sistema possui. Isto mostra que os desafios elencado por (HICKL *et al.*, 2006) para sistemas de IQA ainda não estão sendo encarados pela comunidade de pesquisa.

Ainda com relação aos recursos interativos adotados pelos sistemas de IQA, por mais que a resolução de perguntas complexas apareça como um ponto tratando em uma grande parte dos trabalhos avaliados, este ainda é um grande desafio da área. A maioria dos sistemas analisados tratam este ponto de maneira superficial, delegando-o para o motor de busca subjacente, como em (FLIEDNER, 2006; HARABAGIU *et al.*, 2005b; HARABAGIU *et al.*, 2005a; HICKL *et al.*, 2006), ou dependendo do uso de *templates*, ou *datasets* rotulados como em (OWDA *et al.*, 2007; AL-ZUBAIDE; ISSA, 2011; BORDES *et al.*, 2014). Enquanto a primeira estratégia foge do controle do desenvolvedor, a segunda exige um maior esforço para a construção da base de conhecimento. No entanto, segundo (DIEFENBACH *et al.*, 2018), o uso de sistemas baseados em padrões parece ser a alternativa mais indicada atualmente para a resolução de questões complexas.

Quando considerando os métodos de avaliação de sistemas de IQA empregados, por conta da participação do usuário durante o processo interativo, estes sistemas requerem avaliações qualitativas que meçam o nível de satisfação do usuário com relação ao uso da ferramenta. Contudo, a grande maioria dos sistemas limitam suas avaliações a critérios quantitativos, assim como sistemas de QA tradicionais.

Ao analisar as Questões de Pesquisa definidas na seção 3.1.1, é possível identificar as seguintes respostas:

- ***QP1*:** Por volta de 64% dos trabalhos analisados contavam com a capacidade de interação multi-turno. Esta medida se traduz em 11 trabalhos únicos (não contando continuações). No entanto, esta revisão teve um foco especial em sistemas de IQA, daí sua maior quantidade comparada com a de QAs tradicionais, no entanto, este número é muito menor do que

os cerca de 30 analisados em (DIEFENBACH *et al.*, 2018), onde o autor aborda apenas sistemas de *Knowledge Base Question Answering* (KBQA).

- ***QP2*:** Como citado anteriormente, o diálogo de clarificação é o recurso mais explorado, onde cerca de 35% dos trabalhos analisados implementam este recurso. Os demais números são 35% para questões complexas, 29% para questões complementares, 17% para adaptação ao *feedback* e 11% para a recomendação de informação.
- ***QP3*:** Na Figura 3 é possível ver o histograma dos tipos de fontes adotadas pelos sistemas analisados. É possível verificar que as fontes *RDF* foram as mais utilizadas, seguidas pelos sistemas de *IR* já existentes. Isto demostra que as fontes *RDF*, e.g., EKGs, são uma boa alternativa para darem suporte a tarefa de QA.
- ***QP4*:** Nos trabalhos analisados, cerca de 70% apresentam respostas concisas e diretas para as perguntas dos usuários. A maioria daqueles que não o fazem, são baseados em sistemas de *IR* já consolidados, o que demostra que tais sistemas são limitados por seus motores de busca.
- ***QP5*:** Assim, como identificado em (HICKL *et al.*, 2006), a construção de sistemas adaptáveis e proativos continua sendo um desafio para sistemas de IQA. Além disso, os cenários em que tais sistemas são capazes de responder questões complexas ainda são bastante limitados, em geral, sendo restritos a sistemas de domínios específicos e baseados e padrões. Por fim, a grande maioria dos sistemas são caracterizados como sistemas monólitos, onde há pouco ou nenhum reuso de soluções já existentes, havendo a necessidade da implementação do sistema do "zero", o que consome tempo e recursos.

3.6 Trabalhos Relacionados

Em (ABUJABAL *et al.*, 2017) os autores apresentam uma abordagem para a geração automática dos padrões de consulta suportados por sistemas de TBQA. O sistema tem como entrada um conjunto de pares de questões em linguagem natural e suas respostas. Para a geração dos padrões, o sistema utiliza um *lexicon* para a obter a árvore de dependência da questão que é então utilizada para construir heuristicamente uma consulta SPARQL que capture sua resposta dada como entrada. Esta consulta é então generalizada para uma padrão ao mapear conjuntos de questões para uma mesma consulta. Como uma vantagem, a abordagem apresentada permite a composição de padrões para a resolução de consultas complexas para as quais não são conhecidos padrões completos. Como limitação, a abordagem depende da qualidade do *lexicon* utilizado

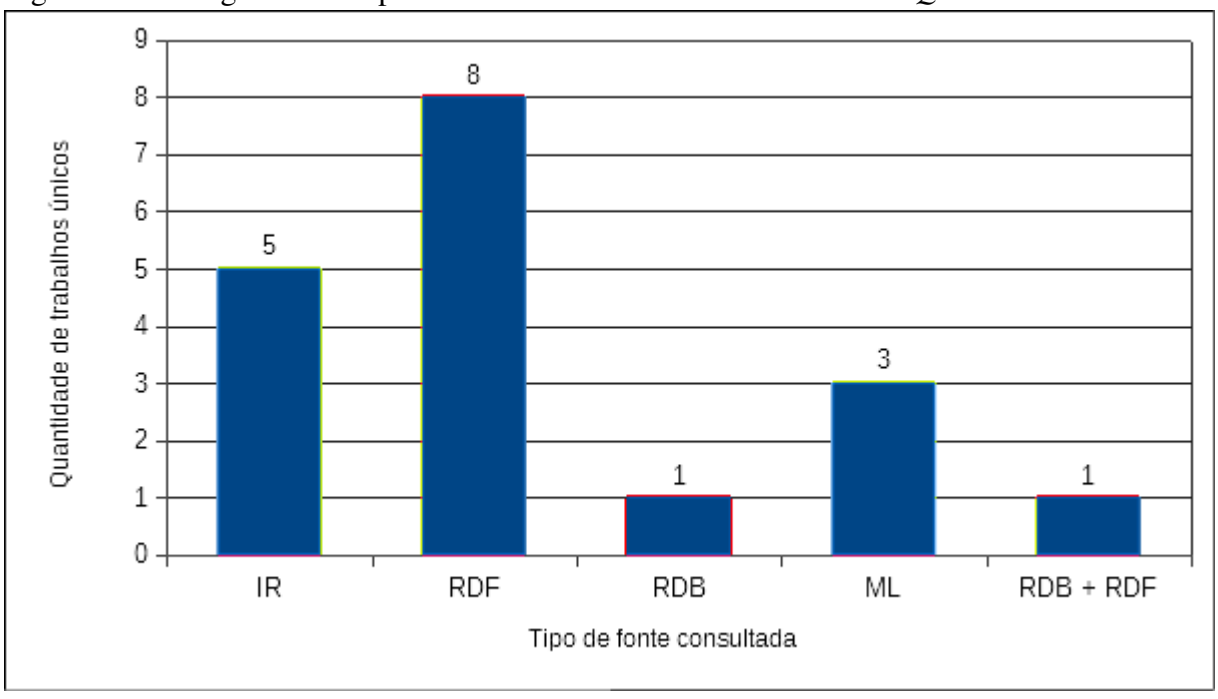
para a maior cobertura de padrões, podendo haver a necessidade da extensão do *lexicon* para domínios específicos. Além disso, o sistema também não permite ao usuário o controle dos padrões suportados pelo sistema. Por fim, os autores não discutem como o sistema pode ser disponibilizado para os usuários, indicando que isto deve ser tratado em cada caso específico.

Em (ZHENG *et al.*, 2018) os autores apresentam uma nova abordagem para interpretação de questões em linguagem natural pelo uso de padrões. Estes padrão são gerados automaticamente tendo como entra o KG da *DBpedia*[6], junto de suas páginas correspondentes da *Wikipedia*[7] contendo os dados como texto em linguagem natural. Os padrão gerados nesta fase são compostos por um padrão de questão em linguagem natural, junto do padrão de triplas que o representa. Durante o tempo de execução da consulta, a abordagem decompõe a questão de entrada em padrões de questões simples gerados numa etapa offline. Estes padrões são então compostos em uma representação da questão original, realizando a desambiguação nos níveis de entidades e estrutural. Esta representação é então utilizada para gerar uma consulta SPARQL que representa questão inicial. Como desvantagem da abordagem, esta necessita a entrada de um conjunto de *corpus* que representam os fatos registrados no KG, o que pode não estar disponível em ambientes empresarias.

Em (BIERMANN *et al.*, 2018) os autores apresentam um sistema de TBQA sobre

[6] https://wiki.dbpedia.org/
[7] https://www.wikipedia.org/

Figura 3 – Histograma dos tipos de fontes adotadas em sistemas de *NLQI*



Fonte: Próprio autor.

KGs que gera automaticamente questões suportadas com base no KG subjacente. O processo de construção das questões suportadas é realizado com base é um pequeno conjunto de padrões de consultas definido pelos autores (e.g., padrões que seguem a estrutura básica de uma tripla, questões iniciadas de um verbo seguido por uma locução prepositiva, questões de contagem, etc.). O sistema constrói então as questões suportadas para cada um dos padrões pré-definidos, gerando variações destas mesmas (e.g., substituídos palavras por sinônimos). Essas questões são então armazenadas num índice que é consultado em tempo de execução para identificar a questão mais provável sendo perguntada pelo usuário. Além disso, o sistema permite a construção interativa de consultas com a função de autocompletar os padrões. Como desvantagem, a abordagem descrita não permite ao desenvolvedor ter controle das questões suportadas pelo sistema, o que dificultaria a implementação do suporte às QCs e questões relevantes específicas para cada empresa. Além disso, os autores não abordam como o sistema pode ser disponibilizado em diferentes canais da demo apresentada, o que dificulta a portabilidade desta.

Em (COCCO *et al.*, 2019) os autores apresentam um sistema de QA baseado em padrões sobre o domínio de gastos públicos. A abordagem apresentada utiliza ML para o aprendizado de padrões SPARQL generalizados de um conjunto existente de pares de questão em linguagem natural e sua respectiva consulta SPARQL. Para a classificação da questão do usuário, é utilizado um classificador baseado em instância para associar a questão à um padrão que chegue mais próximo de representá-la. O trabalho ainda encontra-se em estágio iniciais, no entanto apresenta resultados promissores para o campo de aprendizado de padrões genéricos com base em um pequeno *dataset* construído manualmente.

Em (AVILA *et al.*, 2019a) os autores apresentam *MediBot*, um *chatbot* para a realização de consultas em linguagem natural sobre um grafo de conhecimento no domínio de medicamentos. *Medibot* possui dois modos de operação, onde o primeiro trata-se de um TBQA, onde são utilizadas expressões regulares para classificarem o *template* em que a questão do usuário se encaixa. A abordagem seguida pelos autores depende da implementação manual das expressões regulares e do código de consulta e construção das respostas, o que dificulta o reuso e a aplicação da abordagem para *chatbots* como grandes números de padrões suportados. Além disso, o código de fluxo de conversa do *chatbot* está profundamente ligado a interface de interação do *Telegram*, o que dificultaria o porte do *chatbot* para outros canais.

Muitos dos trabalhos existentes na área de TBQA concentram-se na geração automática de padrões, no entanto, tais abordagens limitam o controle do desenvolvedor sobre

as questões suportadas pelo sistema. Estas abordagens tentam aumentar a cobertura da questões suportadas pelo sistema, o que é um aspecto positivo no contexto de consulta na Web. Contudo, em ambiente empresariais é esperado que as consultas realizadas sejam limitadas a um conjunto limitado e específico de consultas para a realização das atividades da empresa, onde neste contexto de de suma importância que este conjunto seja completamente coberto e de maneira correta. Deste modo, *CONQUEST* garante que o desenvolvedor tenha o total controle sobre o conjunto de padrões suportados, garantindo a corretude das consultas que os respondem. Além disso, a maioria dos sistemas não abordam como o serviço de TBQA possa vir a ser disponibilizado aos usuários, deixando para o desenvolvedor a tarefa de personalizar ou criar do zero os mecanismos de acesso aos sistemas. *CONQUEST* lida com isso ao reusar serviços de mensageiros instantâneos (e.g., *Telegram*) como canal de acesso ao *chatbot*, além de disponibilizar o acesso ao serviço através de uma *API REST* acessível através de requisições HTTP, tudo a partir da execução de uma única instância do *chatbot*.

Como outra de suas características marcantes, *CONQUEST* utiliza o *feedback* recolhido durante o diálogo de clarificação para se adapta ao uso. Isto é realizado nos casos onde o *chatbot* não consegue classificar com precisão a intenção do usuário. Isto pode acontecer em dois cenários (considerando que a questão realizada é respondível):

1. Nenhum *template* alcança um grau de confiança mínimo (limiar de confiança). Neste cenário, a questão dada como entrada pode ser uma variação completamente diferente daquelas utilizadas como conjunto de treinamento, caindo assim no problema da variação linguística. Neste caso, o *framework* adiciona o novo padrão de questão ao seu conjunto de treinamento, treinando o classificador de maneira automática. Deste modo, o *framework* é capaz de adapta-se a novas formas de uma mesma pergunta, antes não conhecidas durante o desenvolvimento. Esta característica também permite que o conjunto de treinamento inicial seja reduzido, pois, este vai ampliando-se e tornando-se mais variável com o uso. Isto ocasiona a redução do tempo necessário para o desenvolvimento do *chatbot*;
2. O *template* mais bem classificado possui um nível de confiança muito próximo aos demais (considerando apenas *templates* com um nível de confiança maior que o limiar). Neste cenário, dois ou mais *templates* podem ser intrinsecamente semelhantes, tornando a distinção entres estes uma tarefa complexa. Para solucionar esta ambiguidade o *chatbot* solicita que o usuário selecione sua intenção de maneira explícita, dentre os *templates* candidatos. Com isto, o *chatbot* reforça um exemplo de questão a um padrão específico,

aumento a confiança da classificação neste (por consequência, diminuindo nos demais). Deste modo, é esperado que com o tempo o *chatbot* adquira uma quantidade de exemplos o suficiente para reforçar a classificação correta.

4 *FRAMEWORK CONQUEST*

Neste capítulo é apresentado *CONQUEST*, um *framework* para a construção de *chatbots* de TBIQA para EKGs.

4.1 Processo Para a Construção de Chatbots de TBIQA

O processo de construção de um sistema de IQA pode variar grandemente dependo do seu domínio, ferramentas já existentes e seu propósito (KONSTANTINOVA; ORASAN, 2013). Deste modo, torna-se necessário que o desenvolvedor projete e implemente do zero os componentes para este tipo de sistema. Neste trabalho, é proposto um *workflow* padrão para o processo de criação de sistemas de TBIQA baseados em padrões para KGs (em especial, EKGs). O *workflow* proposto dá-se a seguir:

1. **Construção dos padrões de questões suportados pelo sistema:** Nesta etapa é realizado o processo de construção de padrões de questões suportadas pelo sistema, onde um padrão é composto por no mínimo um exemplo de padrão da questão em LN, junto da consulta SPARQL que responde a questão sobre o KG;
2. **Construção do mecanismo de processamento de linguagem natural:** Nesta etapa é construído o mecanismo capaz de receber uma questão em LN, transformando-a em uma representação interpretável pelo sistema. Este mecanismo realiza a análise da questão com base em características puramente sintáticas, realizando a normalização do texto, segmentação da questão em sentenças, reconhecimento de entidades nomeadas e a criação da representação computável da questão;
3. **Construção do mecanismo de classificação de uma questão para um dos padrões:** Nesta etapa é construído o mecanismo que recebendo a representação da questão gerada na etapa anterior e a classifica para o padrão que a representa. Este mecanismo pode gerar como saída apenas um único padrão que melhor representa a intenção da questão, ou uma lista de probabilidades para cada um dos padrões suportados pelo sistema;
4. **Definição do fluxo de interação do sistema:** Nesta etapa deve ser definido como o *chatbot* comporta-se em diferentes cenários. Deste modo, deve ser tratado como o *chatbot* comportar-se quando a classificação da questão é inconclusiva ou se faltam parâmetros para a resolução do padrão;
5. **Construção do mecanismo de consulta às fontes de dados:** Nesta etapa deve ser cons-

truído o mecanismo que, dado uma padrão de consulta SPARQL, junto dos parâmetros passados pelo usuário, consegue construir uma consulta SPARQL que de fato responde a questão do usuário. Além disso, deve ser tratado como esta consulta é executada sobre o KG e como seu resultado é retornado para o usuário;

6. **Construção da interface de interação com o usuário:** Nesta etapa é construído o mecanismo que permite o acesso do usuário aos serviços disponibilizados pelo *chatbot* de TBIQA. Nesta etapa é tratado como o sistema irá interagir e apresentar os resultados das consultas ao usuário.

