RAFAEL HENRICO DE SOUSA

DESENVOLVIMENTO DE UM MODELO INTELIGENTE CAPAZ DE RESPONDER QUESTÕES CONSULTANDO BANCOS DE DADOS ESTRUTURADOS (KBS)

LAVRAS – MG

2022

RAFAEL HENRICO DE SOUSA

DESENVOLVIMENTO DE UM MODELO INTELIGENTE CAPAZ DE RESPONDER QUESTÕES CONSULTANDO BANCOS DE DADOS ESTRUTURADOS (KBS)

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, linha de pesquisa em Sistemas Inteligentes, para a obtenção do título de Mestre.

Prof. DSc. Bruno Henrique Groenner Barbosa
Orientador

Prof. DSc. Danton Diego Ferreira
Coorientador

LAVRAS – MG
2022

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Sousa, Rafael Henrico de.
Desenvolvimento de Modelos Inteligentes Capazes de Responder Questões Utilizando Bancos de Dados Estruturados (KBs) / Rafael Henrico de Sousa. - 2022.
102 p. : il.

Orientador(a): Bruno Henrique Groenner Barbosa.
Coorientador(a): Danton Diego Ferreira.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2022.
Bibliografia.

1. Knowledge Base Question Answering. 2. Chatbots e Assistentes Virtuais. 3. Inteligência Artificial. I. Barbosa, Bruno Henrique Groenner. II. Ferreira, Danton Diego. III. Título.

RAFAEL HENRICO DE SOUSA

DESENVOLVIMENTO DE UM MODELO INTELIGENTE CAPAZ DE RESPONDER QUESTÕES CONSULTANDO BANCOS DE DADOS ESTRUTURADOS (KBS)

DEVELOPMENT OF AN INTELLIGENT MODEL CAPABLE OF ANSWERING QUESTIONS BY CONSULTING STRUCTURED DATABASES (KBS)

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, linha de pesquisa em Sistemas Inteligentes, para a obtenção do título de Mestre.

APROVADA em 29 de Abril de 2022.

Prof. DSc. Bruno Henrique Groenner Barbosa	UFLA
Prof. DSc. Danton Diego Ferreira	UFLA
Profa. DSc. Renata Lopes Rosa	UFLA
Prof. DSc. Giovani Bernardes Vitor	UNIFEI - UNI

Prof. DSc. Bruno Henrique Groenner Barbosa
Orientador

Prof. DSc. Danton Diego Ferreira
Co-Orientador

LAVRAS – MG
2022

Dedico este trabalho primeiramente a Deus, por ser fonte de minha vida. Dedico ainda a minha mãe Maria Emília, ao meu pai Valdevino "In Memorian", aos meus irmãos e a todos que acreditaram e acreditam em mim, que me apoiaram ao longo do caminho.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ser essencial em minha vida.

Agradeço aos professores que me conduziram, dia a dia, na estrada do saber, em especial, ao meu professor orientador Bruno Henrique Groenner Barbosa, e ao meu professor coorientador Danton Diego Ferreira, por aceitarem este desafio, pelas orientações, paciência e disposição para ajudar.

Agradeço ao meu pai Valdevino Alves "In Memorian" e minha mãe Maria Emília por seus constantes ensinamentos, amor, e fé em minha capacidade.

Agradeço aos meus irmãos Frederico Lúcio e Waleska de Fátima por sempre estarem ao meu lado quando preciso.

Agradeço a todos os amigos, que partilharam desta jornada, que compartilharam conhecimentos, que trouxeram alegria aos momentos tristes, e que fizeram dos momentos alegres, momentos completos.

Agradeço em especial aos amigos Aleson Gleik e Fernanda Costa, do grupo de pesquisa "Artificial Intelligence and Automation" (AIA) da UFLA, pela constante troca de conhecimentos e companhia na jornada.

Agradeço a Omnilogic Inteligência S/A pelo fomento e apoio a pesquisa, e a sua equipe de pesquisa pelos direcionamentos, conhecimentos e informações fornecidos.

Agradeço em fim, à Universidade Federal de Lavras pela oportunidade da realização do curso de pós-graduação.

Muito Obrigado!!!

Those who can imagine anything, can create the impossible.
(Alan Turing)

RESUMO

O mercado de vendas virtuais (*E-commerce*) tem se expandido muito atualmente devido à facilidade e praticidade proporcionadas por esta via de compra, e pelo fato de que as tecnologias vem se tornando cada vez mais acessíveis. Com o aumento dos consumidores que utilizam este método de compra, a implantação de assistentes virtuais por empresas pode agregar benefícios para ambos os lados da negociação consumidores-empresas, uma vez que o uso de assistentes virtuais pode permitir a automação de tarefas que envolvem meios de comunicação, acelerando-se assim a resolução de problemas e aumentando-se a produtividade para atendimento ao cliente, além de ser capaz de fornecer experiências personalizadas adequadas a cada consumidor. Neste trabalho, são abordados conceitos relevantes ao tópico de *Knowledge Base Question Answering (KB-QA)* e meios de desenvolver-se um modelo *KB-QA* que seja capaz de responder às questões do usuário fornecidas em linguagem natural a partir de informações contidas em bases de conhecimento (*KBs*). Para este fim, são abordadas duas linhas de pesquisa, os modelos completos, capazes de lidar diretamente com a questão em linguagem natural e fornecer respostas baseados na *KB*; e os modelos compostos, que possuem um método para conversão da questão da linguagem natural em uma *query* e um método de consulta a *KB* por meio de *queries*, sendo a *query* constituída de triplas da forma (sujeito, predicado, objeto). A fim de se desenvolver um modelo *KB-QA* com bom desempenho, foi proposto neste trabalho um *Ensemble* de modelos onde são analisados e selecionados métodos de *KB-QA* disponíveis na literatura. Na análise dos modelos *KB-QA* compostos, são aplicadas bases de dados de áreas variadas tais como a *COUNTRIES* e a *FB15k-237*, visando determinar o potencial dos modelos para diferentes estruturas das Knowledge Bases baseado em triplas e, então, utiliza-se a base de dados *WikiMovies* para a análise dos modelos compostos por inteiro. Já para análise dos modelos *KB-QA* completos emprega-se o banco de questões *WebQuestionsSP* que é desenvolvido para ser respondido por meio de consultas à *KB Freebase*. Como principal resultado deste trabalho, são gerados três diferentes *Ensembles*, denominados *Ensemble* simples, *Ensemble* com contramedida e *Ensemble* com contramedida total, que são estado-da-arte para a tarefa de *KB-QA* considerando-se o banco de dados *WebQuestionsSP* com F1-scores de 75,40%, 78,72% e 81,43% respectivamente. Além disso, são apresentados resultados envolvendo o desenvolvimento de um modelo *KB-QA* composto, aqui denominado modelo composto *Parot-MINERVA*.

Palavras-chave: Knowledge Base Question Answering. Assistentes Virtuais. Processamento de Linguagem Natural. Inteligencia Artificial. Chatbots. E-commerce.

ABSTRACT

The virtual sales market (E-commerce) has expanded a lot nowadays due to the ease and practicality provided by this purchase way, and due to the fact that the technologies are becoming more and more accessible. With the increase in consumers who use this method of purchase, the implementation of virtual assistants by companies can add benefits to both sides of the negotiation consumers-companies, since the use of virtual assistants can allow the automation of tasks involving means of communication, thus accelerating problem resolution and increasing customer service productivity, in addition to being able to provide personalized experiences tailored to each consumer. In this work, concepts that are relevant to the topic of Knowledge Base Question Answering (KB-QA) are discussed, and ways to develop an end-to-end KB-QA model that is capable of answering user questions provided in natural language, based on information contained in knowledge bases (KBs) that can be seen as graphs. To this end, two parallel lines of research are addressed, complete models, which are able to directly deal with the question in natural language and provide an answer based on the KB, and composite models, models that are composed of a method for converting the question from natural language in a query and a method of querying the KB via queries, where the query is usually made up of triples of the form (subject, predicate, object). In order to develop an end-to-end KB-QA model via Ensemble by majority vote, superior on terms of metrics like F1-score, this work addresses the generation of a composite KB-QA model, where methods for transforming questions from natural language to queries and methods to query the Knowledge Bases based on these queries are analyzed and selected, the search and analysis of complete KB-QA models is carried out, and then some of the models are selected for the composition of the Ensemble. In the analysis of the composite KB-QA model, databases from different areas such as COUNTRIES and FB15k-237 are applied for independent analysis of the method for querying the KB, in order to determine the potential of the model for different structures of the Knowledge Bases based on triples, and then the WikiMovies database is used to analyze the whole composite model. For the analysis of the complete KB-QA models, the WebQuestionsSP question bank is used, which is developed to be answered through queries to the KB Freebase. As the main result of this work, three different Ensembles are generated, called Simple Ensemble, Ensemble with Countermeasure and Ensemble with total countermeasure, which are state-of-the-art for the KB-QA task considering the WebQuestionsSP database with F1-scores of 75.40%, 78.72% and 81.43% respectively. Besides these results, as an addendum, some analyzes and results are presented involving the development of a composite KB-QA model, here called Parot-MINERVA composite model.

Keywords: Knowledge Base Question Answering. Virtual Assistants. Natural Language Processing. Artificial Intelligence. Chatbots. E-commerce.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o mercado brasileiro de vendas virtuais tem se expandido cada vez mais e comprar via *internet* tem se tornado um hábito dos consumidores devido a facilidade proporcionada pela tecnologia. De acordo com a ABCOMM (2020), no ano de 2018 o comércio eletrônico teve um crescimento de 12% e no primeiro trimestre de 2019 já apresentava um crescimento de 12%. No ano de 2020, acelerado devido as restrições impostas para o varejo físico em função da pandemia de Covid-19 e a necessidade de distanciamento social, apresentou um crescimento de 68% em relação ao ano anterior e no ano de 2021 apesar da retomada gradativa do varejo físico, ocorreu ainda um crescimento de 19% em relação a 2020, com uma projeção de crescimento de 12% para o ano de 2022 (ABCOMM, 2021; E-COMMERCE_BRASIL, 2022).

Com isso, a implantação de assistentes virtuais (ou de *chatbots*) se torna uma opção viável para a redução dos custos empresariais, uma vez que eles permitem a automação de tarefas, resolução de problemas e o aumento da produtividade no atendimento ao cliente, além de permitir a entrega de experiências personalizadas para cada cliente graças às informações pessoais e preferências de consumo fornecidas pelos próprios consumidores às empresas (BLIP, 2019).

Alguns exemplos de assistentes virtuais são a Cortana da Microsoft, a Siri da Apple, Alexa da Amazon, a assistente Google, Samsumg S voice, Nuance Dragon, e o M do facebook (KEPUSKA; BOHOUTA, 2018). Estes assistentes, por serem produtos comerciais de grandes empresas, não possuem suas informações estruturais completamente disponibilizadas, além de serem mais voltados para nível pessoal e não se adéquam às bases de dados empresariais.

Em boa parte dos trabalhos científicos, os termos *chatbot* e assistentes virtuais são utilizados sem distinção. Para os que diferenciam os termos, ainda que não exista uma definição específica para delimitá-los, utilizam diferenças como o fornecimento de respostas mais humanizadas, a capacidade de manter o fluxo de diálogo e a capacidade de auxiliar em tarefas diárias tais como configuração de alarmes, realização de ligações e etc., sendo os assistentes virtuais superiores aos *chatbots* em ambos os aspectos, ou seja, os assistentes virtuais são considerados mais "inteligentes".

No princípio, o primeiro conceito de um modelo de *chatbot* foi desenvolvido no trabalho de Weizenbaum (1966), sendo o termo *chatbot* ainda inexistente, vindo a ser concebido algumas décadas após. O modelo de Weizenbaum (1966) foi denominado programa *ELIZA*, criado com o objetivo de tornar possível alguns tipos de conversações entre o homem e a máquina. O

programa *ELIZA* é baseado no reconhecimento de palavras-chave na sentença fornecida pelo usuário, que devem estar contidas em um dicionário, então para cada palavra-chave associa-se regras para decomposição da sentença e regras para formulação de uma resposta. Desta forma, o programa *ELIZA* não empregava técnicas mais sofisticadas de *machine learning*, e seu modelo era completamente pré-programado.

O primeiro modelo de *chatbot* a utilizar uma técnica de inteligência artificial foi o modelo *JABBERWACKY*, criado em 1988 por Rollo Carpenter, que foi criado para tentar estimular conversas humanas naturais como uma forma de entretenimento. Como fonte de conhecimento, ao invés de utilizar regras manualmente desenvolvidas como os modelos até então desenvolvidos e fornecer respostas baseadas em padrões reconhecidos na entrada fornecida pelo usuário, o *JABBERWACKY* aprendia a conversar conversando, de forma que interações nunca vistas pelo modelo eram guardadas e então podiam ser empregadas posteriormente em interações com outros usuários. Observe que desta forma, a base de dados do *JABBERWACKY* não era uma fonte estática, outro ponto que diferenciava o modelo dos modelos desenvolvidos até aquele ano (PEREIRA et al., 2016).

Um modelo que ganhou bastante destaque dentre os primeiros *chatbots* criados, principalmente por ter sido o primeiro modelo de chatbot *open source* da história, foi o modelo *ALICE (Artificial Linguistic Internet Computer Entity)*, desenvolvido no ano de 1995 por Richard Wallace. O modelo *ALICE* foi visto como uma extensão do programa *ELIZA* por também empregar uma correspondência de padrões heurística para conversar com humanos. No entanto, *ALICE* era capaz de ter conversas mais humanizadas devido ao fato de utilizar processamento de linguagem natural por meio do uso de *Artificial Intelligence Markup Language (AIML)* o que lhe permitia fornecer respostas mais sofisticadas (WALLACE, 2009).

Recentemente, surgiu-se um modelo que tem chamado muita atenção tanto em mídias técnicas quanto na grande mídia, o modelo de linguagem *GPT-3 (Generative Pre-trained Transformer 3)*, criado pelo OpenAI, um laboratório de pesquisa de inteligência artificial, no ano de 2020. O grande destaque do *GPT-3* se dá principalmente por ser o maior modelo de linguagem já desenvolvido na história, com 175 bilhões de parâmetros e 96 camadas, treinado em um corpus de conteúdos da internet com 499 bilhões de *tokens*, o que lhe garante uma capacidade de gerar textos que dificilmente consegue-se distinguir se são textos de autoria humana ou não, além disponibilidade do modelo para uso através de uma *API (Application Programming Interface)* pública (DALE, 2021).

Apesar de toda sua capacidade, o *GPT-3* ainda pode gerar saídas com ausência de coerência semântica, principalmente em sentenças mais longas, gerar textos enviesados com conteúdos que vão contra a moral contidos em seu conjunto de treino, além de que as saídas geradas podem conter afirmações que não condizem com a verdade, o que evidencia que apesar de todo o avanço obtido no campo do Processamento de Linguagem Natural, ainda há muito a ser desenvolvido (DALE, 2021).

Atualmente, o desenvolvimento de um assistente virtual é composto de várias etapas de desenvolvimento, podendo empregar nestas etapas várias técnicas de *machine learning*. Tais etapas consistem desde a compreensão da mensagem formulada em linguagem natural emitida pelo usuário, que deve ser convertida em algum tipo de informação que possa ser interpretada pelo assistente virtual ou *chatbot* instalado na máquina, até a busca e obtenção das respostas a serem fornecidas em linguagem natural ao usuário.

Com base na informação obtida da questão, o assistente virtual deve decidir qual atitude deverá ser tomada em relação à mensagem, podendo ser dentre outras, buscar uma resposta, seja descobrir a informação solicitada pelo usuário ou solicitar mais informações sobre o assunto para clarificação ou desambiguação de resultados, expressar uma opinião, ou mesmo ativar algum tipo de tarefa, por exemplo agendar despertador, fazer uma ligação e etc.. Após determinada a ação a ser tomada, uma resposta em linguagem natural deve ser gerada para a requisição do usuário a fim de fornecer um *feedback*. Adicionalmente, são necessários o desenvolvimento de mecanismos que permitam manter o fluxo de conversação, gerenciando o histórico do diálogo a fim de manter a coerência, e se ater ao contexto para interpretação das informações.

Após a interpretação da mensagem do usuário, caso seja uma solicitação de informação, o assistente virtual precisa determinar a resposta à questão que deverá ser fornecida ao usuário. Como fonte de conhecimento, os assistentes virtuais podem buscar as repostas em bancos de dados de textos não estruturados ou bancos de dados estruturados (em geral denominados *Knowledge Bases*), cuja informação pode ser vista como uma estrutura de grafo, também conhecida como *Knowledge Graph*. A tarefa de responder a questões com base em informações contidas em *Knowledge Graphs* é denominada *Knowledge Base Question Answering (KB-QA)*.

O uso de ambos os tipos de bancos de dados (estruturados ou não estruturados) para a consulta de informação possuem vantagens e desvantagens um em relação ao outro. Uma das vantagens em utilizar bancos de dados estruturados é que dados armazenados em tabelas

também podem ser convertidos para o formato de *Knowledge Base*. Desta forma, pensando-se em dados obtidos por uma empresa, onde boa parte destes dados podem estar armazenados de forma estruturada, empregar modelos capazes de lidar com este formato de informação se mostra mais razoável para a obtenção de respostas. Além disto, as *Knowledge Bases* permitem a inferência de informações em grafos, e reduzem a demanda de armazenamento pelos bancos de dados. Considerando-se tais benefícios das consultas em *Knowledge Bases*, enfatiza-se esta tarefa ao longo deste trabalho.

Para desenvolvimento de modelos voltados para a consulta de informações em *Knowledge Bases (KB-QA)*, diversas abordagens são listadas na literatura, principalmente relacionadas a forma a qual a necessidade de informação contida na questão é representada pelo sistema para consulta à *KB* e como é realizada a busca pelo sistema. As principais abordagens são a abordagem de análise semântica, a abordagem de recuperação de informação e a abordagem de análise semântica neural.

A abordagem de análise semântica aplica regras ou *templates* desenvolvidos com base em informações semânticas e/ou sintáticas das questões para formação de uma representação da forma lógica da questão (em geral um grafo de representação da questão). Um exemplo de modelo nesta linha é o *GAnswer* (HU et al., 2017) que apresenta duas estratégias para a formação da forma lógica, *node first* que prioriza a detecção das entidades da questão para inferir a forma lógica, e a estratégia *relation first* que prioriza a detecção das relações para a representação da questão em forma lógica.

Outro modelo que emprega a abordagem de análise semântica é o *Parot* (OCHIENG, 2020). O *Parot* desenvolve várias regras baseadas nas árvores de dependências das questões para obtenção de uma representação em forma de *SPARQL*, que pode ser usada para consultar à *KB*. Um grande problema com a abordagem de análise semântica é que gerar regras ou *templates* é trabalhoso, e representar alguns tipos de questões pode ser inviável.

A estratégia de recuperação de informação, consiste de representar as questões e informações da *Knowledge Base* através de redes neurais, bem como também são criadas as representações das informações relacionadas a *Knowledge Base*. Para isto, são empregadas complexas estruturas de redes. Um modelo interessante que aplica este tipo de estratégia é o *BAMnet* (CHEN; WU; ZAKI, 2019), que representa a questão através de *embeddings* e então utiliza redes como *LSTM* e mecanismos de atenção para refinamento das representações em conjunto com representações da *KB* via *Key-Value Memories*.

Também da linha de pesquisa de recuperação de informação, o modelo *NSM* (HE et al., 2021) aplica máquinas de estado neural para fazer a representação da questão e da *Kowledge Base* através da técnica *teacher-student*, onde a rede *teacher* é treinada para conseguir fornecer informações intermediárias para direcionar a rede *student* na busca da resposta para a questão, uma vez que os modelos geralmente só possuem informação da resposta final para o treino, sem informações sobre passos intermediários nos grafos. Modelos de recuperação de informação sofrem com erros devido a modelagem de restrições, devido a suas formas de seleção de respostas, que em geral se baseiam em limiares por se tratar de conjuntos de respostas, dentre outros.

Já a estratégia de análise semântica neural visa combinar os modelos anteriores por meio de métodos neurais para gerar representações em forma lógica das questões. Atualmente tais modelos tem apresentado os melhores resultados para a tarefa de *KB-QA* e estão dentre o modelos estado-da-arte para os principais bancos de dados da literatura. Dois modelos interessantes que empregam esta estratégia são o *NS-CQA* (HUA et al., 2020) e o *Multihop* (LAN; JIANG, 2020), ambos empregam técnicas de aprendizado por reforço para gerar as representações das questões, se diferindo principalmente na forma como suas ações de geração do grafo são desenvolvidas.