4.2 *Visão Geral do Framework CONQUEST*

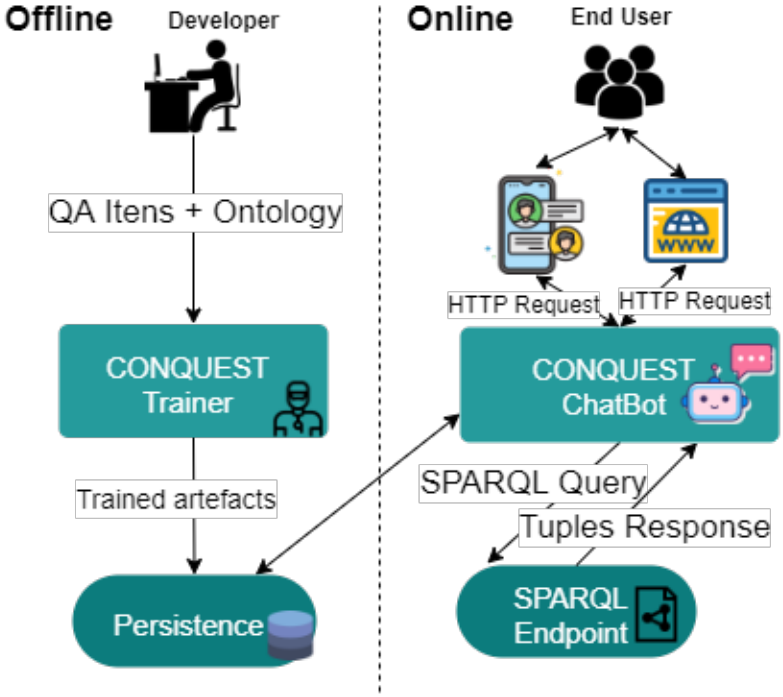
Das etapas apresentadas na Seção 4.1 apenas a primeira é específica para cada aplicação. As demais são etapas genéricas e repetitivas. Portanto, estas (etapas 2-6) tornam-se alvos para estratégias de reuso.

Tendo isto em vista, neste trabalho é apresentado *CONQUEST*, um *framework* para construção de chatbots de TBIQA para EKGs. *CONQUEST* automatiza grande parte do *workflow* de desenvolvimento deste tipo de sistema, tratando automaticamente dos passos 2-6. Assim, delegando para o desenvolvedor apenas a tarefa de construção dos padrões de questões a serem respondidos. Isto economiza tempo e recursos na hora do desenvolvimento de tais sistema, podendo democratizar e impulsionar sua adoção pela comunidade. O código fonte do *framework* pode ser baixo pelo repositório do github[1].

O *framework CONQUEST* é dividido em duas etapas distintas, como apresentado na Figura 4, sendo estas a etapa **offline** e a etapa **online**. Durante a etapa *offline* é executado o processo de "treinamento" do *chatbot*, sendo esta equivalente à etapa de construção do sistema de IQA. Já durante a etapa *online* um servidor *Web*, junto das *API*s de comunicação, como os serviços de IMs (e.g. Telegram), são executados atentando às requisições de questões.

[1] https://github.com/CaioViktor/CONQUEST

Figura 4 – Arquitetura do *framework CONQUEST*.



Fonte: Próprio autor.

Os *chatbots* produzidos por *CONQUEST* durante a etapa *offline* e executados durante a etapa *online* são capazes de automaticamente engajarem-se em diálogos com o usuário. Tal diálogo pode ser utilizado para desambiguar a intenção do usuário ou para a solicitação de parâmetros ausentes na questão. O primeiro caso é realizado quando o sistema não alcança uma alta confiança na classificação da questão para um dos padrões. Já o segundo caso é realizado quando o padrão selecionado exige parâmetros não informados na questão dada pelo usuário (e.g., parâmetros que serão utilizados como filtros em uma consulta).

Além disso, *CONQUEST* utiliza um mecanismo baseado em ML para a classificação de questões. O sistema armazena o histórico de uso, utilizando as questões realizadas como um novo conjunto de treinamento para o modelo. Isto permite que o sistema adapte-se ao uso, passando a dispensar o uso do diálogo de desambiguação. Ademais, tal abordagem permite o aprendizado de novas variações para as questões, atacando o problema da variabilidade linguística.

4.3 Fase de Treinamento *Offline*

Durante a etapa offline, o *framework* interage apenas com o desenvolvedor, isto através do componente *CONQUEST Trainer* (Figura 6). Durante esta etapa, o único artefato

concernente à tarefa de QA a ser produzido manualmente pelo desenvolvedor é o conjunto de *QAI* suportados pelo sistema.

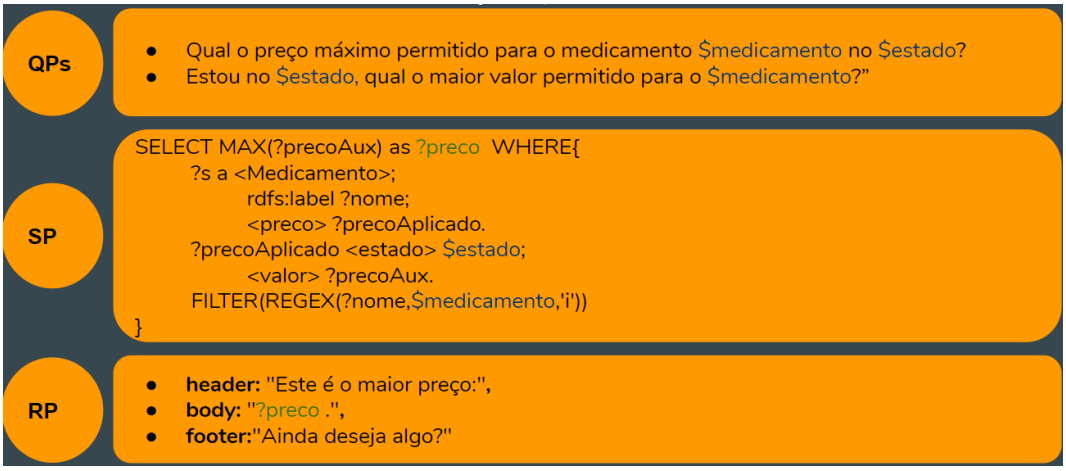
Um *QAI* representa um padrão (*template*) de questão cujo sistema é capaz de responder, onde cada *QAI* possui *slots* que serão preenchidos com informações da questão do usuário, as chamadas variáveis de contexto (*Context Variables* (*CV*)). Um *QAI* é formalmente definido como:

$$QAI_{01} = ([QP_1, QP_2, ..., QP_n], SP, RP)$$

– QP_k: Padrão de Questão (*Question Pattern* (*QP*)) em LN associado à questão. Onde $1 \leq k \leq n$.
– *SP*: Padrão de Consulta SPARQL (*SPARQL query Pattern* (*SP*)) utilizado para consultar as informações no KG.
– *RP*: Padrão de Resposta (*Response Pattern* (*RP*)) em LN apresentado ao usuário.

Na Figura 5 é apresentando um exemplo de *QAI* que representa o padrão de consulta "Busque preço de venda máximo permitido para uma medicamento em uma localização específica".

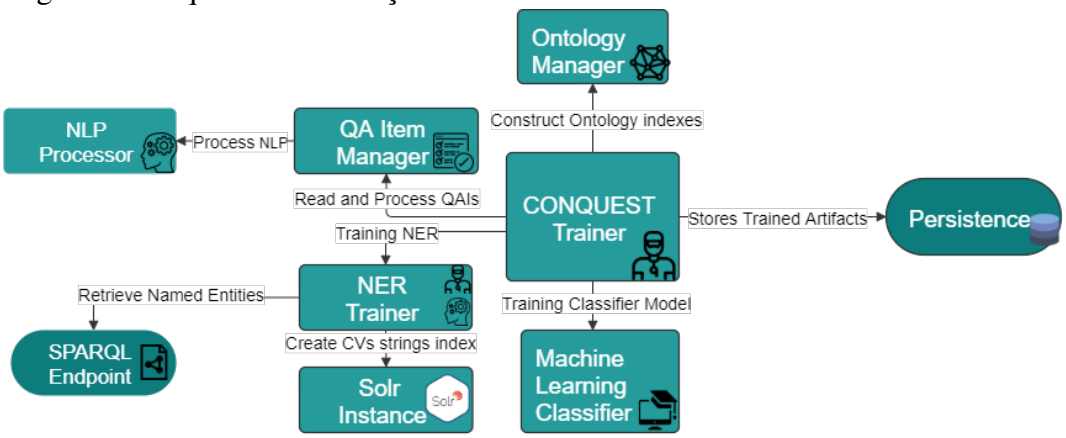
Figura 5 – Exemplo de *QAI*.



Fonte: Próprio autor.

O componente *CONQUEST Trainer* tem como entrada um conjunto de *QAI*s e a Ontologia que formam a base de conhecimento do *chatbot*. Este componente é responsável por "treinar" o *chatbot* para executar a tarefa de TBIQA, armazenando os artefatos "aprendidos" no componente *Persistence* que atuará como a "memória" do *chatbot*.

Figura 6 – Esquema de execução da fase Offline.



Fonte: Próprio autor.

O processo executado durante a fase *offline* pode ser visto na Figura 6. Resumidamente, inicialmente são construídos índices para o esquema da ontologia dada como entrada, estes serão utilizados como suporte para as etapas posteriores. Em seguida, os *QAI*s passados como entrada são lidos e processados, durante este processo é realizada a interpretação semântica das *CV*s, além do processamento das representações vetoriais dos padrões de *QP*s (Com o uso do *NLP Processor*). Dando continuidade, o próximo passo é o treinamento do mecanismo de reconhecimento de entidades nomeadas (*NER*), esta etapa é utilizada para reconhecer possíveis valores para as *CV*s dadas em uma questão. Em seguida, é realizado o treinamento do modelo de classificação de questões, sendo este responsável por classificar uma questão em linguagem natural para um *QAI*. Por fim, os artefatos aprendidos durante esta fase são armazenados na persistência para o seu uso durante a fase *online*.

Nas seções a seguir, as ações executadas durante a fase *offline* são apresentadas em mais detalhes.

4.3.1 Construção de índices

Durante esta etapa são construídos dois índices distintos, o índice de classes e o índice de propriedades. Cada um destes índices possui informações sobre o esquema da Ontologia de domínio sendo consultada. Estes índices são de fundamental importância para as etapas que se seguem, pois, estes dão suporte a interpretação semântica de uma consulta SPARQL *SP* (Seção 4.3.2.2); a descoberta de candidatos para o preenchimento de uma *CV* (Seção 4.3.3); a construção de mensagens de solicitação de *CV*s; construção do *Context Vector* (*CVec*) de uma questão; dentre outras atividades que necessitam de conhecimento sobre o domínio.

Para cada uma das classes definidas na ontologia de domínio, o índice de classes possui informações sobre sua *URI*, lista de propriedades em que esta é o domínio (*rdfs:domain*), lista de propriedades em que esta é o contra-domínio (*rdfs:range*), lista de sub-classes, lista de super-classes, lista de nomes conhecidos (*rdfs:label, skos:preferredLabel, dc:title*) e a sua lista de comentários e definições (*rdfs:comment*).

Para cada uma das propriedades definidas na ontologia de domínio, o índice de propriedades possui informações sobre sua *URI*, lista de classes em que esta possui em seu domínio (*rdfs:domain*), lista de classes em que esta possui em seu contra-domínio (*rdfs:range*), lista de nomenclaturas (*rdfs:label, skos:preferredLabel, dc:title*) e a sua lista de comentários e definições (*rdfs:comment*).

Ambos os índices são construídos uma única vez durante a fase de treinamento *offline* pelo módulo *Ontology Manager*, sendo serializados e armazenados na *Persistence* para serem recuperados durante a fase de execução *online*.

4.3.2 Processamento das QAIs

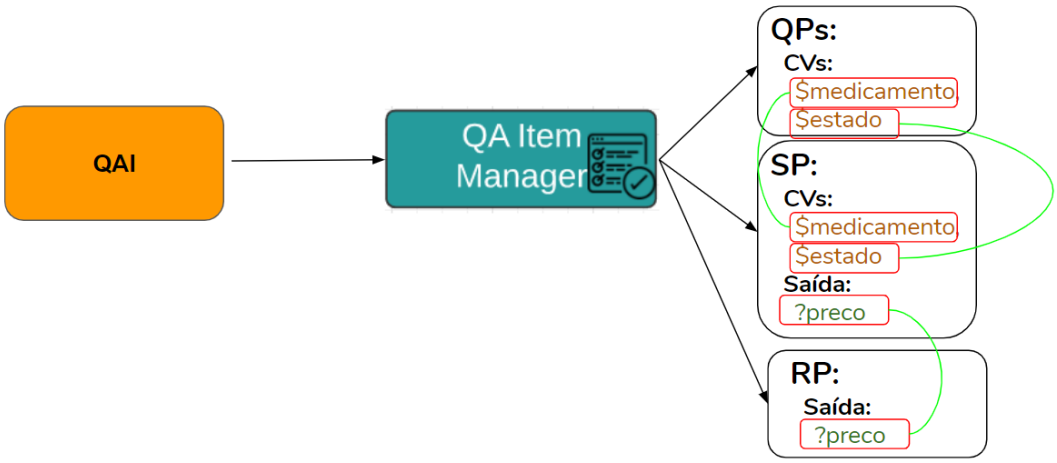
Após a etapa de construção dos índices, vem a etapa que pode ser considerada uma das mais relevantes para o processo de treinamento do *chatbot*, a etapa de processamento dos *QAI*s. Durante esta etapa o módulo *QA Item Manager* lê e interpreta os padrões de questões passados como *QA Itens* (*QAI*s) pelo desenvolvedor. A seguir são detalhados os processos executados durante esta etapa.

4.3.2.1 Checagem da consistência de um QAI

Inicialmente, para cada *QAI* são levantadas as listas de todas as *CV*s e as Variáveis Retornadas (*Returned Variables* (RV)) declaradas na consulta *SP*. Em seguida, para cada *QP* definida em um *QAI*, são checadas se as *CV*s citadas nesta pertencem ao conjunto de *CV*s declaradas em *SP*. Do mesmo modo, é checado se as *CV*s e RVs citadas no padrão de resposta (*RP*) estão contidas no conjunto declarado em *SP*. Caso alguma *QP* ou a *RP* cite uma variável não declarada em *SP*, o sistema interrompe o processo de treinamento, solicitando a revisão pelo desenvolvedor.

Na Figura 7 é apresentando um exemplo de checagem de consistência do *QAI* da Figura 5.

Figura 7 – Checagem de consistência do *QAI* da Figura 5



Fonte: Próprio autor.

4.3.2.2 Parsing e interpretação semântica de uma SPARQL query Pattern (SP)

A etapa de *parsing* de uma *SP* permite a interpretação semântica dos valores esperados para cada uma das *CV*s de um *QAI* de maneira automática. Esta etapa é de suma importância para o desempenho geral do sistema (como visto na Seção 6.3). Tal processo possibilita o uso do conhecimento de domínio presente na Ontologia durante o processamento da questão em LN, permitindo a inserção de *features semânticas* (*CVec*) ao classificador de questões (Seção 4.3.4). Para cada *QAI*, esta etapa tem como entrada sua *SP* e os índices de classes e propriedades da Ontologia. Durante esta etapa são recuperadas as *CV*s, juntas de seu tipo (recurso ou literal), classe, e no caso das do tipo literal, suas propriedades e classes proprietárias.

O tipo de uma *CV* especifica se esta deve ser substituída por uma *URI* que identifica um recurso no KG (no caso desta ser do tipo recurso) ou por um literal (*xsd:string, xsd:double, xsd:integer ou xsd:datetime*). Caso a *CV* seja inferida como sendo do tipo recurso, então o atributo classe representará a classe a qual o recurso substituindo a *CV* deverá ser uma instância. Caso a *CV* seja inferida como sendo do tipo literal, então o atributo classe assumirá os seguintes valores: *xsd:string*, caso esta seja um texto livre (*string*); *xsd:double*, caso esta seja um valor numérico com casas decimais; *xsd:integer*, caso esta seja um valor numérico inteiro; *xsd:datetime*, caso esta seja uma data. Além disso, esta ainda tem dois atributos adicionais, suas "*propriedades proprietárias*" e "*classes proprietárias*". No exemplo de *QAI* dado, a *CV* $state tem <state> como sua "propriedade proprietária" e <Price> como sua "classe proprietário".

```
{SELECT ?name (MAX(?priceAux) as ?price)  WHERE{
  ?s a <Medicine>;
    rdfs:label ?name;
    <price> ?appliedPrice.
```

```
  ?appliedPrice a <Price>;
    <state> $state;
    <value> ?priceAux.
  FILTER(REGEX(?name,$medicine,'i'))}
```

Por questão de conveniência, no decorrer desta dissertação, os pares de propriedades e classes proprietárias serão encarados como uma *string* de forma "*Propriedade@Classe*", sendo esta chamada de "*par-proprietário*".

O *parsing* semântico de uma *SP* é realizado durante a travessia da representação em árvore da consulta SPARQL de *SP* gerada pelo *parsing* de consultas SPARQL da biblioteca RDFLib[2]. Durante o processo de travessia, uma *CV* pode ser descoberta em dois contextos:.

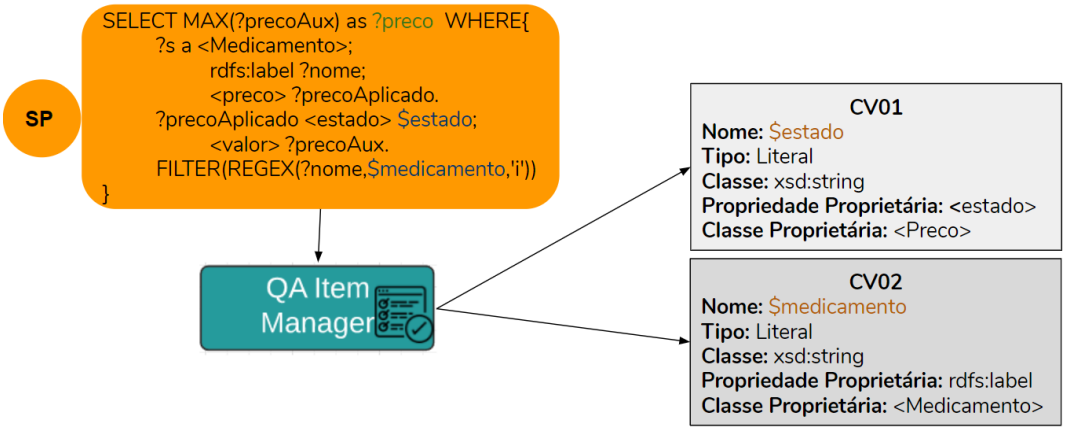
1. **Dentro de triplas:** No primeiro, a *CV* está localizada diretamente na definição das triplas do *graph pattern*[3] na cláusula *WHERE* da consulta SPARQL. Neste caso, se:
   - A *CV* ocupar a posição de sujeito da tripla (e.g., "$var <property> <object>."), é inferido que esta é do tipo recurso (desconsiderando casos de triplas inversas), onde sua classe pode ser recuperada através da existência de uma tripla de definição de classe explícita (e.g., "$var rdf:type <Class>.") ou através da inferência com base nas propriedades que esta relaciona-se;
   - Se a *CV* ocupa a posição de predicado da tripla (e.g., "<subject> $var <object>."), esta é inferida como sendo do tipo recurso e tendo sua classe como *rdf:Property*;
   - Se a *CV* ocupa a posição de objeto da tripla (e.g. "<subject> <property> $var."), esta é inferida como sendo do tipo recurso quando <property> é uma *owl:ObjectProperty* ou quando é localizada uma outra tripla onde esta *CV* apareça como sujeito, caso contrário ela é inferida como do tipo literal, recebendo a classe do contra-domínio de <property> (caso sua classe não possa ser definida, então será vista como um *xsd:string*).
2. **Dentro de expressões:** No segundo contexto, onde a *CV* pode ser localizada dentro de uma expressão (*BIND* ou *FILTER*) dentro da clausura *WHERE*, esta pode estar presente em três casos.
   - No primeiro, a *CV* está envolvida numa operação de comparação binária ($=, <, >\leq, \geq$), onde nestes casos a variável herda o mesmo tipo e classe da variável com a qual está relacionada (seja ela uma *CV* ou uma variável normal da consulta), exceto no caso de desigualdade (!=);
   - No segundo caso, a *CV* está envolvida em uma operação matemática, onde neste caso ela é inferida como um literal da classe *xsd:double*;
   - No terceiro caso, a *CV* está envolvida em uma função embarcada (*built-in function*) do SPARQL (*e.g.*, *REGEX, LANG ,CONTAINS, UCASE, month*), nestes casos o tipo

[2] https://rdflib.readthedocs.io/en/stable/
[3] https://www.w3.org/TR/rdf-sparql-query/#GraphPattern

e a classe da *CV* são inferidos de acordo com a função e qual posição de parâmetro ela ocupa (*e.g.*, "UCASE($var)", indicará que $var é uma literal da classe *xsd:string*). Ainda no terceiro contexto, caso a função sendo aplicada sobre a *CV* seja uma função de comparação, *e.g.*, " REGEX(?normalVar,$var,"i") ", a *CV* "$var" herdará o mesmo contexto da variável "?normalVar", recebendo as mesmas propriedades e classes proprietárias.

Na Figura 8 é apresentando um exemplo de *Parsing Semântico* do *QAI* da Figura 5.

Figura 8 – *Parsing Semântico* do *QAI* da Figura 5



Fonte: Próprio autor.

4.3.2.3 Construção da representação vetorial (QV) de uma QP

Tendo em vista que o sistema já capaz de identificar e compreender as *CV*s utilizadas nos padrões de questões *QP*s, o próximo passo a se tomar é a construção da representação vetorial de cada *QP*. Esta representação é necessária, pois os modelos de classificação são projetados e implementados para receberem vetores numéricos como entrada, o que torna impossível a passagem direta do texto das questões para estes modelos. Deste modo, cada *QP* é mapeado para um vetor que o representa, sendo este chamado de vetor de questão *Question Vector* (QV). Um QV é formado pela concatenação de dois outros vetores sendo o primeiro o vetor de sentença *Vector Sentence* (VS) e o segundo o vetor que representa os tipos das *CV*s utilizadas no *QP*. Portanto, QV é definido da seguinte forma:

– $QV = VS \oplus CVec$, onde $\oplus$ é a operação de concatenação de dois vetores.

O vetor VS que representa um *QP* é construído através do uso de técnicas de *NLP* e *Word Embedding* (LI; YANG, 2018). Por conta do vetor VS ser construído apenas com base no texto do *QP*, este é considerado o vetor de *features textuais*. Os passos necessários para a construção de VS, são:

1. O primeiro passo no processamento do VS de um *QP* é a substituição dos marcadores de *CV*s por símbolos *Out of Vocabulary* (OOV). Um símbolo OOV representa um *token* que não faz parte do vocabulário utilizado pelo *Word Embedding*, assim tal simbolo é

desconsiderado na hora do cálculo do vetor VS;

2. O segundo passo é a normalização da *string*, onde atualmente está sendo realizado apenas o processo de conversão da *string* para *lower case*. Durante este passo, testes foram realizados para avaliar o impacto do uso de técnicas de *stemmização/lemmatização* e remoção de *stop words*. No entanto, enquanto o primeiro veio a degradar o desempenho do sistema, o segundo teve resultados pouco expressivos e variáveis. Um provável motivo para a degradação ocorrida com o uso da *stemmização/lemmatização* é devido ao fato de que tal etapa não foi realizada durante o treinamento do modelo de *word embeddings* utilizado (HARTMANN *et al.*, 2017), deste modo levando a detecção de muitas OOVs durante o cálculo do vetor;
3. O terceiro e ultimo passo é o cálculo do vetor VS. Para este cálculo é utilizada a biblioteca de *NLP*, *SpaCy*(EXPLOSION, 2019). *SpaCy*[4] é uma biblioteca de *NLP* orientada para a produção de sistemas industriais, focando no alto desempenho e praticidade de uso. *SpaCy* já conta com modelos pré-treinados de *word embeddings* para uma ampla variedade de linguagens, dando suporte a mais de 53 linguagens diferentes. No entanto, o desempenho de cada um destes modelos encontra-se em estágios diferentes, com o modelo em Inglês estando em um estágio avançado, alcançando resultados superiores a muitos concorrentes. Contudo, para este projeto o idioma Português do Brasil (PT-BR) é de extrema importância. No entanto, o modelo disponível nativamente na plataforma não veio a alcançar os resultados esperados. Portanto, neste trabalho optou-se pela adoção de um modelo de *word embedding* para o PT-BR de terceiros. Neste trabalho, o modelo de *word embeddings* utilizado foi o modelo GloVe de 100 dimensões[5] produzido por (HARTMANN *et al.*, 2017). Este modelo foi então carregado para o *SpaCy*, deste modo aproveitando todo o *pipeline* de processamento da ferramenta. O modelo final para o PT-BR produzido pode ser baixado no link[6]. Em *CONQUEST* também é possível utilizar outras linguagens, como o Inglês, através do uso dos modelos nativos do *SpaCy* ou de um modelo customizado para uma linguagem específica assim como no caso do PT-BR.

CVec (*Context Vector*) é um vetor que representa o número de *CV*s (entidades nomeadas) necessárias para responder a questão codificada por este. *CVec* é um vetor de $n+3$ dimensões, onde n é o número de pares-proprietários ("Propriedade@Classe") para *CV*s literais da classe *xsd:string* (como visto na Seção 4.3.3). As outras têes dimensões adicionais de *CVec* referem-se as *CV*s literais das classes *xsd:integer, xsd:double* e *xsd:datetime*. Deste modo, para cada *CV* existente no *QAI*, o contador na posição representando o tipo da *CV* será incrementado em 1. O uso do *CVec* permite ao sistema utilizar o conhecimento adquirido através da interpretação semântica da questão realizada na etapa da interpretação semântica de *SP*, além do conhecimento existente na base de conhecimento. Por conta do uso das informações advindas

[4] https://spacy.io/
[5] http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc
[6] https://sourceforge.net/projects/conquest-sqai/files/model_pt-br.tar.xz/download

da interpretação semântica e da base de conhecimento, *CVec* é considerado o vetor de *"features semânticas"* do *template*. O uso das *features semânticas* permite que o sistema alcance uma maior generalização no aprendizado da tarefa de classificação, ajudando a contornar o problema do conjunto escasso de exemplos de treino em fases iniciais do sistema. Como visto na seção 6.3, o uso das *features semânticas* tem um grande impacto positivo no desempenho do sistema de classificação.

4.3.3 Treinamento do módulo de NER

Durante a fase *online*, é necessário que o sistema seja capaz de receber como entrada uma questão qualquer dada em LN e gerar sua representação vetorial QV. Para isto, a construção do vetor de *features textuais* VS pode ser realizada de maneira direta, como a realizada durante a fase *offline*. No entanto, para a construção do vetor de *features semânticas* é necessária a identificação de possíveis valores para as *CV*s presentes na questão. Para este propósito, *CONQUEST* recorre ao uso de um mecanismo de *NER* responsável por reconhecer possíveis valores relevantes para a questão.

O módulo de Reconhecimento de Entidade Nomeadas (*NER*) (NADEAU; SEKINE, 2007) é responsável por identificar possíveis candidatos em uma sentença em LN para valores de *CV*s. Estes candidatos são utilizados para a construção do vetor *CVec* da questão dada como entrada. O uso do *NER* permite que possíveis valores para *CV*s sejam identificados diretamente da questão, dispensando a necessidade da solicitação de cada *CV* individualmente durante a fase de execução *online*. Em *CONQUEST*, o módulo *NER* é treinado para reconhecer possíveis valores para *CV*s do tipo literal (*xsd:string, xsd:double, xsd:integer ou xsd:datetime*):

- Para entidades dos tipos numéricos, tais como *xsd:integer* e *xsd:double*, *CONQUEST* utiliza um mecanismo simples, mas eficaz, de expressão regular para sua identificação.
- Para o reconhecimento de entidades do tipo data (*xsd:datetime*), *CONQUEST* reusa a biblioteca *dateparser* (SCRAPINGHUB, 2019). O *dateparser*[7] fornece módulos para analisar facilmente datas localizadas em quase todos os formatos de *string* comumente encontrados em páginas da web. Com o uso desta biblioteca é possível identificar vários formatos diferentes de datas, desde as datas explicitamente declaradas, quanto datas relativas (e.g., ontem, hoje, uma semana atrás, etc.) em várias línguas e calendários diferentes.
- Para literais da classe *xsd:string*, *CONQUEST*, classifica um candidato para o seu provável par-proprietário. Isto é feito através da consulta dos termos em um índice *Apache Solr* (SMILEY; PUGH, 2011). Para cada par-proprietário utilizado no conjunto de *QAI*s, são buscados seus possíveis valores contidos no EKG, e.g., caso o par-proprietário "*ont:nome@ont:Pessoa*" seja utilizado por uma *CV* do tipo *xsd:string*, então serão recuperados todos os possíveis valores para o atributo *ont:nome* de instâncias da classe *ont:Pessoa*.

[7] https://pypi.org/project/flexible-data-parser/

Estes valores recuperados serão indexados como chaves de busca para o par-proprietário "*ont:nome@ont:Pessoa*". Deste modo, caso o nome de uma pessoa existente no EKG seja consultado, então será retornado seu par-proprietário. Por exemplo, caso exista uma *ont:Pessoa* com o atributo *ont:nome* "*Michael Jackson*" no EKG, e o *NER* encontre esta *string* em uma questão de entrada, este retornará que "*Michael Jackson*" é um candidato para uma *CV* da classe *xsd:string* com par-proprietário *ont:nome@ont:Pessoa*.

4.3.4 Treinamento do classificador de questões

O mecanismo de classificação de questões permite a interpretação da intenção de uma questão emitida em LN pelo usuário. O classificador é responsável por classificar a questão do usuário para um dos padrões de *QAI* suportados, atuando assim como uma peça central para a arquitetura do sistema.

Como mecanismo de classificação, em *CONQUEST* foi optado pelo uso de um classificador baseado em modelos de ML. A escolha por este tipo de classificador se deve por conta de sua alta capacidade de generalização e versatilidade, visando tratar o problema da variabilidade linguística. No entanto, o uso de tal abordagem traz consigo um novo desafio, o problema do pequeno conjunto de amostras de treinamento (*small sample size*, em tradução livre) (YANG *et al.*, 2010). Neste problema, o modelo não consegue convergir para uma boa performance por conta do conjunto restrito de treinamento. É esperado que o sistema encare este problema durante os estágios iniciais da implantação de um *chatbot* construído por *CONQUEST*. Isto se deve ao fato de não ser razoável esperar que o desenvolvedor forneça uma entrada inicial com vários exemplos de treinamento. Nestes casos, como observado nos experimentos preliminares (Tabela 6), o sistema alcança uma má performance quando considerando apenas *features textuais* (representação vetorial da sentença em linguagem natural) durante a classificação. Para solucionar tal desafio, *CONQUEST* realiza a etapa de enriquecimento semântico de *features*. Em *CONQUEST*, além da representação vetorial da sentença, o sistema de classificação utiliza um vetor que indica as possíveis entidades nomeadas existentes na questão (*CVec*).

Durante a fase de treinamento, o classificador tem como entrada duas matrizes: F_{nXm} e L_{nX1}. A matriz F é a *feature matrix* contendo o conjunto de exemplos de treinamento, onde n representa o número de exemplos de questões em LN (*QP*s) de todos os *QAI*s. Para cada linha r_i de F, onde $0 \leq i \leq n$, r_i é a representação vetorial (QV) de uma *QP*, onde $QV = VS \oplus CVec$, como definido na Seção 4.3.2.3. Deste modo, m é igual ao tamanho de QV. A matriz L é a *label matrix*, representando as classes de cada linha da matriz F. Para cada linha rl_i de L, onde $0 \leq i \leq n$, rl_i é o *id* do *QAI* o qual a linha r_i de F faz parte.