Apesar das diferentes abordagens e sua evolução, a tarefa *KB-QA* ainda demonstra uma brecha para grande aprimoramento.Atualmente na literatura, tem-se focado em criar modelos que possam lidar com todas as questões simultaneamente independentemente da estratégia empregada. É importante notar, no entanto, que existe um grande desafio em lidar com questões em linguagem natural, que podem conter erros, faltar informação de suporte ou possuir altas complexidades, além de que existem muitos processos diferentes para interpretação destas informações pela máquina, o que torna um problema de grande espectro de possibilidades. Além disto, há dificuldades relacionadas à inferência da resposta na *KB* que pode faltar informações ou apresentar padrões irregulares de representação das informações.

Para tentar contornar tais dificuldades e gerar um modelo *KB-QA* que possa resultar em um assistente virtual mais robusto para fornecer respostas, desenvolve-se esta pesquisa, de acordo com os objetivos dispostos na Seção 1.1.

1.1 Objetivos

O objetivo deste trabalho é desenvolver um modelo voltado para a tarefa de *Knowledge Base Question Answering (KB-QA)*, por meio de modelos de diferentes abordagens, capazes de responder perguntas realizadas em linguagem natural, a fim de gerar um modelo com melhores desempenhos em aspectos de *f1-score*, precisão, *recall*, micro f1-score, acurácia real, *hits at 1*, taxa de verdadeiros positivos e taxa de verdadeiros positivos desconsiderando questões sem respostas. Este objetivo está sujeito a:

a) Análise e implementação de modelos que possam fazer consultas à *Knowledge Bases* por meio de *queries*;

b) Implementação de métodos para a conversão de questões em linguagem natural para *queries*;

c) Análise e seleção dos modelos fim-a-fim, com a finalidade de desenvolvimento de um modelo *KB-QA* via *ensemble*.

1.2 Estrutura do Trabalho

O texto está estruturado da seguinte forma: no Capítulo 2 são mostradas ferramentas e conceitos que são e que poderão ser usados no desenvolvimento do modelo para consultas a bancos de dados estruturados (*Knowledge Bases*) a partir de questões em linguagem natural (tarefa *KB-QA*), no Capítulo 3 são mostradas as estratégias abordadas para o desenvolvimento do modelo para *KB-QA* via *Ensemble*, através de modelos completos ou compostos, bem como são apresentados os bancos de dados e métricas empregados nas avaliações, no Capítulo 4 são apresentados os resultados e suas análises, e por fim no Capítulo 5 são apresentadas algumas conclusões do projeto e direcionamentos para projetos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos mais relevantes para o trabalho desenvolvido, tais como a composição da estrutura de um *chatbot* ou assistente virtual, processamento de linguagem natural, tipos de redes, além da apresentação de abordagens e modelos fim-a-fim passíveis de serem empregados no trabalho.

2.1 Assistentes Virtuais

Chatbots e assistentes virtuais são considerados interfaces de conversação e, por não haver uma definição específica para delimitá-los, alguns autores usam ambos termos indistintamente. Joshi (2018) os consideram diferentes e fornece as seguintes definições para assistentes virtuais e *chatbots*:

a) Assistente Virtual: é um agente pessoal digital baseado em *software* que auxilia na realização de atividades diárias como configurar alarmes, programar reuniões, fazer ligações, digitar mensagens e assim por diante. É semelhante a um assistente pessoal humano que faz anotações durante reuniões, ajudando a controlar e gerenciar dispositivos inteligentes, obter direções, gerenciar novas notícias e etc;

b) *Chatbot*: é um programa automático usado como meio de interação com humanos via texto ou meios audíveis, geralmente usado pelas organizações para enriquecer programas de serviços ao cliente, onde os clientes podem interagir com *chatbots* para sanar dúvidas sobre produtos, obter informação relacionada a produtos, ou mesmo marcar reuniões com gerentes de produtos.

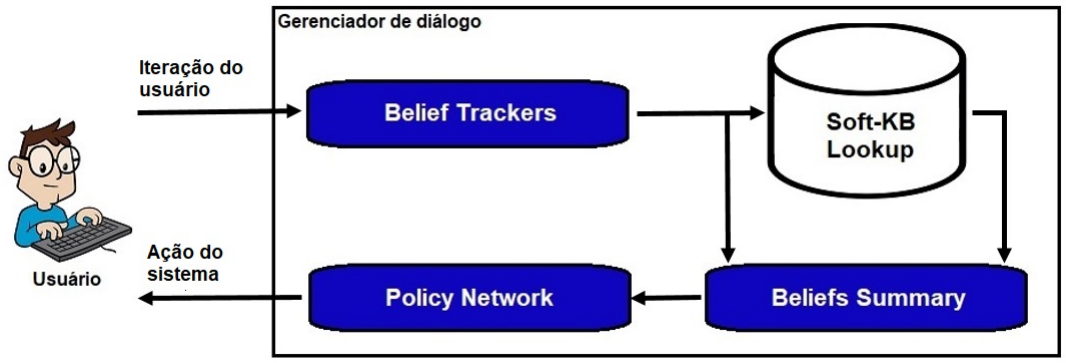
Uma das principais diferenças é que *chatbots* não possuem entendimento das emoções humanas. Assim, apesar de fornecerem respostas acuradas a questões, não são capazes de responder de acordo com o humor do usuário. Assistentes virtuais com o uso de processamento de linguagem natural interagem com o usuário de maneira mais "humanizada", oferecendo maior eficiência e acurácia.

Além disso, outra importante diferença entre um assistente virtual e um *chatbot* é o fluxo de diálogo. Para *chatbots* em geral, quando uma conversa é interrompida no meio o contexto das interações será perdido. Já o assistente virtual emprega técnicas de fluxo de diálogo dinâmicas

para entender as intenções humanas e enriquecer a comunicação. Alguns exemplos de assistentes virtuais são a Cortana da Microsoft, a Siri da Apple, Alexa da Amazon, a assistente Google, Samsumg S *voice*, *Nuance Dragon*, e o M do facebook (KEPUSKA; BOHOUTA, 2018).

O desenvolvimento de um assistente virtual, de maneira simplificada, pode ser dividido em quatro partes fundamentais (Figura 2.1), sendo elas, o *belief tracker*, a *soft-KB lookup*, o *beliefs summary* e a *policy network* (GAO; GALLEY; LI, 2018; DHINGRA et al., 2016):

Figura 2.1 – Estrutura de um assistente virtual



Fonte: Do Autor (2022)

a) *belief tracker*: módulo para resolver correferências e correlações nas iterações do usuário utilizando contexto da conversa, para identificar intenções do usuário, extrair atributos associados e monitorar o estado do diálogo;

b) *soft-KB lookup*: uma interface com a *Knowledge base* ou Banco de dados textual para consultar resultados relevantes (modelos *KB-QA* ou *text-QA*), formando as *queries* de acordo com o histórico de diálogo capturado pelo *belief tracker* e a atual questão do usuário;

c) *beliefs summary*: módulo para extração de estatísticas resumidas do estado do diálogo e representação do estado em um vetor, com base nas saídas do *belief tracker* e da *soft-KB lookup*;

d) *policy network*: módulo para selecionar a próxima ação baseada no estado do diálogo, podendo ser programado ou treinado.

O desenvolvimento de um assistente virtual completo é uma tarefa muito ampla, que pode envolver lidar com diferentes formas de comunicação na entrada tais como voz, imagens,

texto e etc. Além disso, para desenvolvimento de um módulo de conversação são necessários submódulos para identificar se a interação com o usuário é uma solicitação de informação ou opinião, para compreensão de aspectos de empatia e sentimentos, da personalidade do assistente virtual dentre outros e para cada destes submódulos, são necessárias interfaces para diferentes tipos de informações (bancos de dados) tais como bancos conversacionais e não conversacionais, *Knowledge Bases*, perfis do assistente virtual e a dados dos usuários ativos. Por ser uma área de pesquisa muito ampla e com muitas dificuldades envolvidas, nesta pesquisa foca-se na sub-tarefa de *Knowledge Base Question Answering (KB-QA)*. Para maiores detalhes sobre o desenvolvimento de um assistente virtual completo vide o trabalho de Zhou et al. (2020) para o desenvolvimento do assistente virtual denominado *Xiaoice*.

Assistentes virtuais podem ser empregados no comércio para alavancar as vendas. De acordo com Nacaxe (2019), a transformação digital do comércio brasileiro tem avançado significativamente, gerando muitas vantagens, como o aumento da eficiência operacional nas empresas, de onde cerca de 40% dos ganhos financeiros das empresas é proveniente. Assim, o emprego de assistentes virtuais pode se tornar ainda mais impactante, principalmente associado com os atuais avanços da área de processamento de linguagem natural.

2.2 Processamento de Linguagem Natural

O processamento de linguagem natural (*Natural Language Processing (NLP)*) é uma gama de técnicas computacionais desenvolvidas para a análise e representação da linguagem natural e que tem evoluído muito até os dias de hoje, desde a época dos cartões perfurados, onde a análise de uma simples sentença gastava em torno de 7 minutos, até a atual era do *Google* e semelhantes, onde milhões de páginas podem ser processadas em menos de um segundo (CAMBRIA; WHITE, 2014).

O *NLP* permite a realização de várias tarefas relacionadas à linguagem natural, desde as mais simples como *part-of-speech tagging (POS)* cujo objetivo é rotular cada palavra com uma única *tag* que indica seu papel sintático, por exemplo “substantivo”, “verbo”, “advérbio” e etc., a tarefa *named entity recognition (NER)* que visa reconhecer as categorias de elementos atômicos na sentença como “PESSOA” ou “LOCALIZAÇÃO”, e a tarefa *semantic role labeling (SLR)* que define um papel semântico para um constituinte sintático de uma sentença, como “predicado”, “objeto”, “sujeito” e etc., até as mais avançadas como tradução de máquina, re-

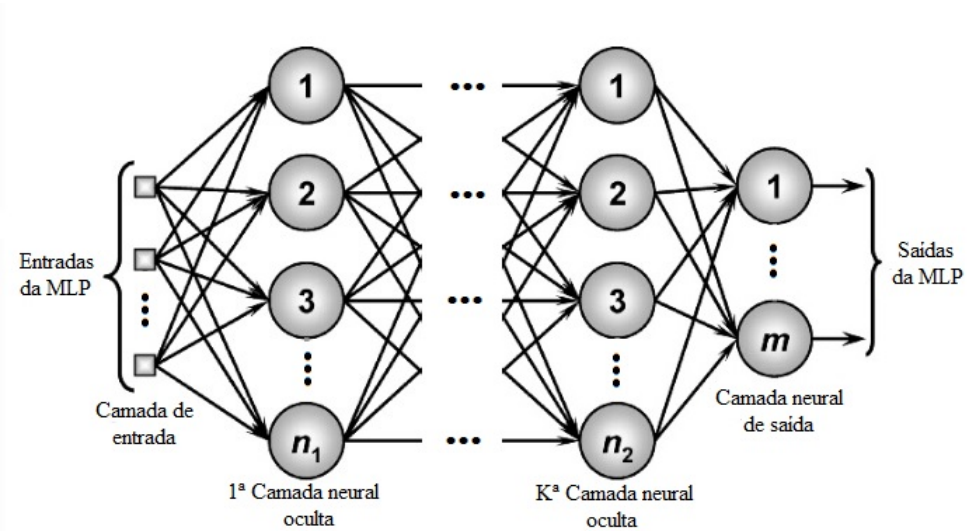
sumo de textos, análise de sentimentos e sistemas de diálogo, também denominados *Chatbots* ou Assistentes Virtuais (YOUNG et al., 2018; COLLOBERT et al., 2011).

Dentro dos modelos de assistentes virtuais, as tarefas de *NLP* são empregadas do início ao fim do processo do diálogo, na obtenção do contexto da conversa, identificação de intenções do usuário, extração de atributos associados, na busca dos resultados relevantes para fornecer ao usuário, e na formulação das respostas. Para cumprir tais tarefas, a máquina precisa representar informações relacionadas à linguagem natural na forma numérica, empregando para isso em geral as redes neurais.

2.3 Redes Neurais

De forma geral, uma rede neural (por exemplo a *Multi-Layer Perceptron (MLP)*) é um sistema que visa modelar ou se assemelhar a maneira em que o cérebro executa tarefas, baseado na sua estrutura. Assim, podem ser descritas de forma superficial, como uma cadeia de várias unidades denominadas neurônios, que são interligados entre si em diferentes camadas, e se ativam em diferentes níveis de acordo com o resultado do processamento dos dados recebidos (Figura 2.2).

Figura 2.2 – Exemplo de rede neural (Um perceptron multi-camadas (*MLP*))



Fonte: Adaptado de Moreira (2018)

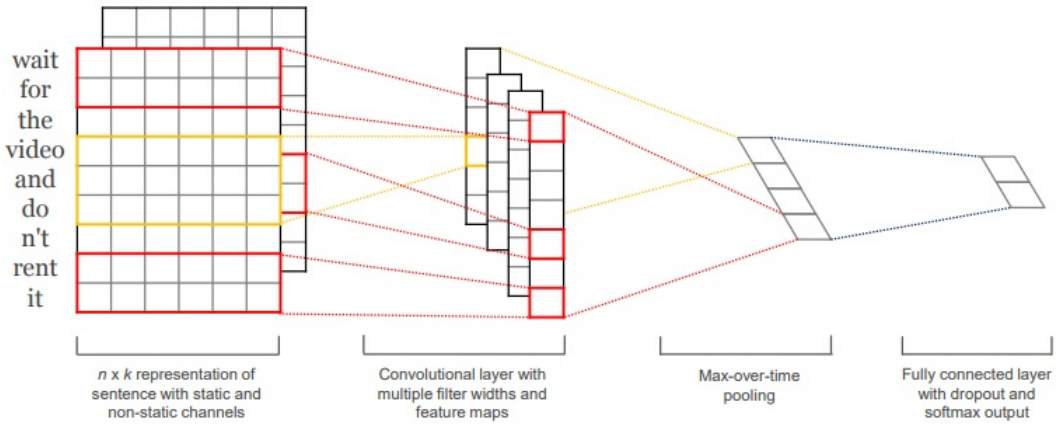
Devido à estrutura completamente conectada entre os neurônios de uma *MLP*, ela tende a gerar grande *overfitting* do modelo aos dados de treino e, além disso, o seu número de parâmetros cresce proporcionalmente ao número de dados de entrada, gerando um alto custo com-

putacional. Desta forma, foi desenvolvida outra classe de rede neural denominada redes neurais convolucionais.

A princípio, as redes neurais convolucionais foram desenvolvidas para análise e classificação de imagens (LECUN et al., 1990). Para contornar os problemas relacionados às *MLPs*, as redes convolucionais tiram proveito do padrão hierárquico dos dados e montam padrões mais complexos usando padrões menores e mais simples, reduzindo a complexidade e conectividade da rede (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Apesar de inicialmente serem voltadas para a análise de imagens, por sua capacidade de extração de características de alto nível, as redes neurais convolucionais foram adaptadas para desenvolvimento de modelos para tarefas textuais e um exemplo pode ser visto na Figura 2.3, onde uma frase é fornecida na entrada da rede, então são aplicados diferentes filtros para combinação de características entre as palavras próximas e aplica-se *pooling* para seleção das características mais relevantes. Com isso são empregadas em várias tarefas de processamento de linguagem natural, como classificação de tipos de questões, análise de sentimentos, resumo de textos, *question answering* e etc.

Figura 2.3 – Exemplo de estrutura CNN para tarefas textuais



Fonte: Kim (2014)

As redes neurais convolucionais se mostram uma forma efetiva para capturar características de conjuntos de palavras, o que pode ser suficiente em alguns tipos de tarefas de classificação de sentenças, no entanto estas redes possuem sensibilidade a ordem das palavras restrita localmente, ou seja, a ordem das palavras só tem efeito de acordo com o tamanho dos filtros, e dependências de longo termo são ignoradas, uma vez que algumas palavras nunca serão capturadas juntas dentro de um mesmo filtro (YOUNG et al., 2018). Para lidar com estes problemas pertinentes as redes neurais convolucionais, é empregada outra classe de redes neu-

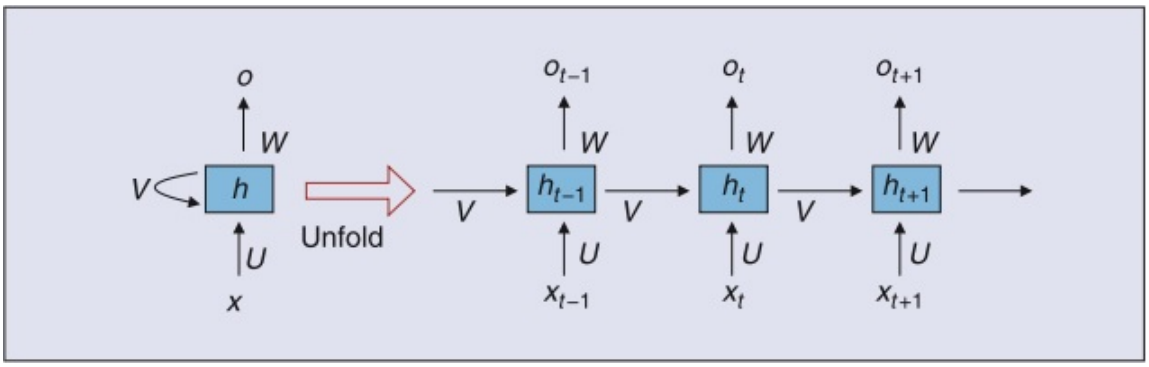
rais denominada redes neurais recorrentes, que são atualmente as mais empregadas para tarefas de processamento de linguagem natural e serão melhor descritas na seção 2.4.

2.4 Redes Neurais Recorrentes

As redes neurais recorrentes (*Recurrent Neural Networks (RNN)*) surgiram com o intuito de ir além das redes neurais comuns, levando em conta o tempo. A abordagem empregada para isto é a de representar o tempo implicitamente por seus efeitos ao invés de expressá-lo explicitamente como é feito em representações espaciais (ELMAN, 1990). Na prática, isto envolve o uso de *links* recorrentes para produzir redes com memórias dinâmicas, onde padrões das unidades ocultas são realimentados no modelo, e as representações internas que são desenvolvidas refletem as questões das tarefas no contexto dos estados internos anteriores.

A estrutura de uma *RNN* simples é mostrada na Figura 2.4 onde o bloco à esquerda da seta vermelha é o modelo geral e os blocos à direita da seta mostram o comportamento da rede em alguns passos de tempo. As setas pretas são os *links* e representam o sentido do fluxo de informações, os *X's* são as entradas fornecidas à rede em cada passo de tempo, os *O's* são as saídas e os blocos azuis são as camadas internas da rede, que contém os *h's* denominados estados internos.

Figura 2.4 – Estrutura de uma rede neural recorrente simples



Fonte: Young et al. (2018)

A rede realiza os mesmos cálculos sobre cada *token* da sequência e cada passo é dependente dos resultados e cálculos anteriores, de onde vem o termo "recorrente". A seguir é feita a formulação do modelo de acordo com Mikolov et al. (2011) e Young et al. (2018).

As entradas $x(t)$ são formadas pela concatenação do vetor $w(t)$ que representa a palavra atual em geral utilizando-se codificação *one-hot encodings* ou *embeddings*), e o vetor $s(t-1)$

que representa a saída da camada oculta no passo de tempo anterior, assim:

$$x(t) = [w(t)^T s(t-1)^T]^T \tag{2.1}$$

onde *s(t)* é descrito pela equação:

$$s(t) = f(Ux(t) + Vs(t-1)) \tag{2.2}$$

e por fim a saída no tempo t é determinada como:

$$o(t) = g(Ws(t)) \tag{2.3}$$

sendo U, V e W os vetores de pesos a serem ajustados em treino, $f(z)$ uma transformação não linear tal como, *tanh, sigmoid* ou *ReLU*, e $g(z)$ uma função de ativação *softmax*.

Devido à sua natureza sequencial e *short term memory*, a *RNN* é empregável a várias tarefas envolvendo o processamento de linguagem natural, tais como modelagem de linguagem (MIKOLOV et al., 2011) que fornece contexto para distinguir palavras e frases, tradução de máquina (LIU et al., 2014) que se trata da tradução de textos de um idioma para outro, reconhecimento de fala (GRAVES; JAITLY, 2014) que permite que máquinas sejam capazes de converter linguagem falada para linguagem textual, a tarefa de *image captioning* (KARPATHY; FEI-FEI, 2017) que gera legendas para imagens dentre outras.