Durante a fase de consulta online, o modelo treinado recebe como entrada um vetor *QP*, que representa a questão do usuário, e retorna como saída um vetor P_q, onde q é igual ao número de *QAI*s suportados pelo sistema. Para cada linha rp_i de P, onde $0 \leq i \leq q$, rp_i representada a probabilidade da questão de entrada ser classificada para o *QAI* de *id* i. O

classificador considera como a intenção da questão, o QAI_c, quando $\forall x(rp_c > rp_x), x \neq c$ e $rp_c \geq min\_confidance$, onde $min\_confidance$ é o hiper-parâmetro que representa o nível de confiança mínimo aceito para uma classificação. Quanto maior for o valor de $min\_confidance$, mais rigoroso será o classificador, o que pode gerar uma maior ocorrência do uso do diálogo de clarificação para definir a intenção da questão do usuário. Quanto menor for o valor de $min\_confidance$, menos rigoroso será o classificador, o que pode gerar uma menor ocorrência do uso do diálogo de clarificação, o que pode levar a interpretações erradas.

Foram realizados experimentos, apresentados na Seção 6.3, para a seleção do melhor algoritmo de treinamento do classificador (classe do modelo). Além disso, foi avaliado o impacto do uso das *features semânticas* nesta tarefa. Os experimentos foram realizados sobre o estudo de caso apresentado na Seção 5, os conjuntos de treinamento|validação e teste são apresentados nos apêndices da seção Apêndice A. Ao fim, *CONQUEST* utiliza um mecanismo de classificação baseado em ML enriquecido com *features semânticas*. O modelo selecionado é o *Naive Bayes Gaussiano*, que aliado ao uso das *features semânticas* performou como um dos melhores modelos testados, tanto em aspectos de taxas de acertos na classificação, quanto no tempo necessário para o seu treinamento. Além disso, esta classe permite a atualização do modelo durante o tempo de execução (*online*). Isto permite que o *chatbot* resultante adapte-se em tempo de execução. Os experimentos, juntos de seus resultados são apresentados na Seção 6.3.

4.3.5 Armazenamento dos artefatos treinados

Durante a fase de treinamento *offline*, *CONQUEST* produz artefatos que são necessários para a execução do *chatbot* sendo produzido. Além disso, durante a fase de execução, informações sobre o estado da interação com o usuário também são armazenadas. Tais artefatos são armazenados no módulo de Persistência (*Persistence*). Os artefatos armazenados no módulo *Persistence*, são:

- ***Ontology Index:*** Salva as informações do esquema da Ontologia de domínio de forma que estas possam ser acessadas diretamente de maneira simples durante a fase *online*. Estas informações são salvas como os índices descritos na seção 4.3.1. Os índices são salvos como objetos *python pickle* (CPYTHON, 2019);
- ***QA Itens:*** Salva as informações processadas dos *QAI*s suportados pelo *chatbots*. Estas informações são utilizadas no processo de interpretação da questão; na conferência e solicitação de parâmetros; na construção da consulta SPARQL; e na construção da resposta. Os *QAI*s são salvos como objetos *python pickle*;
- ***NLP Model:*** Salva o modelo utilizado para o processamento do texto em linguagem natural, incluindo o *workflow* para a normalização e segmentação do texto, o modelo de *word embeddings* e o índice *Apache Solr* utilizado no *NER*. O modelo *NLP* do *SpaCy* é armazenado seguindo método disponibilizado pela biblioteca, enquanto o índice *Apache Solr* é armazenado como um *core* junto de uma instância de execução *Apache Solr*;

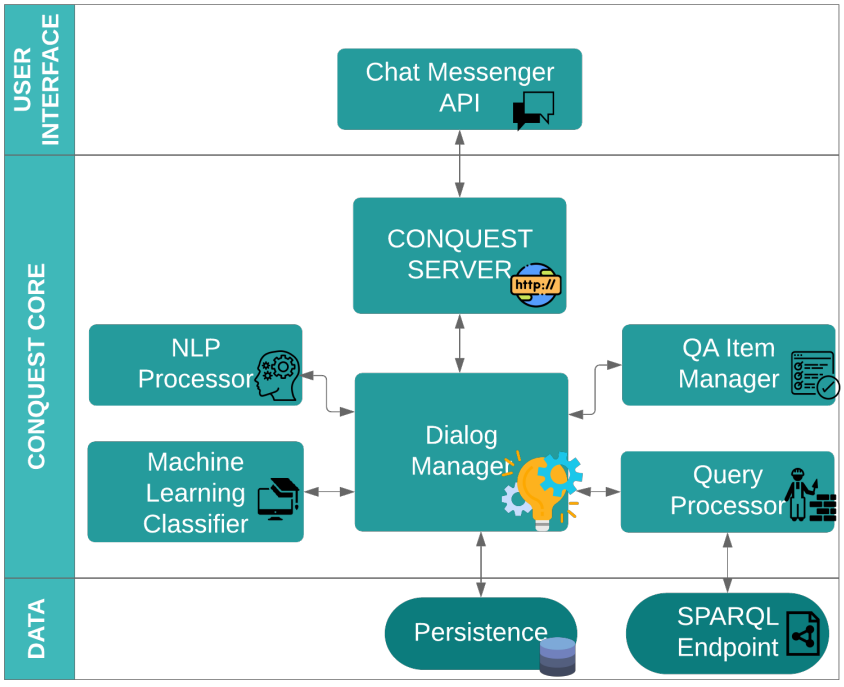
– ***Classification Model:*** Salva o modelo de classificação de uma questão em linguagem natural para um *QAI*. O modelo é salvo como um objeto *python pickle*;
– ***Interaction State:*** Salva o atual estado de uma interação com o usuário durante o *chatflow*. O estado da interação consiste do atual ponto da interação seguindo o *chatflow* 10 e as informações adquiridas até então (e.g., questão dada como entrada, *QAI* classificada, valores para *CV*s e demais informações para um diálogo coerente). Isto garante que o *chatbot* realize interações longas de maneira consistente. Para o armazenamento desta informação é utilizado o banco de dados *NoSQL*, *MongoDB* (MONGODB, 2019).

4.4 Fase de Consultas Online

A fase de consultas *online*, é executada por uma instância de um *CONQUEST Chatbot*. Esta instância acessa o conhecimento presente na "memória" armazenada na *Persistence*. Isto é feito com o objetivo de consultar suas lições aprendidas sobre os padrões de questões (*QAI*s) que esta é capaz de responder; o conhecimento específico do domínio presente na ontologia; os modelos de *NLP* e de classificação de intenção aprendidos; além dos estados das interações ocorrendo durante diálogos com os usuários. Este *CONQUEST Chatbot* executa de maneira automática o tratamento da entrada e saída de mensagens; interpretação da questão do usuário; gerenciamento do fluxo da interação; e a consulta ao EKG.

A Figura 9 apresenta a arquitetura de um *CONQUEST Chatbot*. Esta arquitetura é dividia em três camadas: *User Interface, CONQUEST Core e Data*.

Figura 9 – Arquitetura de um *CONQUEST Chatbot*.



Fonte: Próprio autor.

4.4.1 Camada de Interface com o Usuário

A *User Interface* é a camada que interage diretamente com o usuário durante o processo de consultas, tratando da apresentação das mensagens. Esta camada tem como objetivo fornecer uma interface intuitiva e prática para o acesso do usuário ao *chatbot*. Para tal fim, esta camada possui um conjunto de *APIs* para a comunicação com serviços de mensageiros instantâneos, as *Chat Messenger API*.

Uma *Chat Messenger API* implementa uma interface de comunicação entre um IM específico e o servidor do *CONQUEST Chatbot* (módulo *CONQUEST Server*). Isto é feito através do recebimento das mensagens vindas do IM, seguindo de seu encaminhamento (com o uso de requisições HTTP) para o *CONQUEST Server* e terminando na apresentação da resposta do servidor, utilizando os recursos específicos de cada plataforma.

O uso da *API* de comunicação com o IM separadamente do servidor do *chatbot* permite que um mesmo *chatbot* seja disponibilizado em múltiplas plataformas (e.g., *Telegram, Facebook Messenger, Whatssapp*, etc.) simultaneamente sem que haja a necessidade de duplicação de código. Atualmente, o *framework CONQUEST* dá suporte a *API* do *Telegram*, com as demais estando em desenvolvimento.

4.4.2 Camada de Controle e Processamento, CONQUEST Core

A *CONQUEST Core* é a principal camada da arquitetura de um *CONQUEST Chatbot*, sendo responsável pelo processamento das questões e suas respostas. Esta camada é composta pelos seguintes componentes:

4.4.2.1 Módulo CONQUEST Server

O *CONQUEST Server* é o componente responsável por disponibilizar os serviços do *chatbot* através de requisições HTTP, atuando na fronteira entre a camada de interface e o núcleo do sistema. Este componente atua recebendo requisições HTTP como entrada, encaminhando-as para o *Dialog Manager* e por fim retornado a resposta ao usuário. O *CONQUEST Server* pode ser acessado tanto através de um serviço de IM (e.g. Telegram, Facebook Messenger, etc.), quanto diretamente através de requisições HTTP (e.g., via um *Web Browser* ou outro cliente HTTP), deste modo podendo ser disponibilizando em uma ampla gama de canais simultaneamente.

4.4.2.2 Módulo Dialog Manager

O *Dialog Manager* é o componente central da arquitetura de um *CONQUEST Chatbot*, sendo responsável pelo gerenciamento do fluxo do processamento da solicitação. Este componente realiza a troca de informações entre os demais módulos, atuando como o intermediário. Além disso, este componente é o responsável pelo gerenciamento do fluxo de

diálogo.

4.4.2.3 Módulo NLP Processor

Este é o módulo responsável por receber uma questão *Q* em LN e convertê-la para sua representação vetorial QV. Para este fim, é realizada a sequência de passos:

1. **Tratamento da sentença *Q*:** Neste passo é realizado o processo de normalização (conversão de *Q* para minúsculo) e a *tokenização* de *Q*;
2. **Identificação das entidades nomeadas contidas na sentença:** Este passo é realizado pelo módulo de *NER* treinado na etapa *offline*. Onde, o primeiro tipo de entidades buscadas são as literais da classe *xsd:string*. Para isso, é realizado o processo de janela deslizante de um *n-gram* (SHISHTLA *et al.*, 2008) sobre os *tokens* contidas em *Q*. Inicialmente o valor de *n* é igual ao número de *tokens* de *Q*, onde a janela vai deslizando da esquerda para a direita, uma palavra por vez e diminuindo seu tamanho em 1 cada vez que alcança o final da sequência. Durante este processo, cada *n-gram* é consultado no índice *Solr*, e se este estiver contido, então ele é removido da sequência. Posteriormente são buscadas as entidades do tipo *xsd:datetime* e por fim as dos tipos numéricos (*xsd:integer* e *xsd:double*) como definido na seção 4.3.2.3;
3. **Cálculo do vetor VS de *Q*;**
4. **Cálculo do vetor *CVec* de *Q*;**
5. **Cálculo da representação QV de *Q*.**

Os passos 3, 4 e 5 são realizados como descritos na Seção 4.3.2.3, onde o *CVec* construído no passo 4 usa as entidades encontradas no passo 2.

4.4.2.4 Módulo Machine Learning Classifier

O classificador recebe como entrada a representação vetorial QV de *Q* e retorna o nível de confiança da classificação de *Q* para cada um dos *QAI*s do sistema.

Para a classificação ser considerada correta, o *QAI* melhor classificado deve possuir uma confiança acima de um valor limiar mínimo (*min_confidance*) definido pelo desenvolvedor (por padrão este valor está definido como 80%). Caso a confiança do melhor *QAI* seja menor que este limiar, o sistema engaja-se no diálogo de clarificação para confirmar a intenção do usuário. Esta etapa de clarificação é realizada da seguinte forma:

1. O sistema busca o *QP* do *QAI* candidato com a maior similaridade (*Cosine similarity*[8] entre o VS de *QP* e *Q*) com a questão *Q*;
2. As *CV*s localizadas no *QP* selecionado são substituídas pelas entidades nomeadas encontradas em *Q*, de acordo com sua classe (no caso de *CV*s do tipo *xsd:string*, são buscadas entidades classificadas com o mesmo par-proprietário da *CV*) e seguindo a ordem de

[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

aparição no texto. *CV*s para os quais não foi possível identificar um valor em Q são mantidas com o seu rótulo;

3. A questão gerada na etapa anterior é apresentada ao usuário, solicitando-o para confirmar sua intenção;
4. Caso o usuário confirme a classificação, o sistema insere Q como um novo *QP* no *QAI* classificado (após a substituição das entidades nomeadas pelos respectivos marcadores de suas *CV*s) e atualiza o modelo de classificação considerando este novo exemplo. Caso o usuário negue a classificação, a questão Q é armazenada como um exemplo de questão não respondida.

Caso mais de um *QAI* alcancem uma confiança maior que o limiar ($min\_confidance$), é calculada a distância entre os n (definido pelo desenvolvedor, tendo $n = 3$ como padrão) melhores. Caso a distância entre estes seja menor que um determinado valor definido pelo desenvolvedor (por padrão 0.05), o sistema engaja-se no diálogo de desambiguação. Esta etapa de desambiguação é realizada da seguinte forma:

1. Para cada *QAI* candidato, o sistema busca seu *QP* com a maior similaridade (*Cosine similarity* entre o VS do *QP* e Q) com a questão Q;
2. Para cada *QP* selecionado, as suas *CV*s são substituídas pelas entidades nomeadas encontradas em Q, de acordo com sua classe (no caso de *CV*s do tipo *xsd:string*, são buscadas entidades com classificadas com o mesmo par-proprietário da *CV*) e seguindo a ordem de aparição no texto. *CV*s para os quais não foi possível identificar um valor em Q são mantidas com o seu rótulo;
3. As questões geradas na etapa anterior são apresentadas seguindo a ondem decrescente das confianças dos seus respectivos *QAI*, solicitando ao usuário a interpretação correta;
4. Caso o usuário selecione uma das opções apresentadas, o sistema insere Q como um novo *QP* no *QAI* classificado (após a substituição das entidades nomeadas pelos respectivos marcadores de suas *CV*s) e atualiza o modelo de classificação considerando este novo exemplo. Caso o usuário negue todas as opções, a questão Q é armazenada como um exemplo de questão não respondida.

4.4.2.5 Módulo QA Item Manager

Este módulo recupera as informações sobre o *QAI* classificado pelo sistema. Estas informações são utilizadas para descobrir as *CV*s necessárias para a questão, preenchendo-as ou solicitando-as. Caso o *QAI* classificado possua *CV*s não preenchidas, o *chatbot* inicia o diálogo de solicitação destas.

Para cada *CV* não preenchida, o sistema recupera sua classe. Caso a classe seja uma literal, o sistema recupera as informações sobre o seu par-proprietário contidas nos índices de classes e propriedades. As informações coletas são o rótulo e a definição da classe e propriedades proprietárias da *CV*. Estas informações são utilizadas para a construção de uma mensagem de

solicitação do valor da *CV*. A solicitação gerada apresenta o rótulo da propriedade proprietária da *CV*, junto da definição desta propriedade (o mesmo sendo realizado para a classe proprietária). Este diálogo de solicitação segue um padrão que pode ser alterado pelo desenvolvedor.

O *QA Item Manager* é responsável pela recuperação do *template* de consulta SPARQL, *SP* a ser utilizado na construção da consulta para responder *Q*. Além disso, este módulo também é responsável pela recuperação do padrão de resposta *RP* a ser gerado.

4.4.2.6 Módulo Query Processor

Este módulo recebe como entrada o *template* de consulta SPARQL, *SP* e o seu conjunto de *CV*s preenchidos. É então realizado o processo de montagem e execução da consulta no *Endpoint SPARQL*. Esta consulta é construída ao substituir os marcadores das *CV*s em *SP* por seus respectivos valores obtidos.

Por fim o resultado da consulta é retornado para o *Dialog Manager* que gera a resposta em linguagem natural com base no padrão *RP*.

4.4.3 Camada de Dados

A terceira e última camada é a camada de Dados (*Data Layer*). Esta camada é responsável pelo armazenamento do conhecimento do *chatbot*, sendo este tanto o conhecimento aprendido durante a fase de treinamento, quanto o KG sendo consultado. Esta camada é composta por dois componentes.

- ***Persistence:*** Como definido na Seção 4.3.5, este componente possui o conhecimento aprendido durante a etapa *offline* de treinamento. Este conhecimento é recuperado pelo *Dialog Maneger* e distribuído para os demais módulos do *CONQUEST Core* para que estes possam executar suas tarefas. Além disso, a persistência também é utilizada para armazenar dados sobre o estado da interação com os usuário e novos exemplos para serem utilizados como treinamento para o *chatbot*.
- ***SPARQL Endpoint:*** Este é um componente externo ao sistema. Este sendo acessado através de requisições HTTP com o objetivo de executar consultas SPARQL. Este acesso é realizado através da biblioteca *SPARQL Wrapper* (RDFLIB, 2019) que é responsável por manipular a requisição HTTP e a resposta do *endpoint*.

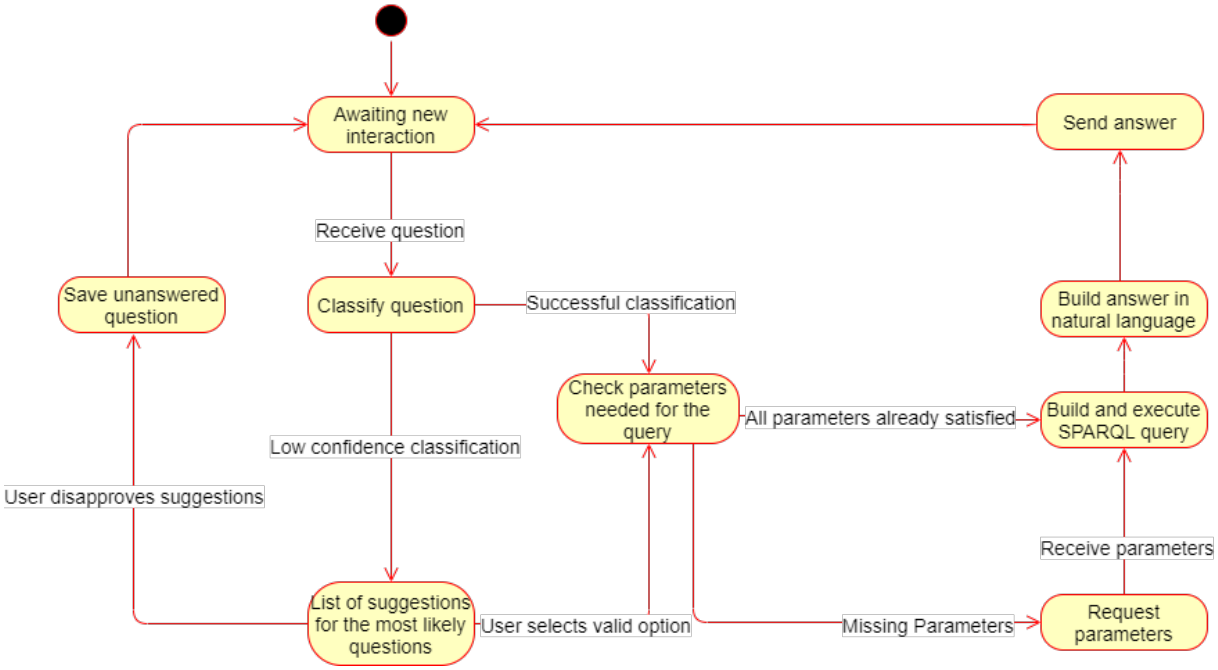
4.5 Fluxo de Interação Seguido Por um CONQUEST Chatbot

Um *CONQUEST Chatbot* possui um fluxo de conversação (*chatflow*) definido tendo em mente o cenário necessário para a realização da tarefa de TBIQA. Este *chatflow* pode ser visto na Figura 10. O *chatflow* é resumido da seguinte forma:

1. O *chatbot* recebe a questão em LN, classificando-a para um *QAI*;

2. Caso esta classificação não seja possível, o *chatbot* realizada o diálogo de *desambiguação*;
3. Após a classificação da questão, o *chatbot* confere se todas as *CV*s foram preenchidas, solicitando-as caso o contrário;
4. O *chatbot* consulta o KG;
5. O *chatbot* constrói a resposta em LN seguindo o *RP* definido no *QAI*;
6. Por fim, a resposta é retornada ao usuário.

Figura 10 – *Chatflow* seguido por um *CONQUEST Chatbot*.



Fonte: Próprio autor.

5 ESTUDO DE CASO: *MEDIBOT*

Como um estudo de caso, neste trabalho é apresentado *MediBot*(AVILA *et al.*, 2019a; AVILA *et al.*, 2019b; AVILA *et al.*, 2019c), um *chatbot* para IQA sobre KGs de medicamentos, poderia ser implementado com o uso do *framework CONQUEST*.

5.1 Apresentação do MediBot

O uso de medicamentos para tratamento de doenças e enfermidades é percebido como uma prática válida para melhorias no que tange ao estado de saúde da população. Contudo, a ingestão inadequada de medicamentos pode acarretar a ocorrência de efeitos colaterais e intoxicação. Segundo a Organização Mundial da Saúde (OMS) (WHO *et al.*, 2002), são definidos como eventos adversos à medicamentos quaisquer ocorrências médicas desfavoráveis que possam ocorrer durante a fase de tratamento medicamentoso.

No Brasil, entre os agentes causadores de intoxicação, como agrotóxicos, drogas ilícitas, pesticidas, inseticidas e alimentos impróprios para consumo, os medicamentos ocupam a primeira posição (CORRÊA *et al.*, 2013). Somente em 2016, houveram aproximadamente 56.937 casos registrados de intoxicação humana, dos quais, 226 evoluíram para óbito, sendo 20.527 gerados em decorrência do uso de medicamentos (SINITOX, 2016).

De acordo com a literatura, ações como melhorar a qualidade das prescrições e prevenir a automedicação inadequada poderiam reduzir a ocorrência de reações adversas e eventual o óbito (QUENEAU *et al.*, 2007). Tendo isto em vista, *MediBot* atua como uma ferramenta de acesso ao conhecimento e conscientização para população sobre medicamentos e seus riscos. *MediBot* atua sobre um KG resultante da integração de diversas do domínio, permitindo o acesso às informações contidas nestas de maneira integrada e simplificada através de uma interface de linguagem natural.

MediBot é um *chatbot* que pode ser utilizado tanto através do aplicativo móvel do *Telegram* e de sua interface *web*, quanto no ambiente *desktop*. Originalmente, *MediBot* foi implementado em JavaScript utilizando NodeJS. *MediBot* pode ser contactado via *Telegram* através do id *@LDM_MediBot*. Para mais informações, vídeos demonstrativos e exemplos acessar o link[1].

MediBot possui um conjunto de consultas SPARQL pré-definidas para as quais a entrada do usuário é mapeada. Oito tipos de consultas foram definidos, conforme mostrado na Tabela 4.

[1] https://anonimoanim10.github.io/mediBot/

Tabela 4 – Tipos de consultas respondidas pelo *MediBot* no modo respostas rápidas.

Tipo de Consulta	**Exemplo**
Medicamentos com um princípio ativo	Quais são os medicamentos com a substância dipirona?
Definição de termos no domínio	Defina Classe Terapêutica
Informações sobre um certo medicamento	Fale sobre o medicamento aspirina
Riscos de um medicamento	Quais são os riscos de um medicamento reopro?
Lista de representações de um medicamento	Quais são as apresentações do medicamento reopro?
Informações sobre a apresentação do código de barra	Me informe sobre a apresentação do código de barras 7896382701801
Preço de uma apresentação com a taxa de ICMS em um Estado	Quais são os preços com taxa de ICMS da apresentação 7896382701801 no estado do Ceará?
Preço para todas as apresentações de um medicamento	Qual o preço do medicamento buscopan?

Fonte: Próprio autor.

5.2 Implementando o *MediBot* no *Framework CONQUEST*

Neste trabalho, o modelo de *word embeddings* utilizado foi o modelo GloVe de 100 dimensões produzido por (HARTMANN *et al.*, 2017). Este modelo foi então carregado no *SpaCy*, deste modo aproveitando todo o *pipeline* de processamento da biblioteca. O modelo final para o Português do Brasil produzido pode ser baixado no *link*[2].

O processo de desenvolvimento mostrou-se bastante natural com a adoção de *CONQUEST*, exigindo que o desenvolvedor dê como entrada apenas algumas variações da questão em linguagem natural. Além disso, durante o uso, o *chatbot* produzido aprende novas variações de uma mesma questão de maneira automática, minimizando o esforço inicial para a construção do conjunto de exemplos de padrões para uma mesma questão. No *link* [3] é possível encontrar os arquivos necessários para a implementação de uma instância do *MediBot* utilizando o *framework CONQUEST*.

A Figura 11 mostra um caso em que o *chatbot* pode responder à pergunta sem exigir a interação do usuário. Neste exemplo, o usuário pergunta o padrão “Quais medicamentos possuem o princípio ativo $substância? ”. Este padrão pode ser fornecido como entrada para o sistema usando um arquivo JSON que representa a estrutura do *QAI* desejado:

```
{
"QPs": ["Quais medicamentos possuem o princípio ativo $substance?"],
"SP": "SELECT  ?medicine WHERE{
  ?s a <Medicine>;
    rdfs:label ?medicine;
    <hasSubstance> ?substance.
  ?substance rdfs:label ?substance_name.
  FILTER(REGEX(?substance_name,$substance,'i'))}",
```

[2] https://sourceforge.net/projects/conquest-sqai/files/model_pt-br.tar.xz/download
[3] https://github.com/CaioViktor/CONQUEST_medibot_experiments

```
"RP":{
    "header": "Aqui estão os medicamentos com este princípio ativo:",
    "body": "?medicine",
    "footer":""
    }
}
```

Figura 11 – Resposta direta à uma questão no Telegram.

Fonte: Próprio autor.

Neste exemplo, o usuário solicita "*Quais medicamentos têm o princípio ativo Abacavir*". Neste caso, o *chatbot* consegue classificar a questão com sucesso, não recorrendo ao diálogo de clarificação. Além disso, todos os parâmetros (valores para as *CV*s) necessários já foram fornecidos, deste modo, não havendo a etapa de solicitação de parâmetros ausentes. Em seguida, o *chatbot* constrói e executa a consulta SPARQL no KG. Por fim, é realizada a construção e apresentação da resposta ao usuário.

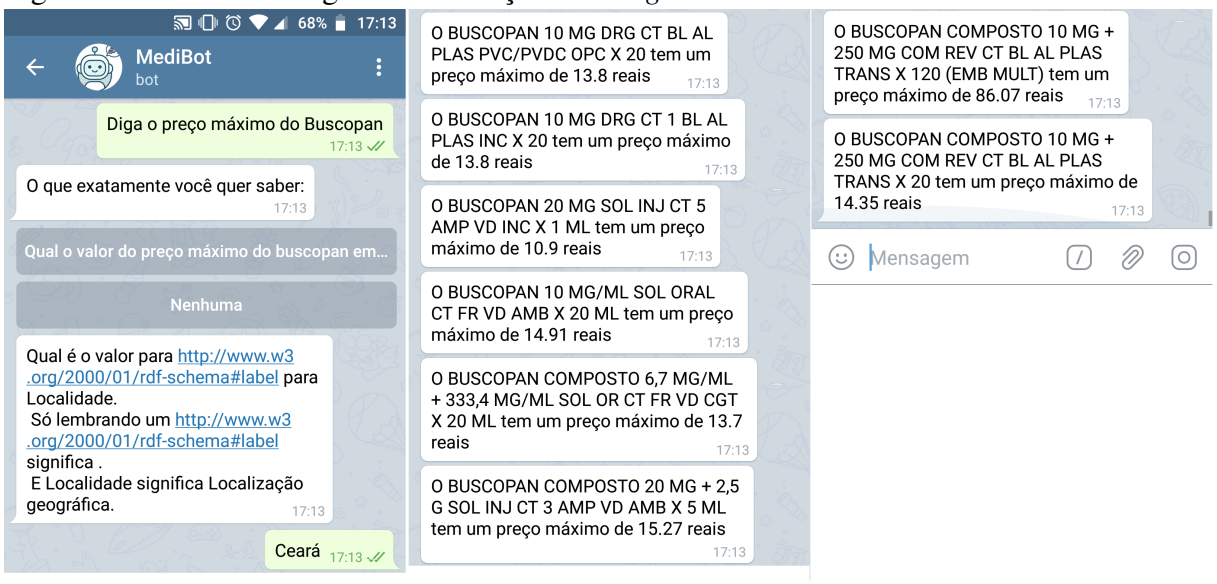
A Figura 12 mostra um caso em que o *chatbot* não pode responder à pergunta diretamente. Considerando o padrão de questão: "Qual o preço máximo para um dado medicamento em um certo estado", onde foi dado como entrada apenas a *QP*: "Qual o preço máximo para o

medicamento $medicamento no $estado". Este template pode ser fornecido como entrada para o sistema usando um arquivo JSON que represente a estrutura do *QAI* desejado:

```
{
    "QPs": [
      "Qual o preço máximo para o medicamento \$medicine no \$state?"
    ],
    "SP": "
    SELECT ?name (MAX(?priceAux) as ?price)  WHERE{
      ?s a <Medicine>;
        rdfs:label ?name;
        <Price> ?appliedPrice.
      ?appliedPrice <state> $state;
        <value> ?priceAux.
      FILTER(REGEX(?name,$medicine,'i'))}
    ",
    "RP":{
        "header": "",
        "body": "O ?name tem um preço máximo de ?preço reais",
        "footer":""
        }
}
```

Neste exemplo (Figura 12), o usuário realiza a pergunta de uma maneira consideravelmente diferente ao padrão conhecido. Deste modo, o *chatbot* tenta solucionar a intenção do usuário apresentando o padrão do *QP* substituindo os valores das *CV*s encontradas pelo *NER* na pergunta original. Ao ter a sugestão confirmada, esta é adicionada como um novo exemplo para este *QAI* (após os valores das *CV*s serem substituídos por seus identificadores, *e.g., "buscopan"* é substituído por $medicine). Contudo, o *chatbot* percebe que ainda falta o valor para a *CV* $state, utilizando assim o tipo inferido da *CV* para realizar sua solicitação. Por fim, após substituir os valores das *CV*s em *SP* e executá-la no *endpoit SPARQL*, o *chatbot* retorna a resposta seguindo *RP*.

Figura 12 – Uso do diálogo de clarificação no *Telegram*.



Caso a mesma pergunta seja feita novamente, o *chatbot* terá uma maior confiança na classificação da intenção do usuário, dispensando o diálogo de clarificação. Esta característica permite que o *chatbot* adapte-se aos termos e "formas de perguntar" de diversos tipos de usuários. Isto garante ao sistema um conjunto de treino muito maior e mais diverso do que o que seria possível de ser implementado pelo desenvolvedor através do uso de expressões regulares, como na implementação original de *MediBot*.

5.3 Avaliação de Usabilidade

Neste trabalho, o método de avaliação baseado em tarefas descrito em (KONSTANTINOVA; ORASAN, 2013) foi utilizado para medir o grau de usabilidade do *MediBot*. Nesse método, são definidos conjuntos de tarefas que são solicitadas a serem executadas por voluntários.

Para avaliar a utilidade prática do *MediBot*, foi elaborado um conjunto de 6 perguntas sobre as informações contidas nas fontes. As perguntas foram feitas para usuários voluntários não envolvidos com o projeto. Nesta avaliação, os usuários utilizaram a ferramenta para responder as questões. As perguntas podem ser divididas em três níveis de dificuldade: (1) Fácil, pode ser respondido com uma única interação com o *chatbot*; (2) Médio, pode ser respondido com pelo menos duas interações; e (3) difícil, exigindo mais de duas interações com o *chatbot*.

As perguntas foram feitas por meio de entrevistas presenciais com os voluntários. Dez voluntários, quatro homens e seis mulheres participaram do estudo. A idade média dos participantes foi de 31 anos, com idade mínima de 22 anos e máxima de 63 anos. Nenhum deles afirmou ter conhecimento técnico sobre o campo de medicamentos. Além disso, dois dos participantes possuíam conhecimento técnico em tecnologia da informação, dentre eles um sobre ontologias.

Tabela 5 – Resultados da avaliação de usabilidade.

ID da Questão	Questão	Tempo médio(s)	Taxa de Sucesso(%)	Dificuldade
Q001	O que é um medicamento tarja preta?	30,53	100	1
Q002	Quais são os riscos do Tylenol?	51,2	100	1
Q003	Quais medicamente possuem o mesmo princípio ativo do Buscopan?	142,10	80	2
Q004	Qual é o preço máximo do Buscopan?	358,96	20	2
Q005	Qual é a relação entre uma substância e uma apresentação	152,05	70	3
Q006	Qual estado do menor preço máximo para o medicamento Orencia?	320,34	10	3
Média		175,86	63,33	

Fonte: Próprio autor.