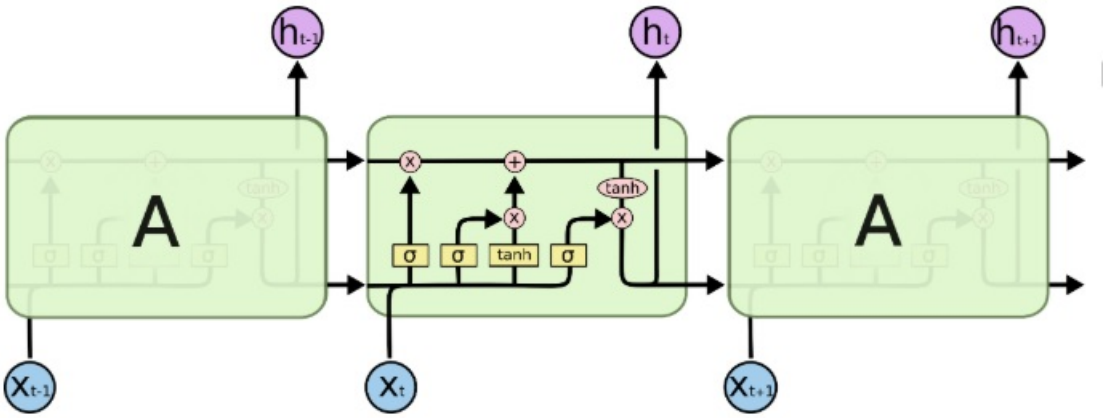
Redes Neurais Recorrentes sofrem com o problema de explosão e desaparecimento do gradiente (*exploding or vanishing gradient problem - EVGP*). Isso ocorre quando a derivada da função de perda (*loss function*) em relação aos parâmetros da rede se torna muito grande ou muito pequena, desta forma durante a atualização dos pesos, estes se tornam grandes demais causando a explosão do gradiente, onde o aprendizado se torna muito impreciso, ou pequenos demais causando o desaparecimento do gradiente, onde o aprendizado se torna insignificante (HANIN, 2018). No intuito de contornar tal problema, foram desenvolvidas às *Long Short Term Memories (LSTM)*.

2.4.1 *Long Short Term Memory*

Similar às redes neurais recorrentes simples, as *Long Short Term Memories (LSTM)* são um tipo especial de Rede Neural Recorrente desenvolvido por Hochreiter e Schmidhuber

(1997) que também possui a mesma estrutura encadeada ao longo do tempo, porém o bloco de memória é composto por quatro camadas neurais que interagem entre si de uma maneira especial, de forma a evitar o problema da dependência a longo prazo. Seu grande diferencial é a existência da célula de estado (*cell state*) que permite a informação fluir pelo bloco sem sofrer grandes alterações. O esquemático do funcionamento de uma *LSTM* ao longo do tempo pode ser visto na Figura 2.5

Figura 2.5 – Estrutura de uma LSTM

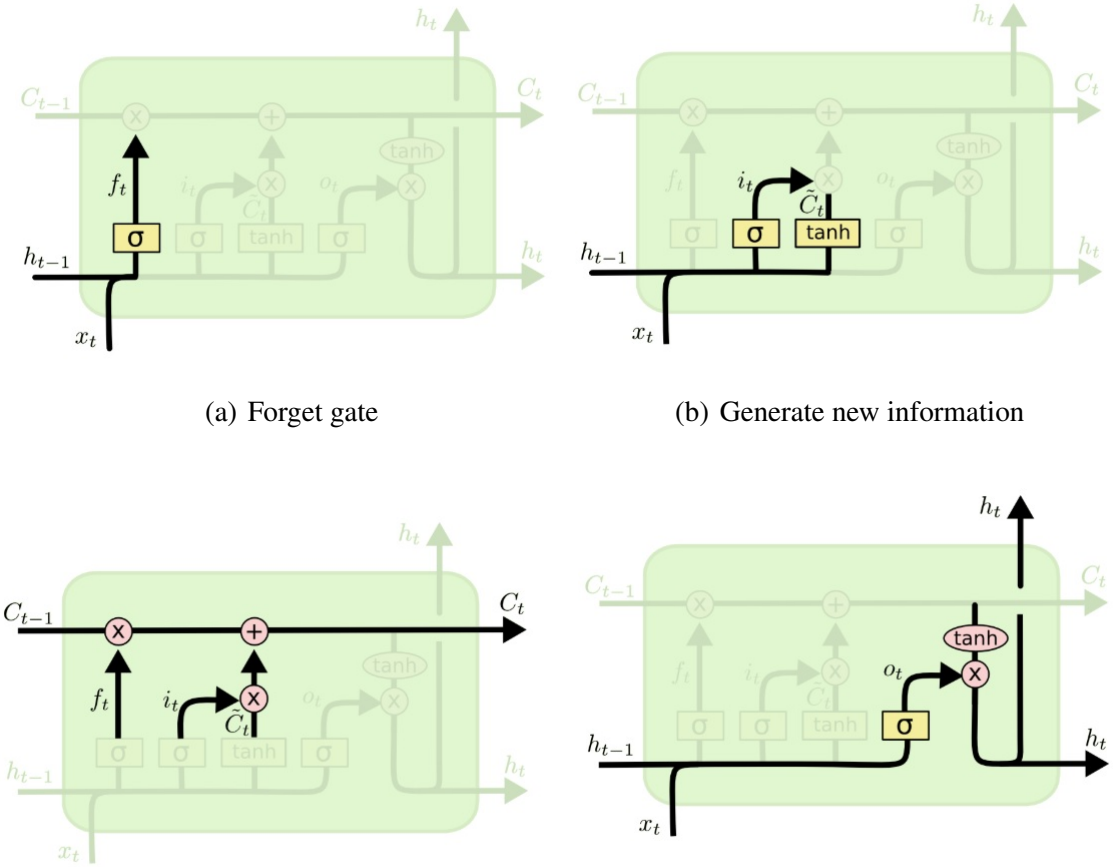


Fonte: Olah (2015)

Na Figura 2.5 os círculos vermelhos representam operações pontuais, os blocos amarelos são camadas de rede neural, as setas representam o fluxo de informação, onde a seta direta na parte superior que atravessa o bloco sem camadas de rede neural é denominada *cell state*. A Figura 2.6 mostra o bloco de memória dividido em quatro fases (*forget gate Figura 2.6(a), generate new information Figura 2.6(b), cell state update Figura 2.6(c), output gate Figura 2.6(d)* que serão descritas a seguir, baseadas na formulação disponível em Young et al. (2018):

a) *Forget gate*: Nesta fase do processamento da informação no bloco *LSTM* (Figura 2.6(a)), a informação proveniente da entrada no passo de tempo atual e a saída gerada no passo de tempo anterior são concatenadas e então são fornecidas a uma camada de rede neural *sigmoid*. O intuito de tal operação é selecionar o que será descartado da célula de estado, sendo gerado para cada valor na célula de estado um valor entre 0 e 1, onde 0 representa se livrar completamente de tal informação e 1 significa manter completamente.

Figura 2.6 – LSTM em quatro fases



(a) Forget gate (b) Generate new information

(c) Cell state update (d) Output gate

Fonte: Adaptado de Olah (2015)

A descrição da formulação da operação é mostrada nas equações:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}, \tag{2.4}$$

$$f_t = \sigma(W_f \dot{X} + b_f). \tag{2.5}$$

b) *Generate new information*: Nesta fase decide-se qual informação será armazenada na célula de estado em duas etapas. Primeiro, as entradas (Equação (2.4)) são alimentadas em uma camada de rede *sigmoid* denominada *input gate layer* (Equação (2.6)) que decidirá quais valores serão atualizados. Segundo, as mesmas entradas são alimentadas em uma camada de rede neural tangente hiperbólica (Equação (2.7)), que cria um vetor de

novos valores candidatos que poderia ser adicionado ao estado:

$$i_t = \sigma(W_i.X + b_i), \tag{2.6}$$

$$\tilde{C}_t = \sigma(W_C.X + b_C). \tag{2.7}$$

c) *Cell state update*: Os resultados gerados pelas camadas de rede neural nos itens anteriores (Equações (2.5), (2.6) e (2.7)) são utilizados para atualizar a célula de estado anterior C_{t-1} para a nova célula C_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{2.8}$$

d) *Output*: A entrada (Equação (2.4) é alimentada em uma camada *sigmoid* denominada *output gate* para decisão de quais partes da célula de estado serão dispostas na saída. Aplica-se então a tangente hiperbólica a célula de estado e então multiplica-se pela saída da camada *sigmoid* e assim, na saída é disponibilizada uma versão filtrada da célula de estado.

$$o_t = \sigma(W_o.X + b_o), \tag{2.9}$$

$$h_t = o_t * tanh(C_t). \tag{2.10}$$

Alguns trabalhos aplicam variantes da *LSTM* descrita. Gers, Schmidhuber e Cummins (2000) adicionaram *peephole connections* permitindo que as *gate layers* acessem a célula de estado. Já Cho et al. (2014) desenvolveram as *Gated Recurrent Units (GRU)* que combina as camadas *forget gate* e *input gate*, e com algumas outras mudanças, torna a saída e a célula de estado uma só, tornando o modelo mais simples.

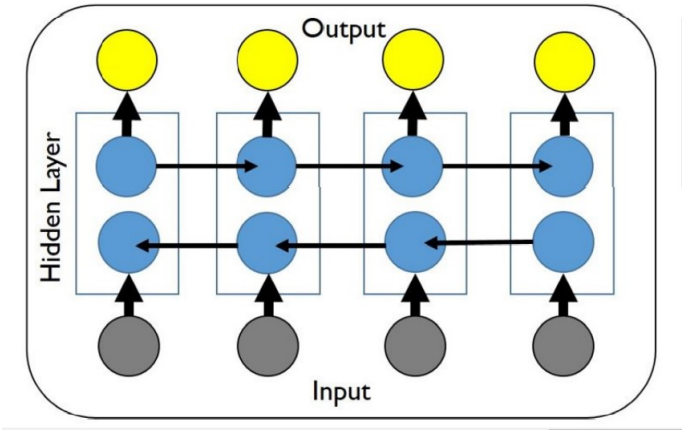
2.4.2 *Bilateral Long Short Term Memory*

Um problema relacionado às redes neurais recorrentes incluindo as *LSTMs*, devido à sua maneira de processar as informações sequencialmente, é que as suas saídas geradas tendem a ser majoritariamente baseadas em seu contexto prévio, e apesar de haver formas de introduzir contexto futuro adicionando-se atraso entre as saídas e os alvos por exemplo, elas não permitem utilizar completamente as dependências anteriores (GRAVES; SCHMIDHUBER, 2005).

Para contornar tal problema, uma solução é a aplicação das *Bilateral Long Short-Term Memories (BiLSTMs)* (Figura 2.7) que possuem um fluxo de informações de duas vias. Para este

fim, a rede *BiLSTM* é composta por duas redes *LSTM*, uma processa a sequência da esquerda para a direita e a outra processa a sequência da direita para a esquerda; ambas são conectadas à mesma camada de saída (MOHAN; GAITONDE, 2018).

Figura 2.7 – Estrutura de uma rede BiLSTM



Fonte: Mohan e Gaitonde (2018)

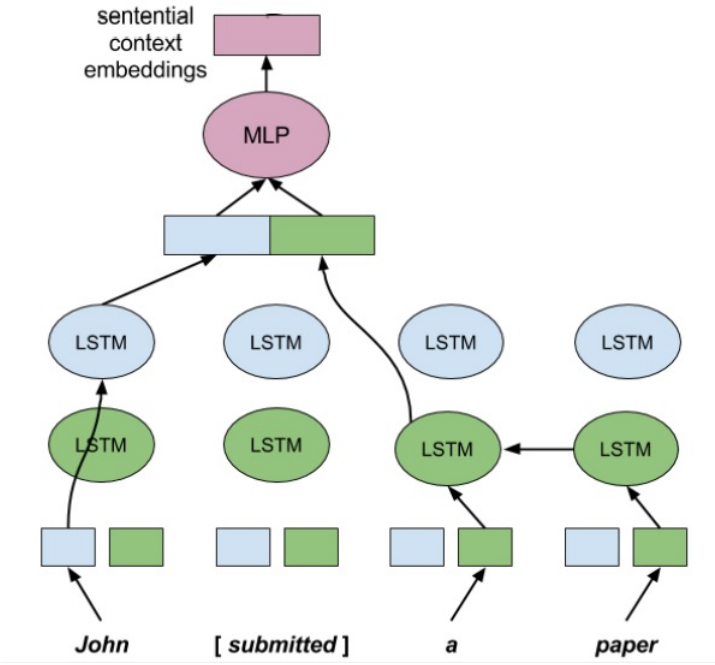
Em alguns casos, uma terceira rede é usada no lugar da camada de saída, como é feito no trabalho de Melamud, Goldberger e Dagan (2016) para *word embeddings*, cuja estrutura é mostrada na Figura 2.8. Uma das *LSTMs* é alimentada com as palavras da sentença da esquerda para a direita e a outra é alimentada com as palavras da sentença da direita para a esquerda, então as saídas geradas são concatenadas e alimentadas em um perceptron multi-camadas, o que permite representar dependências não triviais entre os dois lados do contexto.

2.5 Tipos de Bancos de Dados

Nos últimos anos testemunhou-se um aumento da demanda por agentes conversacionais de *Question-Answering (QA)*, ou seja, modelos voltados para linguagem natural que permitam usuários consultar grandes bancos de dados ou outras fontes de conhecimento.

Tais agentes possuem duas classificações de acordo com a estrutura da fonte de dados as quais são capazes buscar as repostas para as questões do usuário. Um agente é do tipo *KB-QA* se as buscas são feitas em *Knowledge Bases (KBs)*. Caso as buscas sejam feitas em coleções de documentos em linguagem natural, ele é dito um agente *text-QA*. Ambos tipos de bancos de dados serão apresentados nas subseções 2.5.1 e 2.5.2.

Figura 2.8 – BiLSTM com uso de rede neural MLP na saída



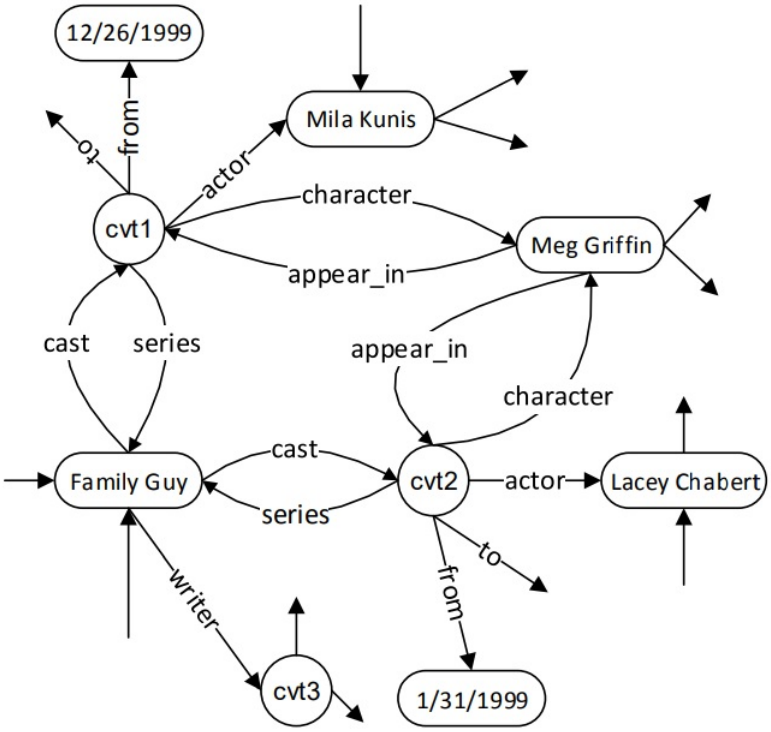
Fonte: Adaptado de Melamud, Goldberger e Dagan (2016)

2.5.1 Knowledge Bases (Bases de Conhecimento)

As *Knowledge Bases* são bancos de dados que possuem fatos do mundo organizados e armazenados de forma estruturada. São bancos de dados ditos estruturados uma vez que os dados tipicamente consistem de uma coleção de triplas da forma sujeito-predicado-objeto (e_1, r, e_2) onde $e_1, e_2 \in \varepsilon$ são entidades e $r \in R$ é uma relação ou predicado. O modelo capaz de responder questões baseando nestes bancos de dados são o componente central de um agente *KB-QA* (GAO; GALLEY; LI, 2018).

Uma *KB* neste formato é também conhecida como *Knowledge Graph (KG)* devido à sua representatividade em forma de grafos, onde as entidades podem ser vistas como os nós do grafo e as relações podem ser vistas como arestas direcionadas que conecta os nós. Um exemplo de subgrafo relacionado ao programa de TV *Family Guy* contido em uma *KB* pode ser visto na Figura 2.9).

Figura 2.9 – Um exemplo de *Knowledge Graph*



Fonte: Yih et al. (2015)

Na literatura, existem muitas *Knowledge Bases* que podem ser empregadas para a consulta de informações. As *Knowledge Bases* podem ser classificadas como de domínio específico, ou seja, seu conteúdo é relacionado a um único grande tema, por exemplo, relacionadas a filmes, medicina, etc., ou podem ser classificadas como de domínio aberto, desta forma seus conteúdos são sobre vários temas que não necessariamente possuem relação entre si.

Como exemplos de *Knowledge Bases* de domínio específico é possível citar *COUNTRIES* (BOUCHARD; SINGH; TROUILLON, 2015), com informações sobre vizinhança de países , *UMLS* (KOK; DOMINGOS, 2007) sobre patologias e tratamentos, *Alyawarra Kinship* (KOK; DOMINGOS, 2007) sobre parentesco de pessoas, *WN18RR* (DETTMERS et al., 2018) sobre relações linguísticas, *Nell-995* (XIONG; HOANG; WANG, 2017) com informações sobre ligas esportivas e *WikiMovies* (MILLER et al., 2016) que contém informações sobre filmes.

Knowledge Bases de domínio aberto em geral são bancos de dados de larga escala, devido a conter assuntos gerais e fatos sobre o mundo. Alguns exemplos das maiores *KBs* encontradas na literatura são *Freebase* (BOLLACKER et al., 2008), *DBPedia* (AUER et al., 2007), e *Yago* (SUCHANEK; KASNECI; WEIKUM, 2007). Além disso, para alguns modelos lidar com bancos de dados grandes pode ser um problema, principalmente por fatores de memória, além

de que estes bancos podem conter muita informação além do que é necessário para responder as questões visadas pelo modelo, então em muitos casos emprega-se versões reduzidas destes. Alguns exemplos de versões reduzidas disponíveis do *Freebase* são *FB15k-237* (TOUTANOVA et al., 2015), *Freebase2M* (BORDES et al., 2015) e *Freebase5M* (BORDES et al., 2015).

Maiores detalhes sobre alguns dos bancos de dados citados aqui podem ser vistos nas Subseções 3.2.1 e 3.1.1.

2.5.2 Bases de Texto Não Estruturado

Os bancos de dados não estruturados são desenvolvidos para a tarefa de *Machine Reading Comprehension (MRC)*, cujo objetivo é desenvolver máquinas que sejam capazes de ler uma passagem ou um conjunto de passagens e então responder qualquer questão sobre as passagens. O modelo *MRC* é o componente central de agentes *text-QA* (GAO; GALLEY; LI, 2018).

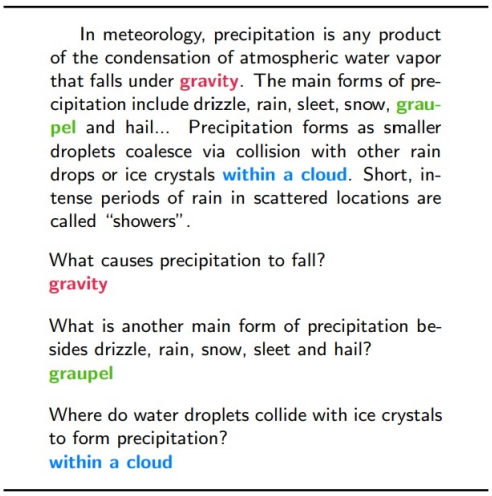
Existem atualmente um grande número de bancos de dados de larga escala deste tipo, criados pela comunidade sobre várias fontes de texto. Os dois mais empregados na literatura atualmente são *SQuAD* (RAJPURKAR et al., 2016) e *MS MARCO* (NGUYEN et al., 2016) devido à sua diversidade e por possuírem um sistema de *leaderboard* próprio, que os tornam mais atraentes ao público e envolvidos da área.

SQuAD é uma sigla para *Stanford Question Answering Dataset* desenvolvido especificamente para *MRC*. Ele consiste de questões construídas por pessoas sobre um conjunto de artigos da *Wikipedia*, onde a resposta para cada questão é um período ou segmento de texto da correspondente passagem de leitura (um exemplo é mostrado na Figura 2.10). O *SQuAD* possui agora uma nova versão que adiciona perguntas que não podem ser respondidas a partir das passagens (RAJPURKAR; JIA; LIANG, 2018).