Como critério de avaliação, foram utilizados três aspectos. Os primeiros critérios foram o tempo necessário para resolver as questões, a taxa de acertos, e finalmente, a opinião pessoal de cada avaliado. Na tabela 5 é apresentado o resultado da avaliação para cada pergunta e sua média final.

Vale ressaltar que apenas as consultas Q001 e Q002 puderam ser realizadas sem conhecimento prévio sobre a ontologia e os tipos de consultas e seus fluxos, enquanto as outras precisavam de consultas sobre essas informações. Esse fato já era esperado, uma vez que *MediBot* possui um conjunto limitado de perguntas predefinidas (que incluem as consultas Q001 e Q002).

Outro ponto interessante é que as consultas Q004 e Q006 demoraram consideravelmente mais, além de terem uma menor taxa de sucesso. Esse fato pode ser explicado porque suas respostas não são representadas de maneira factual na base. No caso de Q004, havia a necessidade de uma operação de comparação, que em geral os usuários tentavam comparar todos os preços existentes, o que exigia muitas interações, levando a uma alta taxa de desistência. Havia a possibilidade de diminuir o número de possibilidades, levando em consideração características da apresentação. Já no caso da consulta Q006, havia uma necessidade de entender como o conceito “estado’ estava relacionado ao “preço’, onde mais uma vez os usuários tentavam fazer todas as comparações. No entanto, tal questão poderia ser resolvida apenas observando o menor valor para impostos e quais os estados que o adotaram.

Durante o relato de opiniões sobre o uso da ferramenta, os principais pontos apresentados foram que a imagem da ontologia e os exemplos de consultas foram muito úteis. Além da preferência por perguntas rápidas e diretas.

6 RESULTADOS E DISCUSSÕES

Neste capítulo é discutido como o *framework CONQUEST* impacta no processo de desenvolvimento de *chatbots* para TBIQA sobre EKGs. Para aferir o impacto da abordagem abordagem sugerida, as discussões a seguir utilizam o estudo de caso apresentado no Capítulo 5. Nas seções a seguir é discutido como o *framework CONQUEST* trata cada um dos passos para a construção de um sistema de TBIQA para KGs apresentados na Seção 4.1.

6.1 Construção dos Padrões Suportados pelo Chatbot

O processo construção dos padrões mostrou-se bastante natural e exigiu que o desenvolvedor inserisse apenas algumas variações da questão em linguagem natural. O arquivo JSON que contém os *QAI*s usados e os dados necessários para implantar uma instância de nossa implementação do *MediBot* com *CONQUEST* estão disponíveis ao público [1]. O exemplo dado a seguir será considerado nas discussões apresentadas nesta seção.

```
{
    "QPs": [
      "Qual o preço máximo para o medicamento \$medicine no \$state?"
    ],
    "SP": "
    SELECT ?name (MAX(?priceAux) as ?price)  WHERE{
      ?s a <Medicine>;
        rdfs:label ?name;
        <Price> ?appliedPrice.
      ?appliedPrice <state> $state;
        <value> ?priceAux.
      FILTER(REGEX(?name,$medicine,'i'))}
    ",
    "RP":{
        "header": "",
        "body": "O ?name tem um preço máximo de ?preço reais",
        "footer":""
        }
}
```

6.2 Criação do Mecanismo de Processamento de Linguagem Natural

Nesta etapa o desenvolvedor deve apenas selecionar o idioma suportado pelo seu *chatbot* sendo produzido. *CONQUEST* utiliza a biblioteca *Spacy* para o processamento de linguagem natural. Esta biblioteca oferece o suporte a mais de 53 idiomas. Contudo, o suporte a cada idioma está em um estágio diferente, enquanto a biblioteca alcança ótimos resultados para o Inglês, o mesmo não pode ser dito para o idioma Português do Brasil. Neste trabalho, o modelo de *Word Embeddings* utilizado foi o modelo *GloVe* de 100 dimensões produzido por

[1] https://github.com/CaioViktor/CONQUEST_medibot_experiments

(HARTMANN *et al.*, 2017). Este modelo foi então carregado no *SpaCy*, deste modo aproveitando todo o *pipeline* de processamento da biblioteca. O modelo final para o Português do Brasil produzido pode ser baixado no link[2].

6.3 Criação do Mecanismo de Classificação de Padrão

A etapa de construção do classificador é transparente para o desenvolvedor, com *CONQUEST* já tendo um modelo de classificação padrão. Foram realizados experimentos para selecionar o melhor modelo e avaliar o impacto do uso de recursos semânticos nessa tarefa. Para estes experimentos foram implementados os 8 modelos de consulta respondidos por *MediBot*.

Como um conjunto de treinamento e validação[3], 10 variações de cada pergunta em linguagem natural foram usadas para cada *template* (*QAI*). Para o conjunto de testes[4], foram utilizados 5 exemplos de variações para cada *template*, diferentes daqueles usados no estágio de treinamento/validação. Os resultados apresentados são a média dos testes realizados 10 vezes. O *script* com os experimentos pode ser encontrado no link [5].

Para estes experimentos foram utilizados os algorítimos da biblioteca *scikit-learn* (PEDREGOSA *et al.*, 2011). A biblioteca *scikit-learn* fornece um conjunto de ferramentas e algoritmos para as tarefas de mineração e análise de dados, incluindo um vasto conjunto de algorítimos de treinamento de modelos de ML. Nestes experimentos, foram testados uma ampla gama de classes de modelos de classificação. Onde, cada modelo foi avaliado com o uso dos valores padrões configurados para os seus hiper-parâmetros. Para a realização da representação vetorial de sentenças foi utilizado o modelo padrão pré-treinado de *Word Embeddings* para o Português que acompanha o *framework CONQUEST*.

Para a avaliação dos modelos foi realizado o processo de validação-cruzada (*cross-validation*)[6], onde o número de iterações de *cross-validation* (parâmetro CV) foi configurado para 5. Deste modo, o conjunto de exemplos foi divido em 5 partes iguais, onde cada uma por vez foi adotada como o conjunto de teste e as demais foram utilizadas como conjunto de treino. Ao fim, a média do desempenho dos 5 modelos foi calculada.

Na tabela 6 é possível visualizar os resultados dos modelos treinados sem o uso das *features semânticas* e com o uso destas. No primeiro caso, o melhor modelo observado foi o modelo baseado no classificador de *Multilayer Perceptron* (MLP) com duas camadas ocultas. Este modelo alcança uma pontuação de 0.926 na métrica F1, o que poderia ser considerado um bom resultado. Além disso, com o conjunto correto de hiper-parâmetros estes resultados tendem a melhorar. No entanto, como *CONQUEST* é pensado para simplificar o processo de desenvolvimento de IQAs, a tarefa de refinamento de modelos complexos é tida como indesejável,

[2] https://sourceforge.net/projects/conquest-sqai/files/model_pt-br.tar.xz/download
[3] https://github.com/CaioViktor/CONQUEST_medibot_experiments/blob/master/training_examples.csv
[4] https://github.com/CaioViktor/CONQUEST_medibot_experiments/blob/master/test_examples.csv
[5] https://github.com/CaioViktor/CONQUEST_medibot_experiments/blob/master/experiments_classifier.ipynb
[6] https://scikit-learn.org/stable/modules/cross_validation.html

principalmente levando em conta que este processo adiciona tempo durante a etapa de treinamento (várias combinações diferentes devem ser testadas). Além disso, o tempo necessário apenas para o treinamento do modelo levou em torno de 00.229768 segundos, sendo um dos modelos mais lentos para o treinamento. Como os *chatbots* produzidos por *CONQUEST* utilizam as questões dadas como entrada em tempo de execução como novos exemplos de treinamento, o que resulta no crescimento constante do *dataset*, o custo para o treinamento do modelo é de crítica importância. O impacto do custo no tempo de treinamento torna-se ainda maior em cenários de atualização *online* do modelo (o que nem mesmo é possível no caso de modelos MLP ao utilizar o algoritmo disponível no *scikit-learn*).

Deste modo, como uma tentativa para melhorar o desempenho do mecanismo de classificação, foram adicionadas as *features semânticas* (como apresentado na seção 4.3.4) à entrada do classificador. As *features semânticas* tentam utilizar o conhecimento do domínio para identificar possíveis entidades relevantes para a questão e seus tipos semânticos. Como visto na tabela 6 o uso das *features semânticas*, no geral, apresenta melhorias significativas nos modelos avaliados, tanto para o tempo de convergência do treinamento, quanto para a pontuação da métrica F1. Neste caso, é digno de nota as melhorias do desempenho do modelo *GaussianNB*, este tendo alcançado o melhor desempenho em todos os aspectos medidos.

Quando comparando o melhor modelo treinado com o uso das *features semânticas* (*GaussianNB*) contra o melhor sem estas (*MLPClassifier 2 layers*) é possível conferir uma ligeira melhora de cerca de 5.075%, o que já pode ser considerado um bom resultado. Já quando comparando os resultados na métrica F1 para o modelo *GaussianNB* sem e com *features semânticas*, é possível notar uma melhora de cerca de 38,014%, o que configura uma grande melhora no desempenho do modelo. No entanto, a verdadeira melhora vem ao comparar o tempo de treinamento levado pelos dois melhores modelos (*MLPClassifier 2 layers* e *GaussianNB*), onde ocorre uma redução de 98,280% do tempo necessário, o que significa que o primeiro leva cerca de 58 vezes mais tempo no treinamento do que o segundo.

Em vários cenários diferentes, o modelo *GaussianNB* com uso das *features semânticas* alcançou ótimos resultados, estando sempre entre os modelos mais bem avaliados. Deste modo, este modelo foi escolhido como o modelo padrão utilizado pelo *framework CONQUEST*.

As vantagens do uso de modelos da classe *GaussianNB* vão ainda além, pois esta é uma das classes de modelos mais simples, não havendo a necessidade da configuração de hiperparâmetros, o que simplifica o seu treinamento. Além disso, o *scikit-learn* oferece nativamente a opção de treinamento *online* para este tipo de modelo. Tal característica permite que o *chatbot* adapte-se ao uso em tempo de execução.

Tabela 6 – Avaliações de modelos.

	Without Semantic Features				**With Semantic Features**			
Classifier	**Precision**	**Recall**	**F1-Score**	**Time(secs)**	**Precision**	**Recall**	**F1**	**Time(secs)**
GaussianNB	0.772	0.712	0.705	00.023068	**0.983**	**0.975**	**0.973**	**00.003952**
BernoulliNB	0.606	0.6	0.565	00.262797	0.664	0.687	0.645	00.069785
LogisticRegression	0.8	0.787	0.764	01.301347	0.958	0.937	0.933	00.040490
SVC auto	0.539	0.575	0.520	00.054511	0.749	0.775	0.731	00.009718
SVC linear	0.916	0.875	0.870	00.048965	**0.983**	**0.975**	**0.973**	00.007134
DecisionTreeClassifier	0.545	0.575	0.534	00.056715	0.858	0.875	0.860	00.007359
RandomForestClassifier	0.534	0.562	0.589	00.108051	0.780	0.762	0.714	00.012728
NearestCentroid	0.631	0.625	0.591	00.004763	0.855	0.837	0.822	00.005290
MLPClassifier 1 layer	0.912	0.887	0.867	00.355934	0.966	0.962	0.946	00.198628
MLPClassifier 2 layers	**0.941**	**0.912**	**0.926**	00.229768	0.966	0.962	0.96	00.176730
AdaBoostClassifier	0.170	0.287	0.191	00.106870	0.325	0.45	0.305	00.133780
AdaBoostClassifier (RandomForest)	0.791	0.737	0.762	00.084082	0.895	0.887	0.847	00.090558
KNeighborsClassifier	0.448	0.475	0.418	00.029685	0.792	0.775	0.757	00.005152
Nearest Neighbor	0.707	0.675	0.657	**00.004416**	0.879	0.875	0.86	00.006486
QuadraticDiscriminant Analysis	0.413	0.437	0.387	00.148888	0.413	0.437	0.387	00.006219
Soft VotingClassifier (GaussianNB + LogisticRegression)	0.772	0.712	0.705	00.218400	**0.983**	**0.975**	**0.973**	00.110563

Fonte: Próprio autor.

Ao fim, o modelo selecionado (*GaussianNB* com *semantic features*) foi avaliado contra o conjunto de teste, alcançando o resultado apresentado na Tabela 7.

Tabela 7 – Resultados sobre o conjunto de teste.

Precision	**Recall**	**F1-Score**
0.979167	0.975	0.974747

Fonte: Próprio autor.

6.4 Defnição do Fluxo de Interação

Como apresentado na Seção 4.5, *CONQUEST* possui um *chatflow* padrão pensado para executar a tarefa de TBIQA tratando os problemas de classificação inconclusiva e parâmetros ausentes. Esta característica livra o desenvolvedor de lidar com a programação do fluxo de conversão utilizando técnicas como máquinas de estados, *scripts* de conversação e etc.

Neste exemplo (Figura 12), o usuário realiza a pergunta de uma maneira consideravelmente diferente ao padrão conhecido. Deste modo, o *chatbot* tenta solucionar a intenção do usuário apresentando o padrão da *QP* substituindo os valores das *CV*s encontradas pelo *NER* na pergunta original. Ao ter a sugestão confirmada, esta é adicionada como um novo exemplo para esta *QAI* (após os valores das *CV*s serem substituídos por seus identificadores, e.g., "buscopan" é substituído por *$medicine*). Contudo, o *chatbot* percebe que ainda falta o valor para a *CV $estado*, utilizando assim o tipo inferido da *CV* para realizar sua solicitação. Por fim, após substituir os

valores das *CV*s em *SP* e executá-la no *endpoit SPARQL*, o *chatbot* retorna a resposta seguindo *RP*.

6.5 Criação do Mecanismo de Consulta

A etapa de construção do mecanismo de consulta é totalmente automática. O padrão de consulta SPARQL, *SP* passado no *QAI* é usado para criar a consulta real a ser executada no EKG. Os marcadores de *CV*s presentes em *SP* são preenchidos com os parâmetros passados pelo usuário no momento da consulta. No exemplo atual, *$medicine* e *$state* em *SP* são substituídos por "Buscopan" e "Ceará", respectivamente.

Além disso, o *framework* também constrói automaticamente a resposta em linguagem natural a ser apresenta ao usuário. Isto é realizado ao substituir os valores das variáveis de saída no padrão de resposta, *RP* passado no *QAI* pelos valores retornados pela execução da consulta no EKG. No atual exemplo, as variáveis de saída *?name* e *?price* no "body" de *RP* serão substituías pelos valores das variáveis *?name* e *?price* para cada linha da resposta da consulta SPARQL.

6.6 Criação da Interface de Usuário

O acesso ao *CONQUEST Chatbot* produzido pode ser realizado tanto através de aplicativos de mensageiros instantâneos (e.g., *Telegram, Facebook Messenger, Skype, etc.*), quanto diretamente através do serviço *REST* via requisições HTTP. No primeiro caso, não há a necessidade do desenvolvedor investir esforço na construção da interface de usuário, ocorrendo o reuso da infraestrutura de serviços já consolidados. Isto dispensa a necessidade da instalação de novos aplicativos e adaptação por parte do usuário. No segundo caso, aplicações externas podem reutilizar os serviços oferecidos pelo *chatbot*, desta maneira estes podem ser integrados em serviços maiores, tais como *chatbots* já existentes construídos com ambientes comerciais (e.g., *Dialogflow, chatfuel, etc.*). Neste caso, o *framework CONQUEST* pode prover apenas a habilidade específica de TBIQA para um *chatbot* "maior".

Por fim, o *framework CONQUEST* permite que uma mesma instância de um *chatbot* esteja presente em diferentes meios, rodando a partir de um único código, o que facilita na manutenção e no incremento do serviço.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentado *CONQUEST*, um *framework* que automatiza grande parte do processo de construção de *chatbots* para TBIQA. Os *chatbots* produzidos utilizam o diálogo com o usuário para solucionar os problemas de classificação inconclusiva e a ausência de parâmetros na questão. Além disso, *CONQUEST* utiliza o aprendizado de máquina para aprender novas maneiras de como uma mesma questão pode ser realizada. Tal característica permite que o *chatbot* adapte-se ao uso.