Já *MS MARCO* é a sigla referente a *MicroSoft MAchine Reading COmprehension*, um banco de dados de larga escala. Todas as questões são amostradas de consultas de usuários reais no mecanismo de busca *Bing*. As passagens de contexto de onde as respostas no banco de dados são derivadas, foram selecionadas usando o mecanismo de busca do *Bing*, e as respostas são geradas por humanos, cujas questões podem ter nenhuma, uma ou múltiplas respostas. Na Figura 2.11 é mostrado um exemplo contendo três passagens, uma questão e uma resposta.

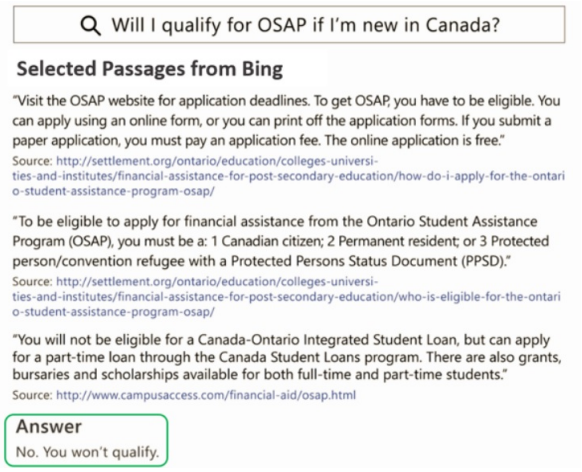
Além destes, é possível citar alguns outros exemplos desenvolvidos para *text-QA* como *DuReader* (HE et al., 2019), *NewsQA* (TRISCHLER et al., 2017), *NarrativeQA* (KOČISKÝ et al., 2018), *SearchQA* (DUNN et al., 2017), *RACE* (LAI et al., 2017), *ARC* (CLARK et al.,

Figura 2.10 – Exemplo passagem-perguntas-respostas SQuAD



Fonte: Rajpurkar et al. (2016)

Figura 2.11 – Exemplo passagens-pergunta-resposta MS MARCO



Fonte: Adaptado de Gao, Galley e Li (2018)

2018) e *ReCoRD* (ZHANG et al., 2018). Informações mais detalhadas sobre suas estruturas são mostradas na Tabela 2.1 que sintetiza como foram geradas as questões, tipos de respostas que possuem, o número de questões e o número de documentos de cada banco de dados.

2.6 Métodos *Deep Learning* aplicados a Grafos

De acordo com Zhang, Cui e Zhu (2020), os métodos de *Deep Learning* aplicados a *Grafos* podem ser divididos em cinco categorias (Figura 2.12) baseados em seus modelos de arquiteturas e em suas estratégias de treinamento, sendo elas, *graph recurrent neural networks, graph convolutional neural networks, graph autoencoders, graph reinforcement learning* e *graph ad-*

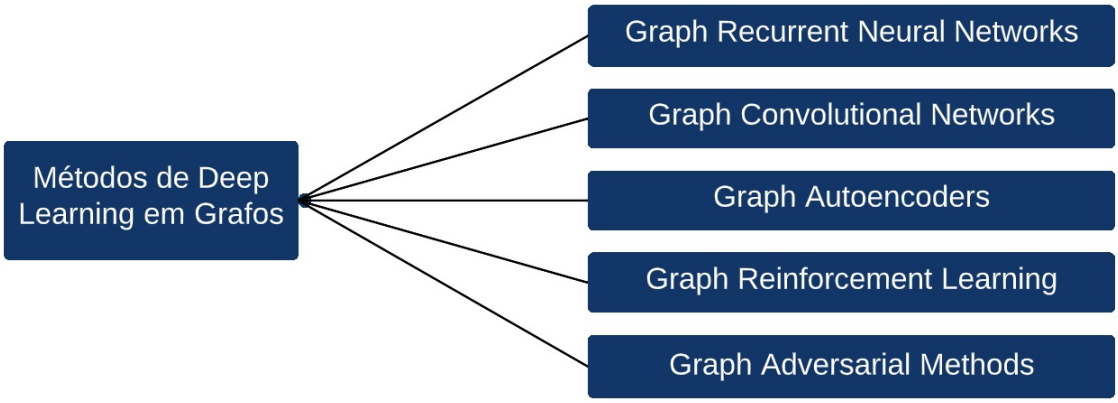
Tabela 2.1 – Informações dos Bancos de dados de texto não estruturados

Dataset	*Question Source*	*Answer*	*#Questionss*	*#Documents*
NewsQA	*Crowd-sourced*	*Span of words*	100k	10k
DuReader	*Crowd-sourced*	*Human generated*	200k	1M
NarrativeQA	*Crowd-sourced*	*Human generated*	46765	1572 *stories*
SearchQA	*Generated*	*Span of words*	140k	6,9M *passages*
RACE	*Crowd-sourced*	*Multiple choice*	97k	28k
ARC	*Generated*	*Multiple choice*	7787	14M *sentences*
SQuAD	*Crowd-sourced*	*Span of words*	100k	536
MS MARCO	*User logs*	*Human generated*	1M	8,8M *passages*

Fonte: Adaptado de Nguyen et al. (2016)

versarial methods. Nas subseções 2.6.1 a 2.6.5 é feita uma breve descrição destas categorias e levantadas algumas de suas aplicações.

Figura 2.12 – Categorização de métodos de Deep Learning em grafos



Fonte: Do Autor (2022)

2.6.1 *Graph Recurrent Neural Networks*

Graph recurrent neural netwoks (Graph RNNs) são desenvolvidas para capturar padrões recorrentes, recursivos e sequenciais de grafos. Podem ser aplicadas, dentre outras, para a tarefa de geração de grafos como feito por You et al. (2018b). Seu custo computacional é muito elevado devido a necessidade de muitos passos do *gradient descent*.

2.6.2 *Graph Convolutional Networks*

Graph convolutional networks (GCNs) são redes que visam imitar as redes neurais convolucionais no âmbito de grafos e, desta forma, elas visam aprender os padrões estruturais locais e globais comuns de grafos. Elas necessitam de estratégias para levar em consideração as características das arestas.

2.6.3 *Graph Autoencoders*

Graph autoencoders (GAEs) tem sido amplamente aplicados em tarefas de aprendizado não supervisionado e são adequados para aprender representações de nós para grafos. A ideia básica por trás dos modelos é observar a matriz de adjacência ou suas variantes como as características brutas dos nós. Permitem também a redução de dimensionalidade.

2.6.4 *Graph Adversarial Methods*

Graph adversarial methods (GAMs): são baseados na ideia aplicada nas *generative adversarial networks (GANs)* de Goodfellow et al. (2014). A ideia é gerar dois modelos ligados onde um é dito gerador e o outro discriminador, sendo o objetivo do gerador gerar dados falsos para tentar enganar o modelo discriminador, e a função do discriminador é distinguir se um dado é real ou gerado pelo modelo gerador. Então ambos modelos se beneficiam sendo treinados juntos via *minimax game*. Podem ser aplicados em conjunto com outras abordagens a grafos e funcionar como fator de regularização.

2.6.5 *Graph Reinforcement Learning*

O que difere os métodos nesta abordagem em relação as outras é basicamente a forma em que os modelos são treinados. O aprendizado por reforço é um paradigma onde um agente inteligente aprende a tomar decisões ótimas ao interagir com um ambiente inicialmente desconhecido (SUTTON; BARTO, 2018).

Para isso, em geral os modelos são formulados como um processo de decisão de Markov *MDP* definidos pela tupla $(\mathscr{S}, \mathscr{A}, \mathscr{R}, \mathscr{P})$, onde $\mathscr{S}$ é o contínuo espaço de estados, $\mathscr{A}$ é o conjunto de ações disponíveis, $\mathscr{P}$ é a matriz de probabilidades de transição de estados e $\mathscr{R}$ é a função de recompensa (GAO; GALLEY; LI, 2018). Em geral são treinados através de *policy gradient* (SUTTON et al., 2000).

You et al. (2018a) e Cao e Kipf (2018) aplicaram esta abordagem na geração de grafos moleculares, onde o processo de geração é visto como um agente de aprendizado por reforço operando no ambiente de geração do grafo através da tomada de decisão de adicionar nós e arestas. As principais diferenças entre os dois modelos é que You et al. (2018a) usaram *graph convolutional networks* para aprender as representações dos nós enquanto em seu trabalho, Cao e Kipf (2018) ao invés de gerarem os grafos através de uma sequência de ações, propuseram gerar diretamente grafos completos.

Para a tarefa de prever os produtos de reações químicas, Do, Tran e Venkatesh (2019) adotaram um agente de aprendizado por reforço onde tal agente agia na seleção de pares de nós no grafo molecular e predizia seus novos tipos de ligação. Para aprender as representações dos nós, foram empregadas *graph convolutional networks* e uma rede neural recorrente para memorizar a sequencia de predição.

Para *knowledge graphs (KGs)* reasoning, Das et al. (2018) e Xiong, Hoang e Wang (2017) empregaram agentes de aprendizado por reforço que tem como objetivo predizer o próximo nó no caminho em cada passo e montam um caminho no *knowledge graph* que é colocado na saída, recebendo recompensa apenas quando chegam ao correto nó de destino. Eles diferem entre si principalmente, pelas informações que utilizam para a busca do destino. Enquanto Das et al. (2018) visa responder uma questão do usuário e portanto tem acesso apenas a uma relação pertinente a questão a ser respondida e o sujeito, Xiong, Hoang e Wang (2017) tem acesso ao sujeito e ao objeto, precisando apenas determinar o caminho, o que torna sua tarefa mais simples.

O modelo *MINERVA* (DAS et al., 2018) demonstra uma abordagem interessante no âmbito da tarefa de *KB-QA* através da aplicação de *Graph Reinforcement Learning*, além de ser referência para muitos outros modelos da literatura, e portanto será abordado mais amplamente na sessão 2.7.1.1 como parte de um modelo *KB-QA* composto. Nas Sub-subseções 2.7.2.3 e 2.7.2.3 são apresentados respectivamente os modelos *Multihop* (LAN; JIANG, 2020) e *NS-CQA* (HUA et al., 2020) que são modelos completos baseados em técnicas de aprendizado por reforço.

2.7 Modelos de Assistentes Virtuais fim-a-fim

Nas subseções seguintes serão apresentados alguns modelos voltados para a tarefa *KB-QA*, também denominados neste trabalho como assistentes fim-a-fim, que são interessantes para a pesquisa, com diferentes abordagens para lidar com as questões em linguagem natural e obtenção de respostas. Como assistentes fim-a-fim, são considerados aqui os modelos para *Question Answering (QA)* capazes de responder a questões em linguagem natural baseados em informações contidas em bases de dados que podem ser interpretadas como grafos de conhecimentos, também denominadas como *Knowledge Bases (KB)*. A estrutura de um modelo fim-a-fim é mostrada na Figura 2.13.

Figura 2.13 – Estrutura geral de um modelo *QA* fim-a-fim

Fonte: Do autor (2022)

Como mostrado na Figura 2.13 um modelo fim-a-fim pode ser dividido em duas etapas. A primeira etapa é responsável pela captura de informações da questão em linguagem natural que possam ser utilizadas pelo modelo para realizar a consulta na *KB*. Em geral, esta etapa envolve a conversão da questão em linguagem natural para triplas ou outra forma de representação mais adequada ao respectivo modelo, e após a conversão, as informações extraídas são aplicadas pelo modelo para consulta e obtenção das respostas adequadas. Nesta pesquisa, são apresentadas duas formas de modelos fim-a-fim, denominados aqui como modelos fim-a-fim compostos e modelos fim-a-fim completos, que são descritos respectivamente nas Subseções 2.7.1 e 2.7.2

2.7.1 Modelos Fim-a-fim Compostos

Estes modelos não foram desenvolvidos a princípio pra lidar com questões em linguagem natural, desta forma, é preciso utilizar-se de alguma técnica para extração de informações das questões em linguagem natural para convertê-las para triplas. Então para que se comportem como um modelo *KB-QA* fim-a-fim, são compostos por um modelo conversor de linguagem natural para triplas e por um modelo de consulta a *KBs* baseado em triplas.

As ferramentas voltadas para consulta a *KBs* a partir de triplas são divididas em 3 tipos de métodos de acordo com Gao, Galley e Li (2018) sendo eles:

a) Métodos simbólicos: são modelos que empregam caminhos relacionais na busca das respostas as questões, onde um caminho relacional é uma sequência $\pi = \{r_1, \ldots, r_k\}$. Um exemplo de caminho relacional é uma sequência de nós $e_1, \ldots, e_{k+1}$ tal que e_i, r_i, e_{k+1} é uma tripla válida;

b) Métodos neurais: são métodos que mapeiam os grafos no espaço neural, onde são feitas as buscas;

c) Métodos de aprendizado por reforço: são modelos que usam agentes baseados em política com estados contínuos baseados nos *KB-embeddings* para atravessar o *KG* e identificar o nó resposta para uma dada questão de entrada.

Alguns exemplos de ferramentas para consultas à *KBs* baseados em triplas interessantes e constituintes do estado da arte da tarefa são o modelo simbólico *CNN-BiLSTM with attention mechanism* (JAGVARAL et al., 2020) que utiliza *BiLSTM* e *CNN* para codificar os caminhos candidatos e usa mecanismos de atenção para relacionar *relational paths* com as relações, o modelo neural *KBAT* (NATHANI et al., 2019) que utiliza *GCN* e mecanismo de atenção para capturar características das entidades e relações em uma vizinhança e o modelo de aprendizado por reforço *MINERVA* (DAS et al., 2018) que usa um agente baseado em aprendizado por reforço para aprender a caminhar no grafo baseado na relação de entrada para encontrar caminhos preditivos.

Como o conversor de linguagem natural para triplas, podem ser tomadas diversas abordagens, seja o desenvolvimento de modelos próprios com o auxílio de ferramentas como a SENNA (COLLOBERT et al., 2011) ou modelos para conversão de triplas contidos em modelos fim-a-fim completos, que em geral são técnicas da etapa de *query understanding* de modelos fim-a-fim *(SP)*, tais como do *Parot* (OCHIENG, 2020) ou do *GAnswer*(HU et al., 2017).

Nesta pesquisa, emprega-se o *MINERVA* (DAS et al., 2018) como o modelo para consultas à *KBs* baseado em triplas, que será melhor descrito na Sub-subseção 2.7.1.1 e como modelo para conversão da questão em linguagem natural para triplas emprega-se a etapa de *query understanding* do modelo completo *Parot* (OCHIENG, 2020) descrito na Subseção 2.7.1.2

2.7.1.1 Modelo para Consulta à *Knowledge Bases* Baseado em Triplas *(MINERVA)*

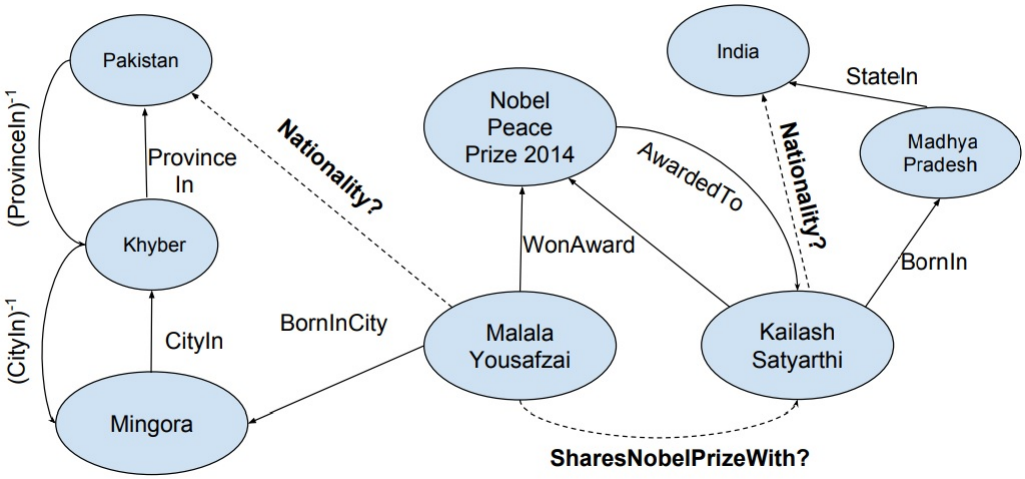
O MINERVA é um modelo desenvolvido para realizar a busca de informações em *Knowledge Graphs* baseado em questões fornecidas pelos usuários em forma de triplas. Diferentemente da grande maioria dos modelos desenvolvidos para consultas a *Knowledge Graphs*, que são voltados para tarefas mais simples, como predizer uma relação ausente entre duas entidades ou avaliar a existência de uma tripla dentro de um *KG*, possíveis de serem realizadas simplesmente determinando-se os caminhos entre duas entidades pré-estabelecidas, o *MINERVA*

determina a segunda entidade que é a resposta a uma questão baseando-se em uma entidade de origem e uma relação (DAS et al., 2018).

A tarefa solucionada pelo *MINERVA* é mais difícil por não ser possível utilizar-se de caminhos aleatórios entre pares de entidades fixas ou aprender a buscar caminhos entre tais pares, uma vez que a entidade destino é desconhecida e podem existir uma infinidade de caminhos combinatórios partindo-se da entidade origem. Para a tarefa, é utilizada a abordagem grafos com aprendizado por reforço. O modelo aprende a navegar no grafo condicionado na questão de entrada para determinar caminhos preditivos.

Como exemplo da tarefa, considerando-se o pequeno grafo na Figura 2.14, pode-se responder a questão *"Who did Malala Yousafzai share her Nobel Peace prize with?"* a partir do seguinte caminho de raciocínio: Malala Yousafzai → WonAward → Nobel Peace Prize 2014 → AwardedTo → Kailash Satyarthi. Note que as linhas tracejadas na Figura 2.14 não são relações existentes diretamente no grafo. O objetivo é aprender automaticamente tais caminhos de raciocínio em *KGs*. O problema de aprendizado é estruturado como um problema de responder *queries*, ou seja, responder questões da forma (Malala Yousafzai, SharesNobelPrizeWith, ?).

Figura 2.14 – Exemplo de grafo e questões sobre suas entidades



Fonte: Das et al. (2018)

O modelo visa eficientemente procurar caminhos para fornecer as respostas usando aprendizado por reforço condicionado à questão de entrada, eliminando qualquer necessidade de caminhos pré-calculados. Dado um grande *KG*, aprende-se uma política, que dada a questão (e_1,r, ?), parte-se de e_1 e aprende a andar até o nó resposta escolhendo-se tomar uma aresta de relação rotulada em cada passo, condicionado à relação da questão e no histórico do caminho

inteiro. O objetivo desse aprendizado por reforço é aprender a tomar uma sequência ótima de decisões para maximizar a recompensa esperada atingindo-se o correto nó resposta.

O problema é modelado como um processo de decisão de Markov parcialmente observável e determinístico em um *KG*. Assim, pode ser visto como a quíntupla $(\mathscr{S}, \mathscr{O}, \mathscr{A}, \delta, R)$. As variáveis são descritas como:

a) *States* ($\mathscr{S}$): estado para codificar a *query* (e_{1q}, r_q), a resposta (e_{2q}) e a atual localização do agente de aprendizado por reforço e_t. Assim, um estado $S \in \mathscr{S}$ é representado por $S = (e_t, e_{1q}, r_q, e_{2q})$ e o espaço de estados consiste de todas as combinações válidas;

b) *Observations* ($\mathscr{O}$): o estado completo do ambiente não é observado. Assim, o que o agente conhece é sua atual localização e_t e a entidade e relação da *query* (e_{1q}, r_q), mas não a resposta (e_{2q}) que permanece oculta. Dessa forma, a função de observação é definida como $\mathscr{O}(S = (e_t, e_{1q}, r_q, e_{2q})) = (e_t, e_{1q}, r_q)$;

c) *Actions* ($\mathscr{A}$): o conjunto de ações possíveis de um estado S consiste de todas as arestas do vértice e_t no grafo. Assim, $\mathscr{A}_S = (e_t, r, v)$, em que r é uma relação (aresta no grafo) e v é o vértice (nó) de destino ao se tomar a aresta r. Dessa forma, em cada estado, o agente pode decidir qual aresta tomar baseado na relação e vértice de destino. Na implementação foi adicionada ainda a ação *"NO_OP"* que sai de um nó e retorna para ele próprio, o que permite que o agente possa permanecer no mesmo estado entre vários passos de tempo, uma vez que o número de passos que o agente executa na busca de uma resposta é um número fixo T. Além disso, foram adicionadas as triplas inversas no grafo para que se permita ao agente desfazer passos potencialmente errôneos;

d) *Transition* (δ): o ambiente evolui deterministicamente apenas atualizando o estado para o novo vértice destino, de acordo com a aresta selecionada pelo agente. A *query* e resposta permanecem as mesmas. Assim, a transição é $\delta(S, A) = (v, e_{1q}, r_q, e_{2q})$ onde $S = (e_t, e_{1q}, r_q, e_{2q})$ e $A = (e_t, r, v)$;

e) *Recompensas* (R): é atribuída uma recompensa terminal $+1$ se a atual localização é a resposta correta e 0 caso contrário. Dessa forma, se $S_T = (e_t, e_{1q}, r_q, e_{2q})$ é o estado final, então recebe-se uma recompensa de $+1$ se $e_t = e_{2q}$ senão 0.

Para solucionar o processo de decisão descrito acima, aplica-se uma política histórico-dependente não estacionária aleatória $\pi = (d_1, d_2, \ldots, d_{T-1})$, onde $d_t : H_t \to \mathscr{P}(\mathscr{A}_{S_t})$ e o histórico $H_t = (H_{t-1}, A_{t-1}, O_t)$ é apenas um sequência de observações e ações tomadas.

A política empregada é parametrizada via rede *LSTM*. Assim um agente baseado em *LSTM* codifica o histórico H_t como um vetor contínuo $h_t \in \mathbb{R}^{2d}$, onde d é o número de dimensões dos vetores de *embeddings* usado para as relações e também para as entidades. Tem-se ainda a matriz de *embeddings* $r \in \mathbb{R}^{|\mathscr{R}| \times d}$ e $e \in \mathbb{R}^{|\mathscr{E}| \times d}$ para as relações binárias e entidades, respectivamente. O *embedding* do histórico H_t é atualizado de acordo com a dinâmica da *LSTM*:

$$h_t = LSTM(h_{t-1}, [a_{t-1}; o_t]), \tag{2.11}$$

em que $a_{t-1} \in \mathbb{R}^d$ e $o_t \in \mathbb{R}^d$ denotam o vetor de representação para ação/relação no tempo $t-1$ e observação/entidade no tempo t respectivamente e $[;]$ representa a concatenação dos vetores. Note que $a_{t-1} = r_{A_{t-1}}$ é o *embedding* da relação correspondente a aresta escolhida pelo agente no tempo $t-1$ e $o_t = e_{e_t}$ é o *embedding* da entidade correspondente ao vértice em que o agente se encontra no tempo t.

Baseado no *embedding* do histórico h_t, a rede política decide qual das ações disponíveis escolher, condicionada à relação da *query*. O *embedding* para cada ação $A \in \mathscr{A}_{S_t}$ é $[r_l; e_d]$, neste caso o índice l indica *label* e o índice d indica *destination*, e empilhando-se os *embeddings* para todas as arestas de saída, obtém-se a matriz A_t. Estes *embeddings* são entradas para uma rede *feed-forward* parametrizada de duas camadas com ReLU, que utiliza a atual representação do histórico h_t e o *embedding* para a relação da *query* r_q e provê na saída uma distribuição de probabilidades sobre as possíveis ações das quais uma determinada ação discreta é amostrada. Assim, a equação que descreve esse procedimento é:

$$d_t = softmax(A_t(W_2 ReLU(W_1[h_t; o_t; r_q]))), \tag{2.12}$$

$$A_t \sim Categórico(d_t). \tag{2.13}$$

É importante observar que os nós no grafo não possuem uma ordem ou número fixo de arestas saindo deles, assim as probabilidades de decisão d_t se baseiam no *simplex* de tamanho $|\mathscr{A}_{S_t}|$. Para sintetizar, de acordo com as Equações (2.11) a (2.13), as variáveis treináveis do modelo são: os parâmetros da rede *LSTM*, os pesos W_1, W_2, correspondentes *bias* que não são

mostrados nas equações por conveniência, além das matrizes de *embeddings* que formam os parâmetros θ da rede política.

Para o treino da rede política π_θ é empregada a função custo da Equação (2.14), onde deseja-se determinar os parâmetros θ que maximizem a recompensa esperada:

$$J(\theta) = \mathbb{E}_{(e_1,r,e_2)\sim D}\mathbb{E}_{A_1,\ldots,A_{\gamma-1}\sim\pi_\theta}[R(S_T|S_1 = (e_1,e_1,r,e_2)], \tag{2.14}$$

em que $\mathbb{E}$ é uma expectativa, D é a quantidade de triplas no conjunto de treino e γ é um fator de desconto. Para solucionar o problema de otimização da Equação (2.14), emprega-se reforço como abaixo:

a) A primeira expectativa ($\mathbb{E}_{(e_1,r,e_2)\sim D}$) é substituída com a média empírica sobre o conjunto de treino;

b) A segunda expectativa ($\mathbb{E}_{A_1,\ldots,A_{\gamma-1}\sim\pi_\theta}$) é aproximada executando múltiplos *rollouts* para cada exemplo de treino, ou seja, cada exemplo de treino é executado várias vezes seguidas. O número de *rollouts* é fixo, onde optou-se por 20 em Das et al. (2018);

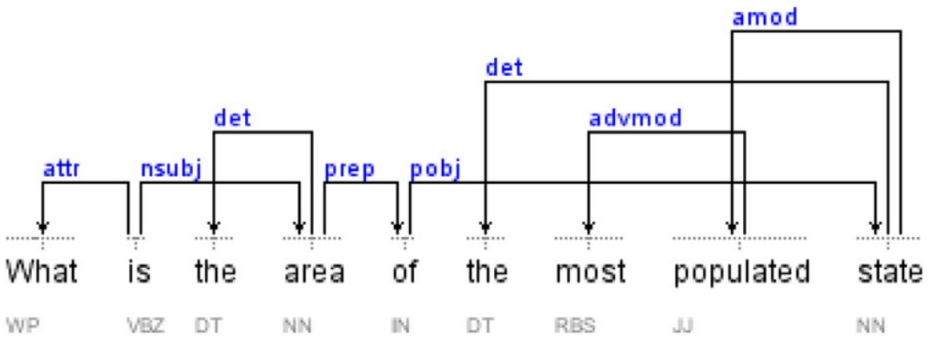
c) Para redução da variância, utiliza-se uma *baseline* variável de controle aditivo. Como *baseline* empregou-se a média móvel da recompensa descontada cumulativa, cujos pesos são selecionados como hiper-parâmetros;

d) Para aumentar a diversidade dos caminhos gerados em treino pela política, adiciona-se ainda um termo de regularização de entropia à função custo (Equação (2.14)) escalado por uma constante β.

2.7.1.2 Modelo para Conversão das Questões em Linguagem Natural para Triplas (*Parot*)

O modelo Parot (OCHIENG, 2020) é um modelo fim-a-fim voltado para a tarefa de *Knowledge Base - Question Answering* do tipo *synonym based*. Sua principal contribuição é uma *framework* baseada em árvores de dependências das questões, capaz de lidar com questões dos usuários que contém sentenças compostas, negações, adjetivos escalares e listas numeradas. Um exemplo de uma árvore de dependências para a questão "*What is the area of the most populated state.*" é mostrado na Figura 2.15.

Figura 2.15 – Exemplo de árvore de dependências para uma questão



Fonte: Ochieng (2020)

Este modelo emprega várias regras heurísticas baseadas em árvores de dependências para converter as questões do usuário da linguagem natural para triplas, então as triplas geradas são processadas por um *lexicon* para correspondência com a *KB*, e então geram-se as *SPARQLs* que serão empregadas para obtenção da resposta. Sua estrutura pode ser dividida em quatro partes, sendo elas, identificação dos alvos da questão, identificação de triplas relacionais e não relacionais, identificação de triplas formadas por adjetivos e o *lexicon*:

a) Identificação dos alvos da questão: a primeira tarefa se trata da identificação das palavras alvos na questão. A palavra alvo é uma variável que é colocada diretamente após *SELECT* em uma consulta *SPARQL* (vide exemplo na Figura 2.16). Para este fim, aplica-se diferentes regras baseadas na árvore de dependências se a questão é do tipo *Wh*, ou seja, questões que iniciam com palavras inciadas em *Wh* tais como *what, when, where, who,*e etc., ou se a questão não é do tipo *Wh*;

b) Identificação de triplas relacionais e não relacionais: a consulta *SPARQL* é composta por um conjunto de triplas (sujeito, predicado, objeto), que são colocadas após a palavra chave *WHERE* (vide exemplo na Figura 2.16), e podem ser vistas também como um grafo da questão em linguagem natural. Nesta etapa, o interesse é identificar tais triplas na questão do usuário, através de regras baseadas na árvore de dependências da questão, considerando-se duas categorias:

   – Consultas baseadas em frases relacionais: questões que contém ao menos uma frase relacional ligando dois nominais, onde uma frase relacional pode ser um verbo transitivo, ou um verbo intransitivo seguido por um complemento preposicional. Desta forma, uma frase relacional considerada pode ser um verbo, um

verbo seguido diretamente por uma preposição, ou um verbo seguido por substantivos, adjetivos ou advérbios terminando em uma preposição;

- Consultas baseadas em frases não relacionais: questões que não possuem uma frase relacional conectando nenhum de seus nominais, por exemplo, na questão "*What is the area of the most populated state*". Em geral, são frases que não possuem outro verbo além do verbo *to be* (ser/estar);
- **Observação**: as questões podem ser compostas, e caso sejam, são divididas em sentenças simples, então aplica-se as devidas regras para estas sentenças simples, e após isto, combina-se as triplas da maneira mais adequada;

c) Identificação de triplas formadas por adjetivos: este passo adiciona triplas complementares às geradas no passo anterior ao padrão *SPARQL* da questão do usuário, obtidas ao lidar com os adjetivos das sentenças. As regras baseadas na árvore de dependências da questão para gerar estas triplas adicionais são desenvolvidas considerando-se duas classes de adjetivos:

- Adjetivos escalares: são os adjetivos que comunicam a ideia de escala. Aos quais em geral podem possuir quantidades numéricas associadas a eles, por exemplo comprimento, altura, etc.. Lida também com superlativos;
- Adjetivos não-escalares: adjetivos que não comunicam ideia de escala, por exemplo nacionalidades, profissões, etc.;

d) *Lexicon*: as palavras das triplas geradas para as questões do usuário precisam ser mapeadas para as correspondentes palavras empregadas na *Knowledge Base*, ou seja, as triplas do usuário obtidas da questão fornecida pelo usuário devem ser convertidas para os devidos predicados e entidades existentes na *KB*. Tal mapeamento é feito utilizando-se um *lexicon* desenvolvido pelos próprios autores (OCHIENG, 2020) utilizando-se uma ferramenta denominada *lemon* (*LExical Model for ONtologies*) de McCrae, Spohr e Cimiano (2011), combinada com as técnicas dispostas no trabalho de Walter, Unger e Cimiano (2013) para geração semi automática do *lexicon*. Então o *lexicon* desenvolvido é empregado para desambiguação das palavras das triplas, identificação de adjetivos escalares positivos e negativos nas triplas da questão, mapeamento das palavras de maneira a minimizar o espaço de busca da resposta, e resolução da exata posição de um conceito como modelado na correspondente *KB*.

Após a formação e seleção das devidas triplas da questão, para completar a informação da consulta *SPARQL* para a questão, são empregados *query filters*, ou seja, informações adicionais contidas na questão submetida pelo usuário, que ajudam a estreitar mais os resultados para encontrar a resposta requerida pelo usuário. Tais informações são a respeito de operadores lógicos, adjetivos, negações e números contidos na questão do usuário.

O modelo é avaliado em três diferentes bancos de dados, o banco de dados *QALD-9* e o banco de dados *Geobase* para questões complexas e o banco de dados *SimpleQuestions* para questões simples, sendo que apenas um subconjunto de 200 questões do *SimpleQuestions* foi utilizado. A *Knowledge Base* que emprega é a *Freebase*. O modelo apresenta resultados de desempenho comparáveis e em alguns casos superiores a modelos fim-a-fim estado da arte como o *GAnswer* (HU et al., 2017) considerando-se estes *datasets*. Os procedimentos completos de análise e categorização das questões, incluindo as regras empregadas, bem como os resultados do modelo, podem ser vistos na íntegra no trabalho de Ochieng (2020).

A maior vantagem do *Parot* é a sua capacidade de lidar com questões complexas, além de fornecer estratégias que permitem a extração de vários tipos diferentes de triplas das questões do usuário, tais como triplas relacionais, não relacionais, envolvendo adjetivos, superlativos, envolvendo negações e etc., que permitem uma maior cobertura dos tipos de questões que podem ser respondidas pelo modelo, além de ser capaz de lidar com questões compostas e de maior complexidade, onde se destaca em relação a outros modelos em desempenho. Por outro lado, o modelo possui uma etapa de análise e categorização significantemente lenta, não é capaz de lidar com questões iniciadas com "*When*", ou seja, questões temporais, apresenta maus resultados quando lidando com questões envolvendo agregação e o *lexicon* empregado não é escalável para grandes ontologias (*KBs*) (OCHIENG, 2020).

O processo de aplicação deste modelo como parte do modelo composto desenvolvido na pesquisa é descrito na Subseção 3.1.2

2.7.2 Modelos Fim-a-fim Completos

Modelos fim-a-fim completos são modelos que foram desenvolvidos por seus autores já levando em consideração questões em linguagem natural, e assim, cada modelo aplica o método mais adequado para sua representação da questão. Os modelos fim-a-fim completos podem ser divididos em três ramos principais de acordo com seus métodos de abordagem a tarefa *KB-QA* sendo eles (FU et al., 2020; LAN et al., 2021):

a) Método de Análise Semântica (*Semantic Parsing-based method (SP)*): modelos que se baseiam neste método, em geral, representam a questão por uma forma logica simbólica, em geral utilizando-se ferramentas de análise sintática e de classificação de palavras, e então executa-as contra a *KB*;

b) Método de Recuperação de Informação (*Information Retrieval-based method (IR)*): os modelos que empregam este método, em geral, primeiro determinam as entidades de interesse contidas na questão em linguagem natural, fazendo-se as devidas correspondências com as entidades da *KB*. Após feito este procedimento, extraem os subgrafos relacionados a entidade tópico contida na questão, que é a entidade raiz do subgrafo, e trata os nós presentes nos subgrafos extraídas como respostas candidatas. Finalmente, baseados em características extraídas das questões e respostas candidatas, empregam *score functions* para modelar as relevâncias semânticas e determinar a resposta final.

c) Métodos de Análise Semântica Neurais *Neural Semantic Parsing-based methods (NSP)*: os modelos que se baseiam neste método constroem suas análises semânticas baseados em redes neurais e/ou aprendizado por reforço, ao invés de se basearem em regras manualmente definidas ou *templates*. Em geral, estes modelos mapeiam as questões para formas lógicas intermediárias e então as converte para *queries* tais como *SPARQLs*.

2.7.2.1 Modelos Fim-a-fim Baseados em *Semantic Parsing*

A princípio, os modelos baseados em *Semantic Parsing* podem ser divididos em quatro categorias, de acordo com Cui et al. (2019), sendo elas:

a) *Rule based*: abordagens baseadas em regras. São definidas como as abordagens que mapeiam questões para predicados utilizando regras manualmente construídas, o que garante uma alta precisão mas uma baixa cobertura da variedade de questões, uma vez que criar regras para um grande número de questões é intangível. Um exemplo é mostrado na Figura 2.16, que mostra a descrição de uma regra, observe que neste caso uma regra equivale a um *template*, para a questão "*What is the name of the movie which has the genre [genre_value]?*", onde o que está contido entre colchetes é uma "entidade variável" que será obtida da pergunta do usuário para formulação do padrão *SPARQL* ao qual o *template* está associado para consulta pelo modelo na *KB* para obtenção da resposta.

Figura 2.16 – Exemplo de conversão de questão em linguagem natural para *SPARQL (Rule Based)*

T1-1: *What is the name of the movie which has the genre [genre_value]?*

```
SELECT    ?movieName
WHERE {
?Movie    prefix:name          ?movieName.
?Movie    prefix:genre         “[genre_value]”^^<xsd:string>. }
```

Fonte: Ou et al. (2008)

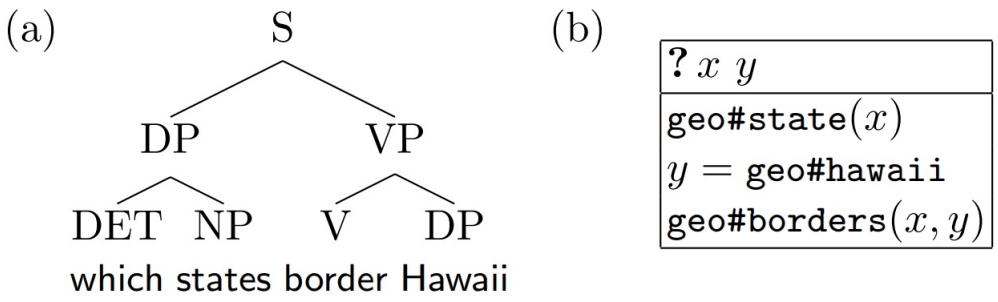
Como mostrado na Figura 2.16, é necessário um *template* para cada relação e cada predicado pode possuir muitas formas de ser descrito em linguagem natural, que devem ser cobertas por tais *templates* que devem ser gerados um a um, o que torna tão difícil a cobertura de um grande número de questões;

b) *Keyword based*: abordagens baseadas em palavras chave ou *Keyword based* são métodos que usam palavras chave na questão e as mapeiam para predicados, também chamados relações, por meio de correspondência de palavras.

Em geral, a seleção das palavras chaves na questão se dá através de regras manualmente desenvolvidas, baseadas nas árvores de dependências das palavras na questão, e apesar de ainda depender de tais regras manuais, exigem uma quantidade limitada delas e possuem uma capacidade maior de automação do processo de conversão para triplas/*SPARQL*, além de conseguir lidar com pequenos erros de correspondências de palavras. Observe que árvores de dependências são algoritmos amplamente estudados atualmente e estão disponíveis em várias bibliotecas de *Machine Learning* pré-treinados em grandes vocabulários, o que facilita seu emprego. Um exemplo para ilustrar a abordagem *keyword based* é mostrado na Figura 2.17 para a questão "*Which states border Hawaii*".

Na Figura 2.17 (a) é mostrada a árvore de dependências da questão e na Figura 2.17 (b) são mostradas as informações obtidas baseadas na árvore de dependências onde *state* é uma relação, *borders* é outra relação, *hawaii* é uma entidade e x é a resposta a ser determinada na *KB* pelo modelo.

Figura 2.17 – Exemplo de conversão de questão em linguagem natural para triplas *(Keyword Based)*



Fonte: Unger e Cimiano (2011)

Note que todas as palavras selecionadas possuem correspondentes por escrita muito próximos na questão e suas representações na *KB*, mas não poderiam ser determinadas se por exemplo na questão tivessem sido usados seus sinônimos inexistentes na *KB*.

c) *Synonym based*: os métodos baseados em sinônimos são uma extensão dos métodos baseados em palavras chave (*keyword based*), levando em consideração os sinônimos dos predicados e é atualmente o método mais empregado nos modelos recentes (CUI et al., 2019). Em geral, os modelos geram os sinônimos para cada predicado e então encontram mapeamentos entre as questões e estes sinônimos, onde cada sinônimo pode ser uma frase, uma palavra ou uma sequência de predicados. Um exemplo de tais mapeamentos pode ser visto na Figura 2.18.

Figura 2.18 – Exemplo de conversão de linguagem natural para triplas *(Synonym Based)*

Relation Phrases	Predicates or Predicate Paths	Confidence Probability
“be married to”	<spouse>	1.0
“play in”	<starring>	0.9
“play in”	<director>	0.5
“uncle of”	<hasChild> <hasChild> <hasChild>	0.8
… …	… …	… …

Fonte: Zou et al. (2014)

Na Figura 2.18 são mostradas as frases relacionais na primeira coluna, que podem ser obtidas de maneira semelhante ao processo empregado por métodos baseados em pa-

lavras chave das questões em linguagem natural, na segunda coluna são mostrados os predicados, ou seja, relações existentes na *KB* que são considerados sinônimos das frases relacionais, e na terceira coluna é expressa a similaridade inferida por uma métrica de similaridade, neste caso expressa como probabilidade de confiança.

É observável que diferente do método *keyword based*, no método *synonym based* a frase relacional obtida da questão em linguagem natural e o predicado na *KB* podem não possuir semelhança em escrita. Ainda assim, existem situações em que o método *synonym based* é limitado, principalmente por não conseguir resolver ambiguidades. Por exemplo, considerando-se a frase relacional *how many people* nas questões "*How many people live in Honolulu?*" e "*How many people visit New York each year?*", a primeira frase relacional pode ser mapeada para o predicado "população", o que não é o caso da segunda que se refere a "número de passageiros" mas o método não permite diferenciá-los;

d) *Template based*: este método tenta cumprir o que falta nos anteriores, sendo mais automatizado que o método *rule based* e melhor que os métodos *keyword based* e *synonym based* para representar completamente a intenção semântica das questões, além de possuir maior potencial para questões mais complexas.