Uma avaliação qualitativa foi realizada para aferir o impacto do uso do *framework* na produção de *chatbots* de TBIQA. Por questão de comparação, o *chatbot Medibot* foi re-implementado com o uso do *framework CONQUEST*. *MediBot* é um *chatbot* para a consulta a um KG sobre medicamentosa e seus riscos apresentado em (AVILA *et al.*, 2019a). A implementação original de *MediBot* foi realizada com o uso de expressões regulares construídas manualmente para a classificação da intenção das questões. Além disso, para a implementação de um novo padrão suportado pelo chatbot havia a necessidade da alteração de diversos pontos do código, o que limita ainda mais o potencial de crescimento da solução. Já com o uso do *framework CONQUEST*, o processo de desenvolvimento mostrou-se bastante natural e rápido, havendo a necessidade apenas do fornecimento do padrão de questão em LN. Além disso, o framework trata automaticamente do processo de desambiguação de intenção e solicitação de parâmetros ausentes, o que não era abordado pela versão orinal do *MediBot*.

Além disso, Experimentos foram executados para avaliar a eficácia dos *chatbots* produzidos por *CONQUEST*. Estes experimentos mostraram que o uso da semântica presente nas consultas SPARQL dos padrões e na base de conhecimento impacta positivamente no treinamento de modelos para a classificação da intenção do usuário. Este impacto é ainda mais nítido para modelos mais simples, onde quando comparando os resultados na métrica F1 para o modelo *GaussianNB* sem e com *features semânticas*, é possível notar a melhora de cerca de 38,014%. Isto configura uma grande melhora no seu desempenho, chegando a alcançar um desempenho de 97,3%. Deste modo, torna-se possível o uso de modelos mais simples como o *GaussianNB* para a classificação das questões. O uso deste modelo traz as vantagens de um baixo custo de tempo para o treinamento (inclusive não havendo a necessidade de otimização de hiperparâmetros) e a possibilidade de atualização do modelo em tempo de execução, o que permite ao modelo adaptar-se durante o uso.

A Tabela 8 apresenta a lista de publicações produzidas no contexto da pesquisa desta dissertação.

Como trabalhos futuros, é planejado o desenvolvimento de um mecanismo para a geração automática de padrões de consulta. Embora *CONQUEST* automatize grande parte do processo de desenvolvimento, ainda é necessário que o desenvolvedor forneça manualmente os padrões suportados como entrada. Deste modo, para que o *chatbot* produzido seja capaz de atender ao maior número de cenários e demandas dos usuários, é necessário que o desenvolvedor

Tabela 8 – Lista de publicações produzidas

Título	Autores	Local de Publicação	Status da Submissão
MediBot: An Ontology based Chatbot for Portuguese Speakers Drug's Users	**AVILA, CAIO**; PEQUENO, VALÉRIA ; VIDAL, VÂNIA ; ROLIM, TULIO ; FRANCO, WELLINGTON ; VENCESLAU, AMANDA ; CALIXTO, ANDERSON ; FELIX DE MOURA, FRANCILDO	21st International Conference on Enterprise Information Systems, 2019	Aceito
MediBot: Um chatbot para consulta de riscos e informações sobre medicamentos	**AVILA, CAIO VIKTOR S.**; ROLIM, TULIO VIDAL ; DA SILVA, JOSÉ WELLINGTON FRANCO ; VIDAL, VANIA MARIA PONTE	XIX Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS), 2019	Aceito
Using chatbots to compare drug prices extracted in real-time from the web	**AVILA, CAIO VIKTOR S.**; VIDAL, VANIA PONTE ; CALDAS, ASLEY ; ROLIM, TULIO VIDAL ; LAVOR, THAYS	25th Brazillian Symposium on Multimedia and the Web - WebMedia '19	Aceito
Um Framework Para a Construção de Chatbots de IQA Baseado em Padrões Sobre Bases de Conhecimento de Domínio Fechado	**AVILA, CAIO VIKTOR**; VIDAL, VANIA	Brazilian Symposium on Databases (SBBD), 2019	Aceito
MediBot: An Ontology-Based Chatbot to Retrieve Drug Information and Compare its Prices	**AVILA, CAIO**; PEQUENO, VALÉRIA ; VIDAL, VÂNIA ; ROLIM, TULIO ; FRANCO, WELLINGTON ; VENCESLAU, AMANDA	Journal of Digital Information Management (JDIM) 2019	Submetido
CONQUEST: A Framework for Building Template-Based IQA Chatbots for Enterprise Knowledge Graphs	**AVILA, CAIO**; FRANCO, WELLINGTON; Gilvan Maia; VIDAL, VÂNIA	25th International Conference on Applications of Natural Language to Information Systems (NLDB2020)	Submetido

Fonte: Próprio autor.

elabore um conjunto de extenso de possíveis cenários padrões. Portanto, é de grande importância que o sistema seja capaz de responder pelo menos às questões mais simples de maneira automática, delegando para o desenvolvedor a tarefa de construir apenas os padrões de questões mais complexos e desafiadores.

REFERÊNCIAS

ABDUL-KADER, S. A.; WOODS, J. Survey on chatbot design techniques in speech conversation systems. **International Journal of Advanced Computer Science and Applications**, The Science and Information (SAI) Organization, v. 6, n. 7, 2015.

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

AL-ZUBAIDE, H.; ISSA, A. A. Ontbot: Ontology based chatbot. In: IEEE. **International Symposium on Innovations in Information and Communications Technology**. [*S. l.*], 2011. p. 7–12.

ALLAM, A. M. N.; HAGGAG, M. H. The question answering systems: A survey. **International Journal of Research and Reviews in Information Sciences (IJRRIS)**, v. 2, n. 3, 2012.

ALLEN, J. **Natural language understanding**. [*S. l.*]: Pearson, 1995.

ATHREYA, R. G.; NGOMO, A.-C. N.; USBECK, R. Enhancing community interactions with data-driven chatbots–the dbpedia chatbot. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Companion of the The Web Conference 2018 on The Web Conference 2018**. [*S. l.*], 2018. p. 143–146.

AVILA, C. V.; CALIXTO, A.; ROLIM., T. V.; FRANCO., W.; VENCESLAU, A.; VIDAL., V. M. P.; PEQUENO., V. M.; MOURA., F. F. D. Medibot: An ontology based chatbot for portuguese speakers drug's users. In: INSTICC. **Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 1: ICEIS,**. [*S. l.*]: SciTePress, 2019. p. 25–36. ISBN 978-989-758-372-8.

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.

AVILA, C. V. S.; VIDAL, V. P.; CALDAS, A.; ROLIM, T. V.; LAVOR, T. Using chatbots to compare drug prices extracted in real-time from the web. In: **Proceedings of the 25th Brazillian Symposium on Multimedia and the Web**. [*S. l.*: *s. n.*], 2019. p. 137–140.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. **Scientific american**, JSTOR, v. 284, n. 5, p. 34–43, 2001.

BIERMANN, L.; WALTER, S.; CIMIANO, P. A guided template-based question answering system over knowledge graphs. In: **Proceedings of the 21st International Conference on Knowledge Engineering and Knowledge Management**. [*S. l.*: *s. n.*], 2018.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data: The story so far. In: **Semantic services, interoperability and web applications: emerging concepts**. [*S. l.*]: IGI Global, 2011. p. 205–227.

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

BRADEŠKO, L.; MLADENIĆ, D. A survey of chatbot systems through a loebner prize competition. In: **Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies**. [*S. l.*: *s. n.*], 2012. p. 34–37.

BRILL, E.; DUMAIS, S.; BANKO, M. An analysis of the askmsr question-answering system. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10**. [*S. l.*], 2002. p. 257–264.

CALDIERA, V. R. B. G.; ROMBACH, H. D. The goal question metric approach. **Encyclopedia of software engineering**, p. 528–532, 1994.

CASSELL, J.; SULLIVAN, J.; CHURCHILL, E.; PREVOST, S. **Embodied conversational agents**. [*S. l.*]: MIT press, 2000.

CERI, S.; BOZZON, A.; BRAMBILLA, M.; VALLE, E. D.; FRATERNALI, P.; QUARTERONI, S. The information retrieval process. In: **Web Information Retrieval**. [*S. l.*]: Springer, 2013. p. 13–26.

CHATSCRIPTNLP. **ChatScript Natural Language tool/dialog manager**. 2018. https://github.com/ChatScript/ChatScript. Accessed: 2018-12-12.

CHEN, H.; CHIANG, R. H.; STOREY, V. C. Business intelligence and analytics: From big data to big impact. **MIS quarterly**, v. 36, n. 4, 2012.

CHO, K.; MERRIËNBOER, B. V.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

CHOWDHURY, G. G. Natural language processing. **Annual review of information science and technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003.

COCCO, R.; ATZORI, M.; ZANIOLO, C. Machine learning of sparql templates for question answering over linkedspending. In: IEEE. **2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)**. [*S. l.*], 2019. p. 156–161.

CORRÊA, A. *et al.* Uma abordagem sobre o uso de medicamentos nos livros didáticos de biologia como estratégia de promoção de saúde. **Ciência & Saúde Coletiva**, SciELO Public Health, v. 18, p. 3071–3081, 2013.

CPYTHON. **pickle Python object serialization**. 2019. https://docs.python.org/3/library/pickle.html. Accessed: 2019-11-25.

CROFT, W. B.; METZLER, D.; STROHMAN, T. **Search engines: Information retrieval in practice**. [*S. l.*]: Addison-Wesley Reading, 2010. v. 520.

DALE, R. The return of the chatbots. **Natural Language Engineering**, Cambridge University Press, v. 22, n. 5, p. 811–817, 2016.

DALRYMPLE, M.; SHIEBER, S. M.; PEREIRA, F. C. Ellipsis and higher-order unification. **Linguistics and philosophy**, Springer, v. 14, n. 4, p. 399–452, 1991.

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.

DAMLJANOVIC, D.; AGATONOVIC, M.; CUNNINGHAM, H. Freya: An interactive way of querying linked data using natural language. In: SPRINGER. **Extended Semantic Web Conference**. [*S. l.*], 2011. p. 125–138.

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.

DORNESCU, I.; ORASAN, C. Interactive qa using the qallme framework. **International Journal of Computational Linguistics and Applications**, v. 1, n. 1-2, p. 233–247, 2010.

EXPLOSION, A. Industrial-strength natural language processing. **Retrieved from Spacy https://spacy. io**, 2019.

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. p. 1608–1618.

FERNANDES, P. C. B.; GUIZZARDI, R. S.; GUIZZARDI, G. Using goal modeling to capture competency questions in ontology-based systems. **Journal of Information and Data Management**, v. 2, n. 3, p. 527–527, 2011.

FLIEDNER, G. Towards natural interactive question answering. In: **Proceedings of the 5th International Conference on Language Resources and Evaluation-LREC-2006. Genoa, Italy**. [*S. l.*: *s. n.*], 2006.

FOUNDATION, A. S. **OpenNLP The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text.** 2004. https://opennlp.apache.org/. Accessed: 2019-01-21.

FRISCHMUTH, P.; KLÍMEK, J.; AUER, S.; TRAMP, S.; UNBEHAUEN, J.; HOLZWEISSIG, K.; MARQUARDT, C.-M. Linked data in enterprise information integration. **Semantic Web**, p. 1–17, 2012.

GANTZ, J.; REINSEL, D. Extracting value from chaos. **IDC iview**, v. 1142, n. 2011, p. 1–12, 2011.

GARBADE, D. M. J. **A Simple Introduction to Natural Language Processing**. 2018. https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/. Accessed: 2020-01-06.

GERBER, D.; NGOMO, A.-C. N. Extracting multilingual natural-language patterns for rdf predicates. In: SPRINGER. **International Conference on Knowledge Engineering and Knowledge Management**. [*S. l.*], 2012. p. 87–96.

GOMEZ-PEREZ, J. M.; PAN, J. Z.; VETERE, G.; WU, H. Enterprise knowledge graph: An introduction. In: **Exploiting linked data and knowledge graphs in large organisations**. [*S. l.*]: Springer, 2017. p. 1–14.

GOOGLE. **Assistant Personal Assistant**. 2018. https://assistant.google.com/. Accessed: 2018-12-11.

GOOGLE. **Google Search Engine**. 2018. https://www.google.com.br/. Accessed: 2018-12-11.

GOYAL, P.; FERRARA, E. Graph embedding techniques, applications, and performance: A survey. **Knowledge-Based Systems**, Elsevier, v. 151, p. 78–94, 2018.

GROUP, R. W. **RDF Resource Description Framework**. 2004. https://www.w3.org/RDF/. Accessed: 2018-12-17.

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowledge acquisition**, Elsevier, v. 5, n. 2, p. 199–220, 1993.

GUI-YANG, J.; FU-ZAI, L.; ZHAN-QIN, X. Enterprise information integration based on knowledge graph and semantic web technology. **Journal of Southeast University (Natural Science Edition)**, v. 44, n. 2, p. 250–255, 2014.

HAKIMOV, S.; UNGER, C.; WALTER, S.; CIMIANO, P. Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In: SPRINGER. **International Conference on Applications of Natural Language to Information Systems**. [*S. l.*], 2015. p. 103–109.

HARABAGIU, S.; HICKL, A.; LEHMANN, J.; MOLDOVAN, D. Experiments with interactive question-answering. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 43rd annual meeting on Association for Computational Linguistics**. [*S. l.*], 2005. p. 205–214.

HARABAGIU, S. M.; MOLDOVAN, D. I.; CLARK, C.; BOWDEN, M.; HICKL, A.; WANG, P. Employing two question answering systems in trec 2005. In: **In Proceedings of the Fourteenth Text REtrieval Conference**. [*S. l.*: *s. n.*], 2005.

HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; RODRIGUES, J.; ALUISIO, S. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **arXiv preprint arXiv:1708.06025**, 2017.

HEATH, T.; BIZER, C. Linked data: Evolving the web into a global data space. **Synthesis lectures on the semantic web: theory and technology**, Morgan & Claypool Publishers, v. 1, n. 1, p. 1–136, 2011.

HENDRIX, G. G. Natural-language interface. **American Journal of Computational Linguistics**, v. 8, n. 2, p. 56–61, 1982. Disponível em: https://www.aclweb.org/anthology/J82-2002.

HENDRIX, G. G.; SACERDOTI, E. D.; SAGALOWICZ, D.; SLOCUM, J. Developing a natural language interface to complex data. **ACM Transactions on Database Systems (TODS)**, ACM, v. 3, n. 2, p. 105–147, 1978.

HICKL, A.; WANG, P.; LEHMANN, J.; HARABAGIU, S. Ferret: interactive question-answering for real-world environments. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the COLING/ACL on Interactive presentation sessions**. [*S. l.*], 2006. p. 25–28.

HIRSCHMAN, L.; GAIZAUSKAS, R. Natural language question answering: the view from here. **natural language engineering**, Cambridge University Press, v. 7, n. 4, p. 275–300, 2001.

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.

IBM. **Business chatbots 3 types of business chatbots you can build**. 2017. https://www.ibm.com/blogs/watson/2017/12/3-types-of-business-chatbots-you-can-build/. Accessed: 2019-01-16.

INC, A. **Siri Personal Assistant**. 2018. https://www.apple.com/siri/. Accessed: 2018-12-11.

JIANG, H. Confidence measures for speech recognition: A survey. **Speech communication**, Elsevier, v. 45, n. 4, p. 455–470, 2005.

KANNAN, A.; KURACH, K.; RAVI, S.; KAUFMANN, T.; TOMKINS, A.; MIKLOS, B.; CORRADO, G.; LUKACS, L.; GANEA, M.; YOUNG, P. *et al.* Smart reply: Automated response suggestion for email. In: ACM. **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [*S. l.*], 2016. p. 955–964.

KAUFMANN, E.; BERNSTEIN, A. How useful are natural language interfaces to the semantic web for casual end-users? In: **The Semantic Web**. [*S. l.*]: Springer, 2007. p. 281–294.

KAUFMANN, E.; BERNSTEIN, A.; ZUMSTEIN, R. Querix: A natural language interface to query ontologies based on clarification dialogs. In: CITESEER. **5th International Semantic Web Conference (ISWC 2006)**. [*S. l.*], 2006. p. 980–981.

KONSTANTINOVA, N.; ORASAN, C. Interactive question answering. In: **Emerging Applications of Natural Language Processing: Concepts and New Research**. [*S. l.*]: IGI Global, 2013. p. 149–169.

KUMAR, V.; JOSHI, S. Incomplete follow-up question resolution using retrieval based sequence to sequence learning. In: ACM. **Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval**. [*S. l.*], 2017. p. 705–714.