   Neste método, uma questão é transformada da linguagem natural para uma representação interna que captura as semânticas e intenção da questão denominada *template* (de maneira automática) e então para cada *template*, aprende-se como mapeá-lo em uma *SPARQL query* contra uma *KB*. Para gerar um *template* substitui-se na questão em linguagem natural as entidades por seus conceitos (exemplo na Figura 2.16) e depois mapeia-se tais *templates* para predicados da *Knowledge Base* de maneira semelhante ao processo realizado para *semantic parsing*. O processo é descrito de maneira mais completa no trabalho de Cui et al. (2019), que utilizam o banco de dados *Yahoo! Answers* para geração dos *templates* e para auxiliar no mapeamento para os devidos predicados.

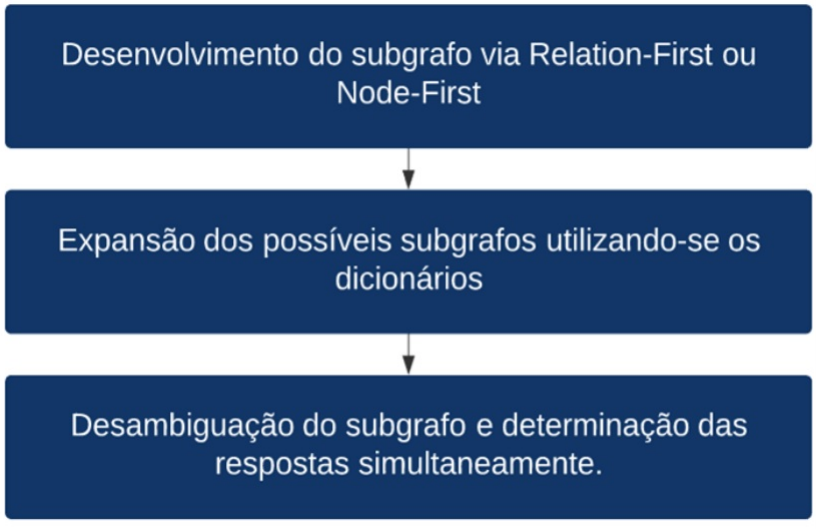
Na Sub-subseção 2.7.1.2 foi apresentado o *Parot* (OCHIENG, 2020) e a seguir é descrito o modelo *GAnswer* (HU et al., 2017), ambos modelos que se baseiam no método *synonym based*.

GAnswer

Envolvidos nas etapas de geração da *SPARQL* para a questão (*question understanding*) e de avaliação da *SPARQL* gerada para determinação da resposta (*query evaluation*) existem dois desafios principais, *phrase linking*, que se refere a possibilidade de uma entidade ou um predicado em linguagem natural, que podem ser compostos por várias palavras, poderem ser relacionados a vários itens na *Knowledge Base*, o que gera a necessidade de desambiguação, ou seja, decidir dentre as possíveis correspondências entre a questão em linguagem natural e os itens na *KB* aqueles mais adequados, e o desafio da *composition*, que se refere a decisão de como os predicados e entidades se relacionam entre si na questão, uma vez que dentro de uma questão em linguagem natural podem haver vários predicados e entidades.

O modelo *GAnswer* (HU et al., 2017), ao invés de gerar diretamente a *SPARQL* da questão na etapa de *question understanding*, gera representações gráficas da questão (*query graph*) que permitem representar a intensão da consulta do usuário, mas também permitem ambiguidades do tipo *phrase linking*, relacionada ao significado das palavras, e ambiguidade composicional, relacionada a estrutura do *query graph*, que são resolvidas na etapa de *query evaluation*. A estrutura geral do modelo é mostrada na Figura 2.19.

Figura 2.19 – Estrutura geral do *GAnswer*



Fonte: Do autor (2022)

O diagrama da Figura 2.19 mostra o processo para resposta de questões em linguagem natural empregado pelo *GAnswer*, onde os primeiros dois blocos correspondem a etapa *question understanding* e o último bloco corresponde a etapa *query evaluation*. Para a interpretação da questão em linguagem natural pelo modelo, desenvolveram-se duas abordagens distintas para

obtenção do subgrafo, sendo ambas definidas em regras baseadas na árvore de dependências da questão, denominadas *Relation-First* e *Node-First*:

a) *Relation-First*: primeiro são extraídas as relações semânticas baseadas na estrutura da árvore de dependências para construir um *query graph* semântico Q^S. As relações semânticas aqui mencionadas se referem a triplas do tipo ($arg1$, rel, $arg2$), onde rel é uma frase relação, e $arg1$ e $arg2$ são as entidades associadas a tal relação. Em Q^S, duas arestas compartilham um ponto final comum se as duas relações correspondentes compartilham uma entidade. Cada nó (entidade) e aresta (relação) pode ter vários candidatos em Q^S após a expansão do subgrafo e então as ambiguidades de *phrase linking* são resolvidas quando as devidas correspondências são estabelecidas entre Q^S e um subgrafo na *KB*. Observe que Q^S possui todas as relações entre entidades definidas previamente, ou seja, o grafo é criado pressupondo-se a possibilidade de apenas uma estrutura fixa, então não há a necessidade de desambiguação composicional da estrutura do grafo gerado para a questão, uma vez que assume-se que o grafo da consulta pode ser fixado unicamente na etapa de *question understanding*.

b) *Node-First*: o principal objetivo desta abordagem é lidar com relações implícitas ou incertas que podem haver na questão do usuário, o que não ocorre na abordagem *relation-first*. Esta abordagem ao contrário da anterior não parte da extração de relações da questão e sim da extração das entidades (nós) presentes na questão, e então os conecta para formar um *query graph*. Na formação de tal *query graph*, diferentemente da abordagem *relation-first*, é permitida a ambiguidade composicional da estrutura do *query graph* com a inserção de arestas incertas e possivelmente inexistentes formando-se assim na etapa de *question understanding* um *query graph* universal Q^U de Q^S que inclui também arestas incertas. Então ao fazer a correspondência entre o *query graph* e a *KB* são permitidas a não correspondência para algumas arestas em Q^U. Desta forma, a ambiguidade de *phrase linking* e a ambiguidade composicional são resolvidas juntas quando uma correspondência aproximada é encontrada, ocorrendo assim, na etapa *query evaluation*.

Verifica-se que a abordagem que gera os melhores resultados é a abordagem *node-first* através de alguns experimentos descritos no próprio artigo do *GAnswer* (HU et al., 2017). O modelo apresenta resultados competitivos em relação a outros modelos estado-da-arte para o

banco de questões *WebQuestions* utilizando-se como *Knowledge Base* o banco de dados *Freebase*, e para o banco de questões (QALD-6) utilizando-se como *KB* o banco de dados *DBpedia*. Além disso, mostra-se que o modelo *GAnswer* tende a obter melhor desempenho para questões de maior complexidade que os outros modelos estado-da-arte. Os resultados completos podem ser vistos em (HU et al., 2017).

Os códigos necessários para a reprodução e aplicação do modelo *GAnswer* estão disponibilizados na íntegra por seus autores no *Github* baseados em linguagem java. Os maiores problemas referentes aos códigos fornecidos é a grande dependência de *APIs* externas, como a Gstore, *API* necessária para acesso das informações da *KB* e a DBpedia lookup, *API* necessária para expansão dos grafos, que para serem implementadas localmente demandam de dados ainda indisponíveis e máquinas de alto-desempenho, com necessidade de alta capacidade de processamento e memória.

2.7.2.2 Modelos Fim-a-fim Baseados em *Information Retrieval*

Os modelos baseados em recuperação de informação (*IR*), diferentemente dos modelos baseados em análise semântica (*SP*), em geral trabalham com as questões em linguagem natural no espaço neural, e desta forma a etapa de conversão para triplas, equivalente a extração de informação da questão, não é separável do modelo para busca das respostas. Desta forma não necessitam do emprego de regras e/ou *templates* manualmente definidos, no entanto, tais modelos tornam difícil sua interpretabilidade. Um exemplo interessante de modelo fim-a-fim baseado em recuperação de informação é o *BAMnet* (CHEN; WU; ZAKI, 2019) que é descrito mais profundamente a seguir.

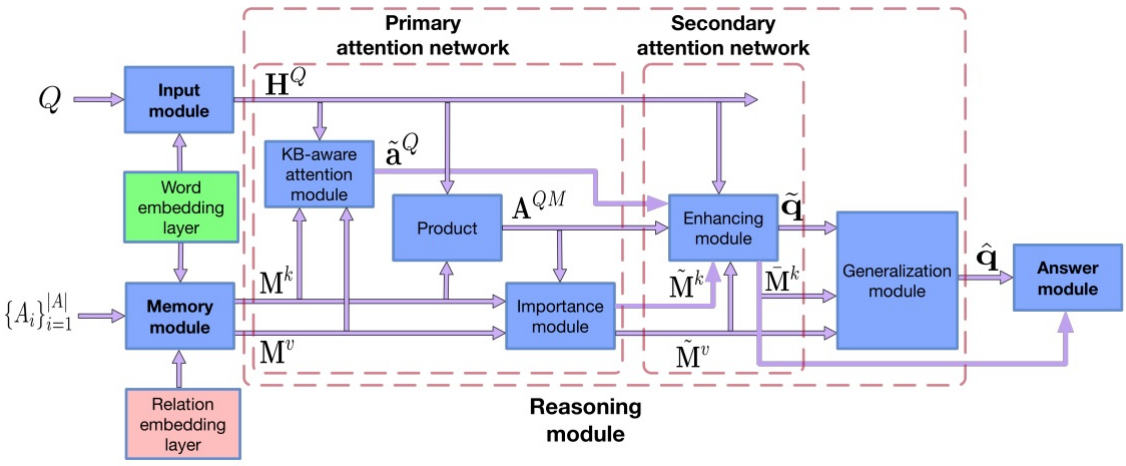
BAMnet

O BAMnet (*Bidirectional Attentive Memory Network*) é um modelo para *Question Answering* fim-a-fim baseado em *embeddings*, ou seja, todas as operações desde a conversão da questão em linguagem natural, seleção das características e busca da resposta na *Knowledge Base* são realizadas no espaço neural (CHEN; WU; ZAKI, 2019). O diferencial do modelo é que ao contrário de outros modelos baseados em *embeddings*, ao fazer a codificação das questões e dos subgrafos da *KB* ele tenta capturar as interações mútuas entre as questões e a referente *KB* através de mecanismos de atenção bidirecionais.

Diferente de outros modelos como o Parot (OCHIENG, 2020) e o GAnswer (HU et al., 2017), o BAMnet não necessita de recursos externos, tais como servidores para busca de entidades para expansão das *queries*. Além disso, diferente de tais modelos, ele é capaz de lidar com a tarefa sem o uso de muitas *handcrafted rules*, ou seja, não necessita de regras desenvolvidas por humanos para obtenção de aspectos chave das questões, como é feito pela grande maioria dos modelos fim-a-fim voltados para *QA* que utilizam-se de várias dessas regras, em geral baseadas nas árvores de dependências das questões (*dependency trees*), como também é feito em Ochieng (2020) e Hu et al. (2017).

A estrutura do modelo *BAMnet* é mostrada na Figura 2.20. Como é possível ver, o modelo é composto por vários módulos menores, mas pode ser descrito em geral como quatro macro-módulos, sendo eles, o *input module*, o *memory module*, o *reasoning module* e o *answer module*:

Figura 2.20 – Estrutura geral do modelo BAMnet



Fonte: Chen, Wu e Zaki (2019)

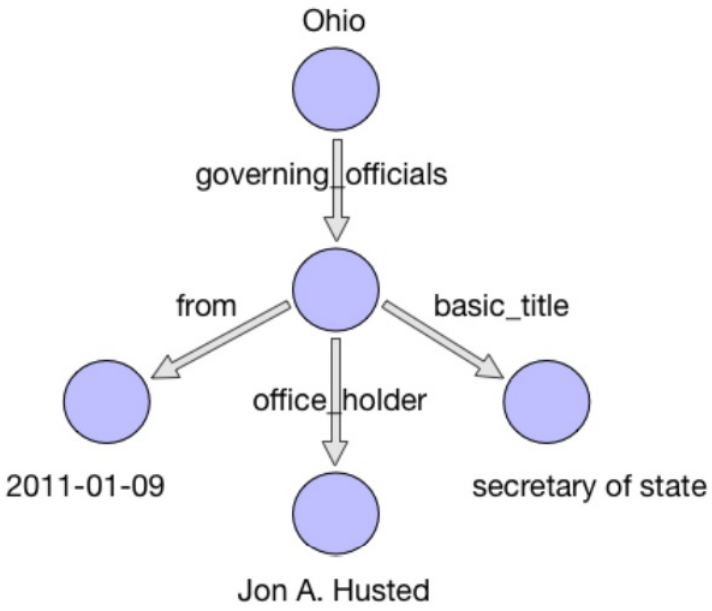
a) *Input module*: módulo de entrada do modelo. Uma questão $Q = \{q_i\}_{i=1}^{|Q|}$ é representada como uma sequência de *embeddings* das palavras (q_i) aplicando-se uma *embedding layer*. Então emprega-se *BiLSTM* para codificar a questão como $H^Q \in \mathbb{R}^{d \times |Q|}$ que é a sequência de estados ocultos gerados pela *BiLSTM*;

b) *Memory module*: este módulo pode ser compreendido em duas sub-etapas, a seleção dos candidatos a resposta (*candidate generation)* e a representação da *Knowledge Base*;

   – *Candidate Generation*: apesar de todas as entidades contidas na *KB* puderem ser candidatas à resposta, isto geraria um custo computacional muito elevado. Desta

forma, considera-se como candidatas à resposta apenas às entidades que estejam próximas da principal entidade tópico da questão na *KB*. Desta forma, após ser determinada a entidade tópico da questão, coleta-se todas as entidades que estejam conectadas a ela dentro de *h hops* como entidades candidatas a resposta. Considerando-se por exemplo a questão "*Who was the secretary of state of Ohio in 2011*" (veja a Figura 2.21), sua entidade tópico é *Ohio* e pelo subgrafo mostrado na Figura 2.21, partindo-se de *Ohio* e utilizando-se $h = 2$, todos os nós do subgrafo seriam considerados candidatos a resposta por estarem a no máximo 2 relações de distância de *Ohio*;

Figura 2.21 – Subgrafo para entidade *Ohio* em *Freebase*



Fonte: Chen, Wu e Zaki (2019)

– Representação da *KB*: para a representação da *KB* são considerados, para cada nó (entidade) candidato à resposta, três tipos de informações, sendo eles o tipo da entidade, de forma que o modelo seja capaz de assimilar os tipos mais prováveis de estarem corretos para uma determinada palavra interrogativa, o caminho da resposta, que compreende a sequência de relações de uma resposta candidata até a entidade tópico, e o contexto da resposta, correspondente as entidades em torno da entidade candidata a resposta, tais como os nós irmãos, que podem ajudar a responder questões que estão sujeitas a restrições. Considerando-se a resposta candidata "Jon A. Husted"na Figura 2.21, pode-se considerar como seu contexto

os nós "secretary_of_state" e "2011-01-09". Para armazenar as repostas candidatas e sua respectiva informação, é utilizada uma rede *Key-Value Memory* (MILLER et al., 2016) que baseia sua etapa de endereçamento na chave de memória e sua etapa de leitura no valor de memória o que fornece uma maior flexibilidade para codificar conhecimento a priori via separação de funcionalidade;

c) *Reasoning module*: é o módulo responsável por interpretar os dados, a questão e suas respostas candidatas, e refinar a seleção da resposta, constituído pelo módulo de generalização, e a rede bidirecional de duas camadas constituída de duas etapas. A rede primária de atenção, correspondente a primeira etapa, é responsável por focar nas partes importantes de uma questão com base na *KB* e focar nos aspectos importantes da *KB* com relação a questão, e a rede secundária de atenção, correspondente a segunda etapa, pretende aprimorar os vetores da questão e da *KB* explorando mais a atenção de duas vias. Então o módulo de generalização usa a representação da questão $\tilde{q}$ gerada para consultar sobre as chaves de memória via um mecanismo de atenção e buscar as informações mais relevantes do valor de memória, que são usadas para atualizar o vetor da questão via *GRU* e para posterior normalização;

d) *Answer module*: módulo para seleção da resposta. Baseado na representação da questão Q que é $\hat{q}$ e a representação das respostas candidatas $\{\overline{M}_i^k\}_{i=1}^{|A|}$, onde $|A|$ é o número de respostas candidatas e k representa a chave de memória, calcula-se a pontuação de similaridade $S(\hat{q}, \overline{M}_i^k)$ para cada par (questão, resposta-candidata) e então ranqueia as respostas candidatas de acordo com esta pontuação de similaridade.

Para maiores detalhes relacionados à formulação e transformações dos dados ao longo da estrutura vide (CHEN; WU; ZAKI, 2019). O modelo é treinado utilizando-se dois bancos de dados, o *Freebase* como a *Knowledge Base*, que possui 41 milhões de entidades, 19 mil relações e 596 milhões de afirmações (triplas), e o banco de dados *WebQuestions* que contém as questões a serem respondidas, sendo composto por 3022 questões para treino, 756 questões para desenvolvimento e 2032 questões para teste. O banco de dados contendo as questões, *WebQuestions*, possui as seguintes informações para cada questão:

a) Respostas: uma lista com as respostas corretas para cada questão;

b) Entidades: a lista de entidades contidas na questão;

Figura 2.22 – Dicionário para um subgrafo partindo da entidade "augerian_cup"

{"algerian_cup": {"name": ["Algerian Cup"], "alias": [], "notable_types": [], "type": ["/common/topic"], "neighbors": {}, "id": "/m/06md5w"}}

Fonte: Do autor (2021)

c) Questão: a questão em linguagem natural (inglês);

d) ID: identificação da questão, um número inteiro único para cada questão do banco;

e) *FreeBaseKey*: a entidade tópico da questão contida no *Freebase*, ou seja, a entidade tópico ouro da questão;

f) *FreeBaseKeyCands*: uma lista de entidades do *Freebase* candidatas a entidade tópico para a questão.

Já o banco de dados contendo o grafo, o *Freebase*, é composto por vários dicionários no formato mostrado na Figura 2.22, que compõem subgrafos da *KB* onde as chaves são entidades e cada entidade contém um dicionário com as seguintes informações:

a) *name*: o nome da entidade escrito em linguagem natural;

b) *alias*: uma lista com os possíveis "apelidos"da entidade, que pode ser útil para desambiguação;

c) *notable_types*: uma lista contendo os tipos mais relevantes da entidade;

d) *types*: uma lista contendo tipos da entidade;

e) *neighbors*: um dicionário, onde suas chaves são as relações que conectam a entidade1 (pai) as entidades2 (filhas) e a cada chave estão assimilados dicionários das entidades filhas;

f) *id*: um código alfa-numérico para identificação da entidade, que é único para cada entidade do banco de dados.

O código *python* empregado para a execução do *BAMnet* está disponível no *GITHUB* dos autores de seu artigo (CHEN; WU; ZAKI, 2019). São fornecidos os códigos para o treino do modelo *BAMnet* utilizando-se as entidades tópico ouro, para teste do modelo, para o treino do seu preditor de entidade tópico bem como para teste do preditor, um código para teste do

modelo utilizando-se o preditor de entidade tópico, além dos códigos para pré-processamento do banco para o preditor e para o modelo BAMnet.

Com relação aos modelos fim-a-fim até aqui descritos, o modelo possui algumas vantagens como a não necessidade de regras desenvolvidas por humanos para extração de informações das questões, uma menor dependência de recursos externos além da menor necessidade de recursos computacionais, por exemplo consumo de memória. Este modelo não fornece técnicas para extração de triplas das questões, que possam ser empregadas no *MINERVA* por exemplo, com a finalidade de montar um modelo fim-a-fim composto, por tratar todo o processo desde a questão de entrada em linguagem natural até a determinação das respostas no espaço neural.

2.7.2.3 Modelos Fim-a-fim Baseados em *Neural Semantic Parsing*

Os modelos que utilizam métodos de análise semântica neural *NSPs*, assim como os modelos que utilizam o método de recuperação de informação *IRs*, não necessitam do uso de *templates* ou uso de regras manualmente definidas como os modelos de análise semântica *SPs*. No entanto, estes modelos bem como os modelos de análise semântica ainda geram alguma forma de representação da questão em formas lógicas intermediária tais como *SPARQLs*. A seguir, são apresentados respectivamente os modelos *Multihop Complex Questions* (LAN; JIANG, 2020) que as vezes é chamado neste trabalho apenas por *Multihop*, e o modelo *Neural Symbolic Complex Question Answering* (HUA et al., 2020) denominado *NS-CQA*, que se baseiam no método de análise semântica neural.

Multihop

Aqui é descrito o modelo completo desenvolvido no trabalho de Lan e Jiang (2020). O modelo, que é denominado aqui por *Multi-hop Complex Questions* ou apenas *Multihop*, é uma abordagem criada a fim de lidar com questões complexas em linguagem natural. Em se tratando de questões complexas, podem ser abordados dois tipos de complexidade (LAN; JIANG, 2020):

a) Questões de relação única com restrições: um exemplo de questão deste tipo é a questão "*Who was the first president of the U.S.?*" onde a única relação existente entre a entidade-resposta e a entidade "*U.S.*" é "*president_of*", mas existe ainda a condição "*first*" que deve ser satisfeita;

b) Questões com múltiplos passos de relações: um exemplo de questão deste tipo é a questão "*Who is the wife of the founder of Facebook?*", cuja resposta está associada a "*Facebook*" através de dois passos de relações, sendo elas, "*wife_of*" e "*founder_of*".

A abordagem feita por Lan e Jiang (2020) visa lidar com ambos os tipos de complexidade simultaneamente, através de uma abordagem de *Semantic Parsing* combinada com aprendizado por reforço que será brevemente descrita a seguir.

O processo de geração do *query graph*, ou seja, representação da questão em forma de grafo, até a busca da resposta pode ser genericamente descrito em um processo de quatro etapas:

a) Partindo-se de uma entidade básica encontrada na questão, denominada entidade tópico, identificar um caminho relacional central que conecte a entidade tópico a uma variável *lambda*;

b) A partir do caminho relacional central encontrado no passo anterior, adicionar uma ou mais restrições encontradas na questão. Uma restrição pode ser uma entidade básica ou uma função de agregação junto com uma relação;

c) Com todos os *query graphs* candidatos gerados nos dois passos anteriores, ranqueá-los medindo-se suas similaridades com relação à questão;

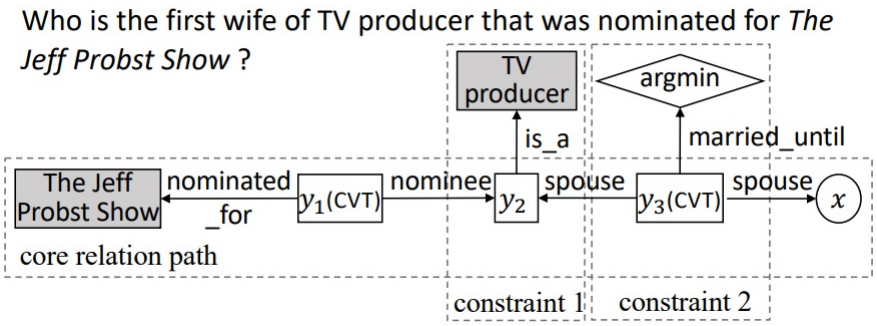
d) Executar os *query graphs* melhor ranqueados contra a *KB* para obter as entidades respostas.

Um exemplo de *query graph* é mostrado na Figura 2.23 para a questão "*Who is the first wife of TV producer that was nominated for The Jeff Probst Show?*" onde pode ser visto o caminho relacional central (*core relation path*), uma restrição com entidade básica (*constraint 1*) e uma restrição com função de agregação (*constraint 2*).

Para geração dos *query graphs* candidatos, o modelo *Multi-hop Complex Questions* usa a técnica *beam search* iterativamente, isto é, mantém-se apenas os *top-K t-hop* caminhos relacionais antes de gerar os caminhos relacionais de tamanho *(t+1)-hop*. Para isso assume-se que a t-ésima iteração produz um conjunto de *K query graphs* denominados como $\mathscr{G}_t$. Então, na iteração $t+1$, para cada $g \in \mathscr{G}_t$, aplica-se uma das seguintes ações para expandir-se g por mais uma aresta e mais um nó:

a) Ação *extend*: esta ação extende o caminho relacional central por mais uma relação em $\mathscr{R}$. Se o *query graph* atual contém apenas uma entidade tópico e a ação *extend* encontra

Figura 2.23 – Exemplo de um *Query Graph* Multi-hop com Restrições



Fonte: Lan e Jiang (2020)

uma relação *r* que conecte a *e* na *KB*, expande o caminho por *r*, onde *r* pode ainda ser duas relações conectadas por um nó especial denominado *CVT* que não representa uma entidade básica, e torna o outro fim de *r* numa variável *lambda x*.

Caso o atual *query graph* já tenha uma variável *lambda*, a ação *extend* muda *x* para uma variável existencial *y*, encontra todas as ligações de *y* na *KB* executando o atual *query graph* contra a *KB*, encontra uma relação *r* conectada a uma destas entidades e então conecta *r* a *y*. O outro fim de *r* se torna a nova variável *lambda x*;

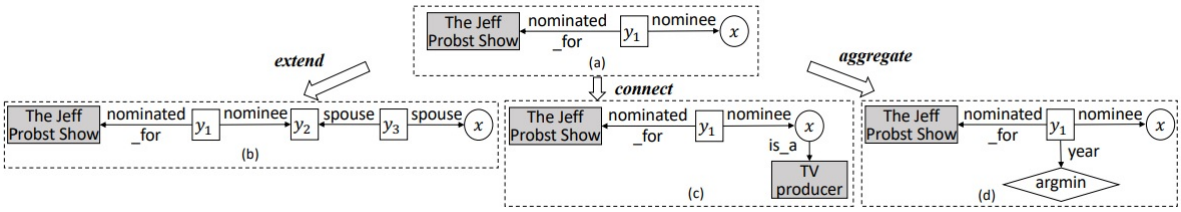
b) Ação *connect*: além da entidade tópico que é colocada no início do atual caminho relacional central, podem haver ainda outras entidades básicas contidas na questão. A ação *connect* conecta uma tal entidade básica *e* a uma variável *lambda x* ou a uma variável existencial conectada a *x* que seja um nó *CVT*. Para decidir qual relação *r* será usada para conectar *e* e *x*, executa-se o atual *query graph* contra a *KB* para determinar todas as conexões de *x* e então encontra-se uma relação que exista entre uma destas entidades e *e*;

c) Ação *aggregate*: para detecção de uma função de agregação na questão é utilizado um conjunto de palavras chaves pré-definidas. Assim, a ação *aggregate* conecta a função de agregação detectada como um novo nó conectado à variável *x* ou a uma variável existencial conectada a *x* que seja um nó *CVT*.

Um exemplo das ações é mostrado na Figura 2.24, onde foi considerada a questão "*Who is the first wife of TV producer that was nominated for The Jeff Probst Show?*" onde em (b) é tomada uma ação *extend* em relação ao *query graph* em (a), em (c) é tomada uma ação *connect* em relação ao *query graph* em (a) e em (d) é tomada uma ação *aggregate* em relação ao *query*

graph em (a), mas considerando-se a questão "*Who is the first person that was nominated for The Jeff Probst Show?*".

Figura 2.24 – Exemplo das Ações que Podem Ser Tomadas para a Geração de um *Query Graph*



Fonte: Lan e Jiang (2020)

A expansão é realizada em todos os *query graphs* $g \in \mathcal{G}_t$ e todas as ações que são aplicáveis são tomadas para cada g. Desta forma, considerando-se $\mathcal{G}'_{t+1}$ como o conjunto de todos os *query graphs* resultantes, utiliza-se uma função de avaliação para ranquear todos os *query graphs* em $\mathcal{G}'_{t+1}$ e então coloca-se os *top-K* deles em $\mathcal{G}_{t+1}$ que sejam melhor avaliados que qualquer grafo $g \in \mathcal{G}_t$.

Para avaliar os *query graphs* no fim da t-ésima iteração, os *query graphs* candidatos em $\mathcal{G}'_t$ são ranqueados derivando-se um vetor de características de 7 dimensões v_g para cada grafo $g \in \mathcal{G}'_t$ e então passa-se estes vetores em uma camada completamente conectada seguida por uma camada *softmax* a fim de gerar $p(g|Q)$.

As 7 características derivadas de cada *query graph* contida em v_g são:

a) A primeira dimensão vem de um modelo *BERT* baseado em correspondência semântica, através da conversão de g em uma sequência de *tokens* seguindo a sequência de ações que foram tomadas para construir g e adicionando-se as descrições textuais das entidades e relações envolvidas em cada passo de forma sequencial, ignorando-se variáveis existenciais e variáveis *lambda*;

b) A segunda dimensão de v_g é a pontuação acumulada de *entity linking* de todas as entidades basicas no *query graph*;

c) A terceira dimensão de v_g é o número de entidades básicas que aparecem no *query graph*;

d) A quarta dimensão de v_g é o número de tipos de entidades no *query graph*;

e) A quinta dimensão de v_g é o número de expressões temporais no *query graph*;

f) A sexta dimensão de v_g é o número de superlativos no *query graph*;

g) A sétima é o número de entidades resposta do *query graph*.

Para treinar o modelo são empregadas as questões pareadas com suas respostas corretas, sem utilizar-se os reais *query graphs* das questões. É utilizada a técnica de aprendizado por reforço descrita em Das et al. (2018) para aprender uma função política de maneira fim-a-fim, de modo a selecionar-se as ações a serem tomadas em cada passo de tempo t para geração dos *query graphs*. Como medida de recompensa é empregado o *F1 score* das respostas preditas com respeito às reais respostas corretas.

Para avaliação do modelo, o *Freebase* é utilizado como a *KB* e são empregados três diferentes bancos de dados: o *Complex Web Questions (CWQ)*, *Complex Questions (CQ)* e o *Web Questions SP (WBQ)*. Dentre estes, o banco com maior número de questões complexas e maior dificuldade é o *CWQ*. De acordo com os resultados obtidos em Lan e Jiang (2020), o modelo atinge o estado da arte para os três bancos de dados em que foi avaliado. Para resultados mais detalhados vide Lan e Jiang (2020).

Apesar disto, o modelo ainda sofre com erros provenientes da geração de *query graphs*, que correspondem a 65% dos erros, devido a falha na predição de relações difíceis de serem corretamente detectadas, devido a fatores como o uso de abreviações (LAN; JIANG, 2020). Falhas provenientes da etapa de detecção de entidades, com a não detecção de algumas entidades contidas nas questões, são responsáveis por 27% dos erros, e 6% dos erros ocorrem pelas limitações do processo empregado para a geração de *query graphs* (LAN; JIANG, 2020).

O *Multihop* mostra um desempenho interessante em relação aos bancos de dados que é empregado como um modelo completo fim-a-fim, particularmente em relação ao *Web Questions SP* com um *F1-score* de 74,0%, e portanto, pretende-se utilizá-lo em paralelo com outros modelos de forma que se complementem, a fim de aprimorar ainda mais seus resultados para a tarefa de *Knowledge Base Question Answering (KBQA)*. Além disto os *query graphs* gerados pelo *Multi-hop Complex Questions* podem ser interessantes para a adaptação de um modelo fim-a-fim composto utilizando-os junto ao *MINERVA*, por exemplo.

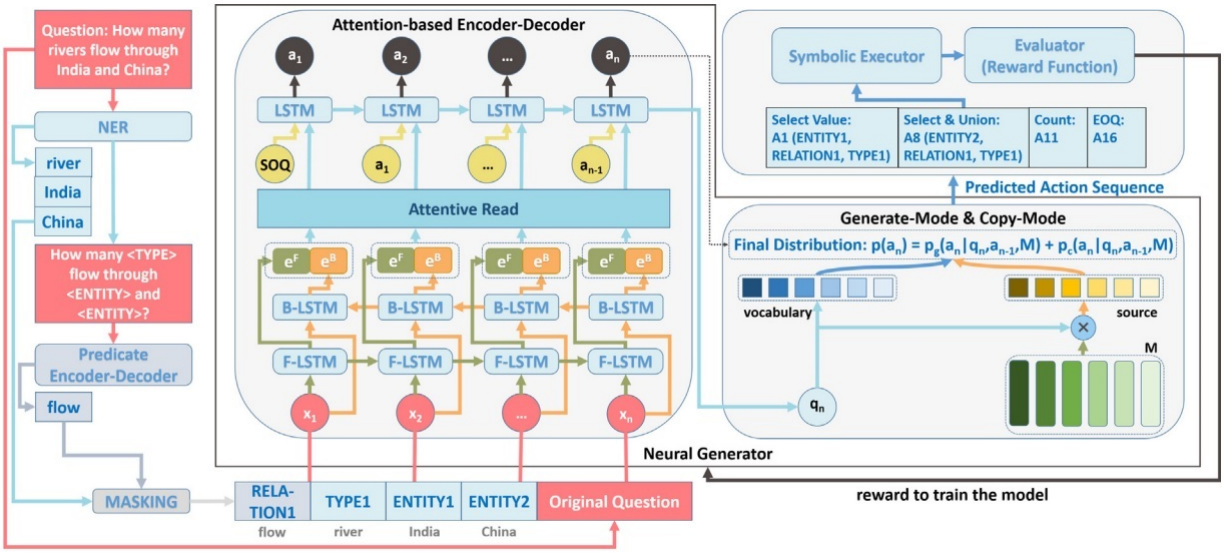
NS-CQA

Outro trabalho que faz uma abordagem interessante a tarefa de *Knowledge Base Question Answering (KB-QA)* com base no método de Análise Semântica Neural *(NSP)* é o modelo

Neural-Symbolic Complex Question Answering (NS-CQA) (HUA et al., 2020). Assim como o *Multihop* (LAN; JIANG, 2020), o *NS-CQA* é um modelo desenvolvido com foco em ser capaz de solucionar questões complexas e, consequentemente, acaba sendo capaz de solucionar questões simples.

De forma simplificada, o *NS-CQA* consiste de dois módulos básicos para lidar com as questões e respondê-las: o Gerador Neural, com a função de transformar a questão em linguagem natural em uma sequência de ações primitivas; e o Executor Simbólico, com a função de executar as ações primitivas geradas para a questão contra a *Knowledge Base* a fim de se determinar a resposta da questão. Na Figura 2.25 é mostrada a arquitetura do *NS-CQA*.

Figura 2.25 – Estrutura geral do modelo *NS-CQA*



Fonte: Hua et al. (2020)

A estrutura do *NS-CQA* exibida na Figura 2.25 pode ser dividida em três grandes estágios de tarefas, onde o primeiro estágio compreende às tarefas que antecedem o bloco do Gerador Neural (*Neural Generator* na Figura 2.25, denominado Analisador Semântico. O segundo estágio é o próprio bloco do Gerador Neural e o terceiro estágio é o Executor Simbólico. Estes estágios são descritos de forma resumida a seguir. Para uma explicação mais completa da estrutura do modelo vide o trabalho de Hua et al. (2020).

a) Analisador Semântico: este estágio é responsável por reconhecer artefatos da *Knowledge Base* que são relevantes para a questão. Dada uma questão em linguagem natural, primeiramente o Analisador Semântico reconhece menções a entidades (por exemplo

"China") e menções a classes de entidades (por exemplo "*river*") utilizando-se de uma *Bidirectional-LSTM-CRF* (YIN et al., 2016) para rotular as menções.

Feito isto, as entidades e tipos são referenciadas com as correspondentes entidades e tipos contidas na *KB* utilizando-se similaridade literal e similaridade semântica. Após isto, as entidades e tipos na questão são substituídos por coringas para gerar padrões, e emprega-se um modelo convolucional *Seq2Seq* (GEHRING et al., 2017) para gerar padrões para as correspondentes relações.

Observação: Os coringas são *tokens* que representarão as relações, entidades e tipos de entidades encontradas na questão de maneira mais abstrata para redução do vocabulário, de forma que os mesmos coringas podem ser usados em diferentes questões para representar diferentes entidades, relações e tipos de entidades.

b) Gerador Neural: este estágio é responsável por gerar o conjunto de ações primitivas, baseado em um modelo de atenção *Seq2Seq* aumentado com mecanismos de cópia e mascaramento, o que permite a redução do tamanho do vocabulário do decodificador e alivia o problema de *Out Of Vocabulary (OOV)*. Desta forma, dada uma questão com *tokens* $(w_1, \ldots, w_m)$ o gerador prediz os *tokens* $(a_1, \ldots, a_m)$. O modelo trabalha com um conjunto de 17 tipos de ações primitivas que podem ser vistas na Tabela 2.2.

Tabela 2.2 – Ações primitivas implementadas pelo *NS-CQA*

ID	Action	Retrieved Key–Value Pairs	Output
A1	$Select(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e : \{e_2\}\}$
A2	$SelectAll(et, r, t)$	$\{e_2 \mid e_1 \in et, e_2 \in t, (e_1, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e_1 : \{e_2\}\}$
A3	$Bool(e)$	$value = 1$ *if* $e \in \mathcal{D}$; *otherwise* $value = 0$	$\mathcal{D} = \{bool : value\}$
A4	$ArgMin$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists (e_1 : \{e_2\}) \in \mathcal{D}, \forall e' : (e' : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A5	$ArgMax$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists (e_1 : \{e_2\}) \in \mathcal{D}, \forall e' : (e' : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A6	$GreaterThan(e)$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists (e_1 : \{e_2\}) \in \mathcal{D}, \exists e'_2 : (e : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A7	$LessThan(e)$	$\{e_1 \mid e_1 \in \mathcal{D}, \exists (e_1 : \{e_2\}) \in \mathcal{D}, \exists e'_2 : (e : \{e'_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq \lvert\{e'_2\}\rvert\}$	$\mathcal{D} = \{e_1 : \{e_2\}\}$
A8	$Inter(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cap \{e : \{e_2\}\}$
A9	$Union(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} \cup \{e : \{e_2\}\}$
A10	$Diff(e, r, t)$	$\{e_2 \mid e_2 \in t, (e, r, e_2) \in \mathcal{K}\}$	$\mathcal{D} = \mathcal{D} - \{e : \{e_2\}\}$
A11	$Count$	$Card(\mathcal{D}) = \lvert\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}\}\rvert$	$\mathcal{D} = \{num : Card(\mathcal{D})\}$
A12	$AtLeast(n)$	$\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \geq n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A13	$AtMost(n)$	$\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert \leq n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A14	$EqualsTo(n)$	$\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}, \lvert\{e_2\}\rvert = n\}$	$\mathcal{D} = \{e : \{e_2\}\}$
A15	$GetKeys$	$\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}\}$	$\mathcal{D} = \{key : \{e\}\}$
A16	$Almost(n)$	$\{e \mid e \in \mathcal{D}, \exists (e : \{e_2\}) \in \mathcal{D}, \lvert\lvert\{e_2\}\rvert - n\rvert \leqslant \alpha\}$ where α is predefined	$\mathcal{D} = \{e : \{e_2\}\}$
A17	EOQ	*End of sequence*	$\mathcal{D}$

Fonte: Hua et al. (2020)

Na Tabela 2.2 a coluna ID lista os identificadores de cada tipo de ação, na coluna *Action* são listadas as estruturas de cada uma das ações e seus nomes, na coluna *Retrieved Key-Value Pairs* são descritas as instruções que devem ser executadas pelo Executor Simbólico para cada ação e na coluna *Output* são mostradas as saídas geradas por cada ação, onde $\mathscr{K}$ representa a *KB*, $e, e_1, e_2, \ldots$ representam entidades, r representa um relação, t representa um tipo e $\mathscr{D}$ representa um dicionário chave-valor que armazena resultados intermediários.

Ao incorporar mecanismos de mascaramento, os nomes de artefatos da *KB* usados para compor ações são substituídos por mascaras tais como *ENTITY1*, *TYPE1* e *PREDICATE1*. Por exemplo, considerando se a questão "*What rivers flow in India but not China?*"seria gerada a sequência de ações "*Select(India, flow, river), Diff(China, flow, River), EOQ*"que após substituição dos reais nomes dos artefatos nas ações por máscaras resultaria na sequência "*Select(ENTITY1, PREDICATE1, TYPE1), Diff(ENTITY2, PREDICATE1, TYPE1), EOQ*". Os mapeamentos entre os nomes reais e as mascaras são gravados no modelo para que depois sejam usados para recuperar os reais nomes dos artefatos da *KB* nas ações quando estiverem sendo executadas.

Ao incorporar o mecanismo de cópia reduz-se o espaço de busca. O mecanismo de cópia replica todos os símbolos mascarados na sequência de entrada para formar a saída ao invés de deixar o decodificador gerá-los a partir do seu vocabulário, e desta forma o decodificador apenas necessita gerar as ações primitivas reduzindo ainda mais seu vocabulário.