KWOK, C.; ETZIONI, O.; WELD, D. S. Scaling question answering to the web. **ACM Transactions on Information Systems (TOIS)**, ACM, v. 19, n. 3, p. 242–262, 2001.

LEARN, S. **F1 sklearn.metrics.f1_score**. 2019. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html. Accessed: 2020-01-06.

LEARN, S. **Precision sklearn.metrics.precision_score**. 2019. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html. Accessed: 2020-01-06.

LEARN, S. **Recall sklearn.metrics.recall_score**. 2019. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html. Accessed: 2020-01-06.

LI, H.; MIN, M. R.; GE, Y.; KADAV, A. A context-aware attention network for interactive question answering. In: ACM. **Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [*S. l.*], 2017. p. 927–935.

LI, Y.; YANG, T. Word embedding for understanding natural language: a survey. In: **Guide to Big Data Applications**. [*S. l.*]: Springer, 2018. p. 83–104.

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.

MAGAZINE, C. **Chatbot Report 2018 Global Trends and Analysis**. 2018. https://chatbotsmagazine.com/chatbot-report-2018-global-trends-and-analysis-4d8bbe4d924b. Accessed: 2019-01-15.

MALLETT, R.; HAGEN-ZANKER, J.; SLATER, R.; DUVENDACK, M. The benefits and challenges of using systematic reviews in international development research. **Journal of development effectiveness**, Taylor & Francis, v. 4, n. 3, p. 445–455, 2012.

MANNING, C. D.; MANNING, C. D.; SCHÜTZE, H. **Foundations of statistical natural language processing**. [*S. l.*]: MIT press, 1999.

MAYO, M. **NLP The Main Approaches to Natural Language Processing Tasks**. 2018. https://www.kdnuggets.com/2018/10/main-approaches-natural-language-processing-tasks.html. Accessed: 2018-12-17.

MICROSOFT. **Bing Search Engine**. 2018. https://www.bing.com/. Accessed: 2018-12-11.

MICROSOFT. **Cortana Personal Assistant**. 2018. https://www.microsoft.com/en-us/cortana. Accessed: 2018-12-11.

MICROSOFT. **Satori Understand Your World with Bing**. 2019. https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/. Accessed: 2020-01-03.

MILLER, G. A. Wordnet: a lexical database for english. **Communications of the ACM**, ACM, v. 38, n. 11, p. 39–41, 1995.

MITKOV, R. **Anaphora resolution: the state of the art**. [*S. l.*]: Citeseer, 1999.

MONGODB, I. **MongoDB The database for modern applications**. 2019. https://www.mongodb.com/. Accessed: 2019-11-25.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. **Lingvisticae Investigationes**, John Benjamins, v. 30, n. 1, p. 3–26, 2007.

OWDA, M.; BANDAR, Z.; CROCKETT, K. Conversation-based natural language interface to relational databases. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops**. [*S. l.*], 2007. p. 363–367.

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

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 40th annual meeting on association for computational linguistics**. [*S. l.*], 2002. p. 311–318.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PETHERBRIDGE, N. **RiveScript What is RiveScript**. 2019. https://www.rivescript.com/about. Accessed: 2019-01-29.

QUARTERONI, S. Personalized, interactive question answering on the web. In: **Coling 2008: Proceedings of the workshop on Knowledge and Reasoning for Answering Questions**. [*S. l.*: *s. n.*], 2008. p. 33–40.

QUARTERONI, S.; MANANDHAR, S. User modeling for adaptive question answering and information retrieval. In: **FLAIRS Conference**. [*S. l.*: *s. n.*], 2006. p. 776–781.

QUARTERONI, S.; MANANDHAR, S. A chatbot-based interactive question answering system. **Decalog 2007**, v. 83, 2007.

QUARTERONI, S.; MANANDHAR, S. Designing an interactive open-domain question answering system. **Natural Language Engineering**, Cambridge University Press, v. 15, n. 1, p. 73–95, 2009.

QUENEAU, P. *et al.* Emergency department visits caused by adverse drug events. **Drug Safety**, Springer, v. 30, n. 1, p. 81–88, 2007.

RANI, M.; MUYEBA, M. K.; VYAS, O. A hybrid approach using ontology similarity and fuzzy logic for semantic question answering. In: **Advanced Computing, Networking and Informatics-Volume 1**. [*S. l.*]: Springer, 2014. p. 601–609.

RDFLIB. **SPARQL Wrapper SPARQL Endpoint interface to Python**. 2019. https://rdflib.github.io/sparqlwrapper/. Accessed: 2019-11-26.

REITER, E.; DALE, R. **Building natural language generation systems**. [*S. l.*]: Cambridge university press, 2000.

REN, Y.; PARVIZI, A.; MELLISH, C.; PAN, J. Z.; DEEMTER, K. V.; STEVENS, R. Towards competency question-driven ontology authoring. In: SPRINGER. **European Semantic Web Conference**. [*S. l.*], 2014. p. 752–767.

SAHA, D.; FLORATOU, A.; SANKARANARAYANAN, K.; MINHAS, U. F.; MITTAL, A. R.; ÖZCAN, F. Athena: an ontology-driven system for natural language querying over relational data stores. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 9, n. 12, p. 1209–1220, 2016.

SCRAPINGHUB. **dateparser Date parsing library designed to parse dates from HTML pages**. 2019. https://pypi.org/project/dateparser/. Accessed: 2019-11-25.

SHAH, H.; WARWICK, K.; VALLVERDÚ, J.; WU, D. Can machines talk? comparison of eliza with modern dialogue systems. **Computers in Human Behavior**, Elsevier, v. 58, p. 278–295, 2016.

SHAWAR, B. A.; ATWELL, E. **A comparison between Alice and Elizabeth chatbot systems**. [*S. l.*]: University of Leeds, School of Computing research report 2002.19, 2002.

SHEFFIELD, T. U. of. **GATE General Architecture for Text Engineering**. 1995. https://gate.ac.uk/. Accessed: 2019-01-21.

SHISHTLA, P. M.; PINGALI, P.; VARMA, V. A character n-gram based approach for improved recall in indian language ner. In: **Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages**. [*S. l.*: *s. n.*], 2008.

SINITOX. **Sistema Nacional de Informações Toxicológicas Registro de intoxicações no Brasil**. 2016. Acessado: 10-02-2019. Disponível em: https://sinitox.icict.fiocruz.br/sites/sinitox.icict.fiocruz.br/files/Brasil1_1.pdf.

SMILEY, D.; PUGH, D. E. **Apache Solr 3 Enterprise Search Server**. [*S. l.*]: Packt Publishing Ltd, 2011.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: **Advances in neural information processing systems**. [*S. l.*: *s. n.*], 2014. p. 3104–3112.

TANG, L. R.; MOONEY, R. J. Using multiple clause constructors in inductive logic programming for semantic parsing. In: SPRINGER. **European Conference on Machine Learning**. [*S. l.*], 2001. p. 466–477.

TAPSAI, C.; MEESAD, P.; HARUECHAIYASAK, C. Tls-art: Thai language segmentation by automatic ranking trie. In: **9th International Conference Autonomous Systems**. [*S. l.*: *s. n.*], 2016.

VOORHEES, E. M. Overview of the trec 2003 question answering track. In: CITESEER. **TREC**. [*S. l.*], 2003. v. 2003, p. 54–68.

W3. **OWL OWL Web Ontology Language**. 2013. https://www.w3.org/OWL/. Accessed: 2019-01-03.

W3C. **Semantic Web W3C Semantic Web Frequently Asked Questions**. 2009. https://www.w3.org/2001/sw/SW-FAQ#What1. Accessed: 2018-12-14.

WALKER, M.; KAMM, C.; LITMAN, D. Towards developing general models of usability with paradise. **Natural Language Engineering**, Cambridge University Press, v. 6, n. 3-4, p. 363–377, 2000.

WESTON, J.; BORDES, A.; CHOPRA, S.; RUSH, A. M.; MERRIËNBOER, B. van; JOULIN, A.; MIKOLOV, T. Towards ai-complete question answering: A set of prerequisite toy tasks. **arXiv preprint arXiv:1502.05698**, 2015.

WHO *et al.* **Safety of medicines: a guide to detecting and reporting adverse drug reactions**. [*S. l.*], 2002.

YANG, P.; YANG, Y. H.; ZHOU, B. B.; ZOMAYA, A. Y. A review of ensemble methods in bioinformatics. **Current Bioinformatics**, Bentham Science Publishers, v. 5, n. 4, p. 296–308, 2010.

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.

APÊNDICE A – CONJUNTOS DE DADOS UTILIZADOS NO TREINAMENTO E TESTE DO CLASSIFICADOR

Nesta Seção são apresentados os *datasets* utilizados como conjuntos de treino e teste do classificador. As Tabelas 9-15 apresentam os exemplos utilizados na fase de treinamento do modelo. As Tabelas 17-20 apresentam os exemplos utilizados na fase de teste do modelo.

Tabela 9 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
quais medicamentos possuem o princípio ativo $OOV?	0
o $OOV é o princípio ativo de quais medicamentos?	0
indique medicamentos com a substancia $OOV	0
retorne remédio a base de $OOV	0
quero saber o medicamentos com $OOV	0
Quais medicamentos possuem a substância $OOV?	0
retorne os medicamentos com princípios ativos $OOV	0
Busque os medicamentos com o $OOV	0
Quais medicamentos possuem a substancia $OOV	0
Rertorne medicamentos que contenham a substancia $OOV	0

Fonte: Próprio autor.

Nota: Parte 1.

Tabela 10 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
quero saber a defingção de $OOV	1
O que significa $OOV	1
defina $OOV	1
fale sobre $OOV	1
me diga o que é um $OOV	1
explique o que é $OOV	1
o que é um $OOV	1
qual é a definição de $OOV	1
você poderia me falar sobre $OOV	1
busque a definição de $OOV	1

Fonte: Próprio autor.

Nota: Parte 2.

Tabela 11 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
Fale sobre o medicamento $OOV	2
O que você sabe sobre o remédio $OOV	2
Pesquise informações sobre o medicamento $OOV	2
Quais dados você sabe sobre o remédio $OOV	2
você poderia me falar sobre o medicameno $OOV?	2
O que você sabe sobre o remédio $OOV	2
Pesquise informações sobre o medicamento $OOV	2
Me informe sobre o medicamento $OOV	2
Quais dados você sabe sobre o remédio $OOV	2
você poderia me falar sobre o medicameno $OOV?	2

Fonte: Próprio autor.

Nota: Parte 3.

Tabela 12 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
Quais são os riscos do medicamento $OOV?	3
Quais são os riscos do remédio $OOV?	3
fale sobre os riscos do $OOV	3
devo ter algum cuidado ao tomar o medicamento $OOV?	3
Cite possívei motivos para não tomar o medicamento $OOV	3
O remédio $OOV tem algum risco?	3
Posso tomar o medicamento $OOV?	3
Cite possívei motivos para não tomar o medicamento $OOV	3
Quais as contra-indicações do $OOV?	3
Em que casos o medicamento $OOV é contraindicado	3

Fonte: Próprio autor.

Nota: Parte 4.

Tabela 13 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
Quais são as apresentações do medicamento $OOV?	4
Liste as apresentações do medicamento $OOV	4
Diga quais são as apresentações do medicamento $OOV	4
Em quais formatos eu posso comprar o medicamento $OOV?	4
O medicamento $OOV pode ser encontrado em quais apresentações	4
Me diga as apresentações do remédio $OOV?	4
Quais apresentações estão disponíveis para o medicamento $OOV?	4
Em quais apresentações eu posso comprar o remédio $OOV?	4
Me dê as apresentações do medicamento $OOV?	4
O medicamento $OOV pode ser encontrado em quais apresentações	4

Fonte: Próprio autor.

Nota: Parte 5.

Tabela 14 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
Retorne as informações sobre o medicamento com o código de barras $OOV	5
Fale sobre a apresentação com o código de barras $OOV	5
O que você sobre sobre o medicamento de código de barras $OOV	5
Fale sobre a apresentação com o ean $OOV	5
Você pode me dar informações sobre a apresentação com código de barras $OOV	5
Retorne as informações sobre o medicamento de ean $OOV	5
Fale sobre a apresentação com o ean $OOV	5
O que você sobre sobre o medicamento de ean $OOV	5
Você pode me dar informações sobre a apresentação com código de barras $OOV	5
Busque uma apresentação pelo código de barras $OOV	5

Fonte: Próprio autor.
Nota: Parte 6.

Tabela 15 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
Fale sobre a apresentação com o código de barras $OOV em $OOV	6
O que você sobre sobre o medicamento de código de barras $OOV em $OOV	6
Me dê informações sobre o código $OOV em $OOV	6
Retorne as informações sobre o medicamento de ean $OOV em $OOV	6
Fale sobre a apresentação com o ean $OOV em $OOV	6
Retorne as informações sobre o medicamento de ean $OOV em $OOV	6
Fale sobre a apresentação com o ean $OOV em $OOV	6
O que você sobre sobre o medicamento de ean $OOV em $OOV	6
Você pode me dar informações sobre a apresentação com código de barras $OOV em $OOV	6
Busque uma apresentação pelo código de barras $OOV em $OOV	6

Fonte: Próprio autor.
Nota: Parte 7.

Tabela 16 – Dataset utilizado como conjunto de treino para o classificador

Padrão	Classe
quanto custa um $OOV?	7
Qual o preço do $OOV?	7
Me diga os preços das apresentações do medicamento $OOV	7
Quais os preços das apresentações do $OOV?	7
Você poderia me informar os preços máximos do $OOV?	7
Quanto custa um $OOV?	7
Me informe o valor de um $OOV	7
Quais os preços máximos para as apresentações do medicamento?	7
quais os preços das apresentações do $OOV?	7
Diga os preços de um $OOV	7

Fonte: Próprio autor.

Nota: Parte 8.

Tabela 17 – Dataset utilizado como conjunto de teste para o classificador

Padrão	Classe
quais outro medicamentos possuem o mesmo princípio ativo do $OOV?	0
me diga os medicamentos que compartilham o princío ativo com o $OOV	0
quais são as possíveis alternativas ao $OOV?	0
cite os remédios com a mesma substância do $OOV	0
o $OOV tem o mesmo princípio ativo de quais medicamentos?	0
o que é um $OOV?	1
descreva um $OOV	1
gostaria de saber mais sobre um $OOV	1
fale sobre o termo $OOV	1
explique o conceito $OOV	1

Fonte: Próprio autor.
Nota: Parte 1.

Tabela 18 – Dataset utilizado como conjunto de teste para o classificador

Padrão	Classe
dê informações sobre o medicamento $OOV	2
o que você sabe sobre o $OOV?	2
quero saber informações sobre o remédio $OOV	2
fale sobre o medicamento $OOV	2
informe-me sobre o $OOV	2
quais riscos o $OOV possui?	3
posso ter problemas ao tomar o $OOV?	3
o medicamento $OOV possui contraindicações?	3
é seguro tomar o $OOV?	3
quais são os riscos do $OOV?	3

Fonte: Próprio autor.

Nota: Parte 2.

Tabela 19 – Dataset utilizado como conjunto de teste para o classificador

Padrão	Classe
quais são as apresentações do medicamento $OOV?	4
em quais apresentações o $OOV pode ser encontrado	4
em quais formatos o $OOV está disponível?	4
quais são as diferentes apresentações do $OOV?	4
quais são as variações do remédio $OOV disponíveis para a compra?	4
fale sobre o medicamento com o código de barras $OOV	5
quero informações sobre o remédio com o código $OOV	5
o que você sabe sobre o código de barras $OOV?	5
o que é o produto com código $OOV?	5
explique o medicamento com código $OOV	5

Fonte: Próprio autor.

Nota: Parte 3.

Tabela 20 – Dataset utilizado como conjunto de teste para o classificador

Padrão	Classe
qual o preço máximo para o produto de código $OOV na localidade $OOV	6
qual o maior preço permitido para o remédio com código de barras $OOV aqui em $OOV?	6
qual é maior valor que o medicamento de código $OOV pode custar em $OOV?	6
gostaria de saber o custo máximo permitido para o medicamento com código de barras $OOV	6
quanto é o máximo que devo pagar para o remédio com o código de barras $OOV na região da $OOV?	6
quais o preços das apresentações do $OOV	7
quanto custa cada apresentação do medicamento $OOV?	7
quanto eu pagaria em cada apresentação do remédio $OOV?	7
diga os preços das apresentações do $OOV	7
por quanto sai cada apresentação do $OOV	7

Fonte: Próprio autor.
Nota: Parte 4.