O *encoder* é composto pela *Bi-LSTM* contida na entrada do Gerador Neural (vide Figura 2.25) que recebe como entrada os artefatos da *KB* gerados pelo Analisador Semântico concatenados com a questão original e gera um vetor de codificações e_i para cada passo de tempo i. Os vetores $(e_1, \ldots, e_T)$ são vistos como uma memória de curto prazo M, que é salva para uso no modelo *copy-mode* do decodificador.

O *decoder* como tradicionais modelos *Seq2Seq* é uma outra rede *LSTM* que gera um vetor oculto q_t a partir do *token* de saída anterior (a_{t-1}), o estado oculto anterior q_{t-1} e o vetor de contexto c_t. O vetor de contexto c_t é obtido passando-se o vetor de estado do passo anterior q_{t-1} por uma camada de atenção. Os *tokens* de saída a_t são preditos pelo *decoder* seguindo uma probabilidade mista de dois modelos, denominados *generate-mode* e *copy-mode*.

O modelo *generate-mode* calcula *scores* para predição do *token* a_t a partir do vocabulário de saída com base no vetor de estado oculto q_t, onde a_t é um *token* v_i contido no vocabulário de saída. Já o modelo *copy-mode* calcula *scores* para predição do *token* a_t considerando-se como candidatos os *tokens* x_j contidos na cópia feita da entrada $(x_1, \ldots, x_T)$. Por fim, o decodificador calcula uma probabilidade mista com base em q_t e M e seleciona-se a_t final. Os cálculos e equações empregados são descritos em Hua et al. (2020);

c) Executor Simbólico: este é o estágio responsável por executar as ações primitivas geradas pelo Gerador Neural. Primeiro, o Executor Simbólico analisa os *tokens* de saída produzidos pelo Gerador Neural e monta as ações uma a uma. Após isso, dada a sequência de ações gerada, a partir da primeira ação, o Executor Simbólico as executa em ordem, uma a uma, operando sobre os resultados intermediários da ação anterior até que se encontre a ação *EOQ*, onde o resultado do último passo de execução é retornado como a resposta final.

Um exemplo da execução das ações do *NSCQA* pode ser visto na Tabela 2.3 considerando-se as questões "*Which country has maximum number of rivers*"e "*What rivers flow in India but not China?*. Nos exemplos, para cada uma das questões são geradas 3 ações, onde na primeira linha de cada exemplo é mostrada a sequência e ordem das ações tomadas, na segunda linha de cada exemplo são mostradas as informações recuperadas da *KB* para cada ação e na terceira linha é mostrado o resultado da execução da ação, ou seja, o dicionário $\mathscr{D}$ após cada execução.

Tabela 2.3 – Ações primitivas implementadas pelo *NS-CQA*

	Actions	**Action1:** *SelectAll (country, flow, river)*	**Action2:** *ArgMax*	**Action3:** *EOQ*
Question1: Which country has **maximum number** of rivers?	Retrieved Key–Value Pairs	{China:{Indus, Satluj}} {India:{Indus, Satluj, Godavari}} {Russia:{Volga, Moskva, Neva, Ob}} {USA:{Mississippi, Colorado, Rio Grande}}	/	/
	Dictionary	{China:{Indus, Satluj}} {India:{Indus, Satluj, Godavari}} {Russia:{Volga, Moskva, Neva,Ob}} {USA:{Mississippi, Colorado, Rio Grande}}	{Russia:{Volga, Moskva, Neva, Ob}}	{Russia:{Volga, Moskva, Neva, Ob}}
	Actions	**Action1:** *Select (India, flow, river)*	**Action2:** *Diff (China, flow, river)*	**Action3:** *EOQ*
Question2: What rivers flow in India **but not** China?	Retrieved Key–Value Pairs	{India:{Indus, Satluj, Godavari}}	{China:{Indus, Satluj}}	/
	Dictionary	{India:{Indus, Satluj, Godavari}}	{India:{Godavari}}	{India:{Godavari}}

Fonte: Hua et al. (2020)

No exemplo da Tabela 2.3 para a questão "*Which country has maximum number of rivers*"é gerada a sequência de ações "*SelectAll(country, flow, river), ArgMax, EOQ*". A primeira ação se trata de uma *SelectAll*, onde a relação é *flow* e possui duas variáveis de tipo (*country* e *river*), não existindo uma entidade na ação. Executando-se a primeira ação, primeiramente recupera-se todas as entidades pertencentes ao tipo *country* contidas na *KB* como chaves, e então, as entidades de tipo *river* conectadas com as entidades do tipo *country* pela relação *flow* são recuperadas como os valores. Por exemplo, como demonstrado na Tabela 2.3 uma entidade do tipo *country* é *China* onde as entidades do tipo *river* conectadas através da relação *flow* são {*Indus, Satluj*}. Os pares chave-valor recuperados são então armazenados no dicionário $\mathscr{D}$ como o resultado intermediário desta ação. Após isto, é executada a ação *ArgMax*, que não carrega nenhuma entidade ou tipo de entidade. A ação *ArgMax* encontra a chave que possua o maior número de elementos como seus valores. No exemplo disposto na Tabela 2.3, para a primeira questão, a chave *Russia* possui o maior número de valores e, portanto, é mantida em $\mathscr{D}$ enquanto as outras são descartadas como resultado intermediário da segunda ação. Por fim, toma-se a ação *EOQ* que simplesmente indica o fim da sequência de ações e retorna o resultado intermediário da ação anterior contido em $\mathscr{D}$ como o resultado final.

Uma análise semelhante pode ser feita para a segunda questão apresentada na Tabela 2.3 considerando-se a sequência de ações que para ela é apresentada. A forma de execução de cada ação implementada pelo *NS-CQA* é mostrada na Tabela 2.2.

O treino do modelo utiliza a técnica de aprendizado por reforço, onde o estado, ação e recompensa no tempo t são respectivamente s_t, a_t e r_t. O estado no tempo t, dada uma questão q, é definido pela própria questão q e a sequência de ações geradas até aqui: $s_t = (q, a_{0:t-1},$ e os *tokens* das ações são gerados pelo Gerador Neural. No último passo de decodificação T, a sequência inteira de ações é gerada e então o Executor Simbólico a executa para produzir a resposta de saída ans_0. Portanto, a recompensa só é calculada após o último passo de decodificação, quando ans_0 é a resposta. Para o cálculo da recompensa é utilizada uma Função de Recompensa Adaptativa (*ARF*) que é uma comparação adaptativa entre a resposta ans_0 gerada pelo modelo e a resposta *gold* ans_g. Além disso, aplica-se uma Recompensa Bônus guiada por currículo *(CRB)* que considera a proximidade da sequência de ações geradas com sequências de ações que geraram melhores resultados, e a novidade de sequências novas de ações geradas que não se assemelham a outras geradas para atribuir recompensas não-zero para casos que não

fornecem respostas corretas. A descrição completa do treino, com maiores detalhes pode ser encontrada no trabalho de Hua et al. (2020).

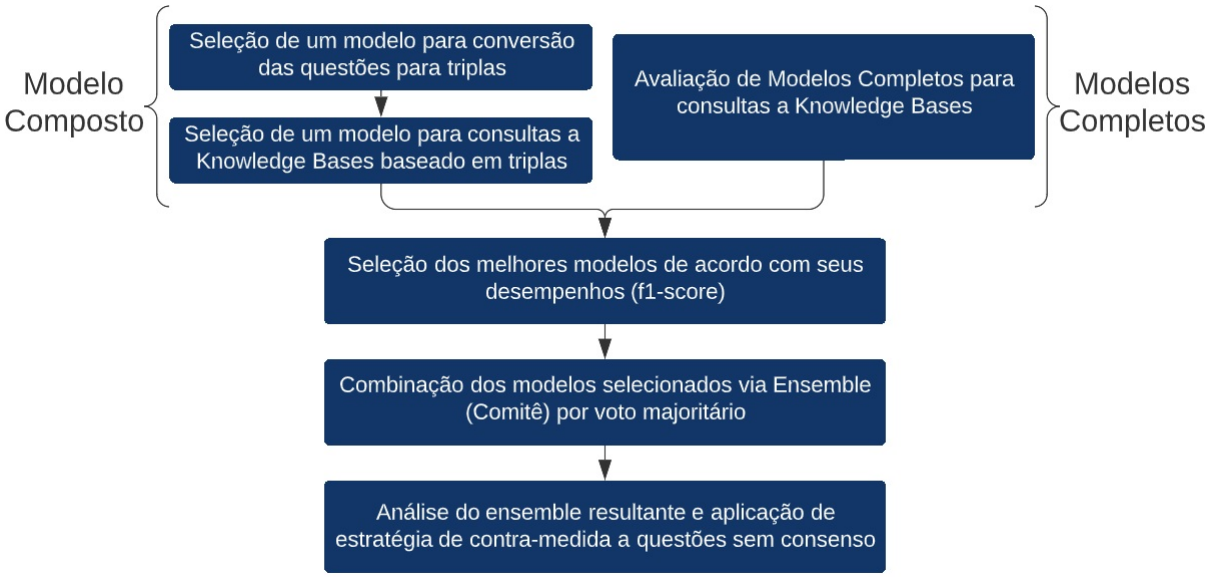
Algumas das fontes de erros observadas no modelo são a correspondência entre entidades e tipos de entidades encontrados na questão com as entidades/tipos contidas na *KB*, a geração de ações semanticamente incorretas ou sem significado, o treino pode ser guiado por amostras pobres devido a forma que é implementado, os argumentos das ações podem ser preditos em ordens incorretas e podem ocorrer erros ao considerar-se aproximações em questões quantitativas ou de comparação.

O *NS-CQA* é avaliado nos bancos de dados de questões *Complex Question Answering (CQA)* que é um banco de larga-escala, e o *WebQuestionsSP*, que é o banco de referência desta pesquisa, onde a *Knowledge Base* empregada para consulta das respostas para ambos os bancos de dados é a *Freebase*. No *WebQuestionsSP* o *NS-CQA* atinge um *F1-score* de 72,04%, que é um valor muito bom e competitivo para tal banco de dados considerando-se que o maior valor de *F1-score* encontrado na literatura até aqui é 74,00%, obtido pelo *Multihop* (LAN; JIANG, 2020).

3 MATERIAIS E MÉTODOS

Para o desenvolvimento de um assistente virtual é necessário o desenvolvimento de 4 módulos como mostrado na Seção 2.1. Neste capítulo, é apresentado e descrito o processo de avaliação de várias ferramentas, sejam elas modelos compostos 3.1 ou completos 3.2, que possam compor o módulo *soft-KB lookup*. Após avaliá-las, são utilizadas técnicas de combinação via *ensemble* de algumas delas para que se possa obter um modelo mais robusto para a tarefa de *Knowledge Base Question Answering (KB-QA)* que possa servir como o modelo final para o módulo *soft-KB lookup*. Desta forma, a metodologia empregada será de acordo com o fluxograma disposto na Figura 3.1.

Figura 3.1 – Fluxograma de desenvolvimento do Ensemble



Fonte: Do Autor (2022)

O capítulo foi dividido em quatro seções, de maneira que a primeira seção contém os detalhes da implementação de um modelo fim-a-fim composto próprio voltado para *KB-QA*, a segunda lista modelos fim-a-fim completos de *KB-QA*, na terceira é mostrada uma estratégia de combinação dos modelos *KB-QA* para desenvolvimento de um modelo final que no futuro poderá ser empregado como o módulo *soft-KB lookup* do assistente virtual, e na quarta seção são mostradas as métricas utilizadas nas análises de desempenhos.

3.1 Modelo Fim-a-fim Composto

Estes modelos não foram desenvolvidos a princípio pra lidar com questões em linguagem natural, desta forma, é preciso empregar alguma técnica para extração de informações das

questões em linguagem natural para convertê-las em triplas. Então para que se comportem como um modelo de assistente virtual fim-a-fim, são compostos por conversor de linguagem natural para triplas mais modelo de consulta a *KBs* baseado em triplas, que são descritos nas Subseções 3.1.2 e 3.1.3. Na Subseção 3.1.1 são mostrados os bancos de dados utilizados nas avaliações do modelo.

3.1.1 Bancos de Dados

Nesta seção é realizada uma breve descrição dos bancos de dados empregados para avaliação do modelo composto gerado. O modelo é avaliado de duas formas independentes: primeiramente o modelo para consulta a *Knowledge Bases* baseado em triplas é avaliado individualmente, a fim de verificar suas capacidades e limitações além de confirmar os resultados dispostos em seu artigo, e por segundo, o modelo fim-a-fim composto é avaliado em um banco de dados *KB-QA*.

Os bancos de dados empregados para avaliação do modelo para consulta a *Knowledge Bases* baseado em triplas são de áreas variadas, conhecidos e empregados recentemente na literatura visando determinar o potencial do modelo para diferentes estruturas das *"Knowledge Bases"*, sendo eles:

a) *COUNTRIES* (BOUCHARD; SINGH; TROUILLON, 2015): as *queries* são todas do tipo *Located_in*. As entidades são países, continentes e sub-regiões e seu grafo é composto por apenas duas relações (*Located_in e Neighbor_of*);

b) *Unified Medical Language System (UMLS)* (KOK; DOMINGOS, 2007): as entidades neste banco são conceitos médicos (por exemplo, antibiótico, doença) e as relações são predicados, dentre os quais estão trata, diagnostica e etc.;

c) *Alyawarra Kinship* (KOK; DOMINGOS, 2007): os fatos contidos nessa base de dados são referentes às relações de parentesco entre membros da tribo *Alyawarra* da Austrália Central. As entidades são pessoas da tribo;

d) *WN18RR* (DETTMERS et al., 2018): é um subconjunto do *Wordnet*, e assim as entidades que contém são de áreas bem variadas. As relações definem relações linguísticas entre as entidades, por exemplo, hiperônimo, forma derivacional relacionada, membro merônimo e etc;

e) *FB15k-237* (TOUTANOVA et al., 2015): baseada no *Freebase* que é um grande banco de dados de fatos em geral, contendo por exemplo, nacionalidade de pessoas, recebimento de prêmios, linguagem falada em países e etc., de onde removeu-se triplas inversas e selecionou-se apenas parte das triplas;

f) *Never-Ending Language Learner (Nell-995)* (XIONG; HOANG; WANG, 2017): são extratos de fatos a partir de textos encontrados em páginas da web. Em geral as entidades estão relacionadas com o meio esportivo, tais como nomes de estádios, de jogadores, etc., e as relações relacionam atletas a equipes, estádios, localizações de nascimento, dentre outros.

Informações mais completas sobre os bancos de dados anteriormente citados podem ser vistas na tabela 3.1, que mostra as quantidades de entidades, de tipos de relações, de triplas que compõem os grafos (fatos), e de questões (*queries*).

Tabela 3.1 – Informações dos bancos de dados empregados para avaliação do modelo para consulta a *Knowledge Bases* baseado em triplas

Knowledge Base	#entidades	#relações	#fatos	#*queries*
COUNTRIES	272	2	1158	24
UMLS	135	49	5216	661
KINSHIP	104	26	10686	1074
WN18RR	40945	11	86835	3134
NELL-995	75492	200	154213	3992
FB15K-237	14505	237	272115	20466

Fonte: Do Autor (2022)

Para avaliação do modelo composto é empregado o banco de dados *WikiMovies*, um banco voltado para a tarefa *Knowledge Base Question Answering (KB-QA)*, que é constituído de *Knowledge Base (KB)* própria e questões em linguagem natural geradas por *templates* criados por anotadores humanos que podem ser respondidas com base nas informações contidas na sua *KB*. As características do banco de dados são mostradas na Tabela 3.2.

Tabela 3.2 – Características do Banco de Dados *WikiMovies*

Dataset	#Entidades	#Relações	#Fatos	#Questões Empregadas
WikiMovies	43230	9	196453	10000

Fonte: Do Autor (2022)

O banco de dados *WikiMovies* contém questões relacionadas a informações de filmes, como linguagens, atores, diretores, e etc.. Um exemplo de questão contida é "*Which is a*

film written by Herb Freed?". Neste sentido, as entidades contidas na *KB* são atores, escritores, filmes, datas de lançamentos, linguagens e etc. As relações referem-se às propriedades dos filmes, sendo elas: *directed_by, starred_actors, release_year, in_language, has_genre, has_imdb_rating, has_imdb_votes, has_tags*. As questões empregadas no experimento (valor descrito na Tabela 3.2) são o conjunto de treino das questões de WikiMovies.

3.1.2 Conversor de linguagem natural para triplas (Parot)

Para a conversão das questões de linguagem natural para triplas, diversas abordagens podem ser tomadas, seja desenvolvendo-se modelos próprios para resolver a tarefa com o auxílio de ferramentas como a SENNA (COLLOBERT et al., 2011) ou mesmo usando modelos para conversão de triplas contidos em modelos fim-a-fim completos, que são em geral, técnicas da etapa de *query understanding* de modelos fim-a-fim (SP), tais como Parot (OCHIENG, 2020) ou GAnswer (HU et al., 2017).

Para cumprir esta tarefa, optou-se pela implementação parcial do *Parot* devido à sua capacidade de lidar com a grande maioria dos tipos de questões tais como questões contendo sentenças compostas, negações, adjetivos escalares e listas numeradas, e devido à sua relativa facilidade de reimplementação por ser um modelo que emprega várias regras heurísticas baseadas em árvores de dependências para converter as questões em linguagem natural para *SPARQLs*, de onde podem ser obtidas as triplas.

A grosso modo, uma *SPARQL* pode ser vista como um conjunto composto de uma ou mais triplas da forma (e_1, r, e_2), onde r é alguma relação contida entre e_1 e e_2 na questão e e_1 e e_2 podem ser entidades contidas na questão ou entidades-variáveis a serem determinadas na *Knowledge Base*. Para lidar com o *MINERVA*, a representação da questão no padrão *SPARQL* é desnecessária, pois neste caso a única informação útil são as triplas contidas dentro da cláusula *Where{}* (vide um exemplo de uma *SPARQL* na Figura 2.16), e desta forma, para o modelo composto implementado, as triplas geradas para a clausula *Where{}* pelo *Parot* para as questões são empregadas como as *queries* para consultar à *KB* através do *MINERVA*. A estrutura da geração de tais *SPARQLs* pelo *Parot* pode ser dividida em quatro partes, quais sejam, identificação dos alvos da questão, identificação de triplas relacionais e não relacionais, identificação de triplas formadas por adjetivos e o *lexicon*. Para mais detalhes vide Sub-subseção 2.7.1.2.

Pensando no modelo composto, a etapa de identificação dos alvos da questão pode ser descartada pois se está interessado apenas no conteúdo do interior da *SPARQL*, ou seja, suas

triplas, que serão utilizadas pelo modelo para consulta à *KB* para obtenção da resposta da questão. A etapa de identificação das triplas relacionais e não relacionais, descrita na Sub-subseção 2.7.1.2, é implementada da seguinte forma:

a) Consultas (sentenças) baseadas em frases relacionais: na reimplementação realizada, segue-se o padrão *pos-tagging* mostrado na Figura 3.2 para determinação da relação r e então, determina-se as entidades e_1 e e_2 respectivamente como a entidade mais próxima de r na questão pela esquerda e a entidade mais próxima de r na questão pela direita. Se uma entidade real for encontrada, ela é considerada entidade na tripla gerada, caso contrário, é considerada como variável. Visando o modelo que foi selecionado para consulta à *KB* (*MINERVA*), é realizada uma adaptação nas triplas geradas nos casos onde e_2 é uma entidade real e e_1 é uma variável, invertendo-se as posições de e_1 e e_2 na tripla e modificando-se a relação r com um sinal de inversão.

   O *Parot* gera uma tripla extra que funciona como uma restrição de tipo para as variáveis, mas como o *MINERVA* não permite mais de uma tripla por questão, desconsidera-se tais triplas de restrição. Além disso, o *Parot* considera a tripla em ambos os sentidos (e_1, r, e_2) e (e_2, r, e_1), mas aqui, considera-se para o modelo de assistente composto apenas o sentido conveniente ao *MINERVA*. Como um exemplo, considerando-se a questão "*Which rivers flow through Alaska*", a tripla gerada a ser fornecida ao *MINERVA* é (*Alaska,flows_through*$^{-1}$*,?rivers*).

   A regra descrita na Figura 3.2 só é válida no caso onde a relação ou frase relacional encontra-se entre duas entidades. São implementadas regras especiais para o caso de questões iniciadas com *Who* e para o caso de questões onde a relação não se localiza entre duas entidades, de acordo com as seguintes equações:

$$\forall w_e, w_h, w_i.(nsubj(w_e, w_h) \wedge dobj(w_e, w_i)) \Rightarrow Triple(w_i, w_e, ?w_h) \tag{3.1}$$

$$\begin{aligned} \forall w_e, w_h, w_i, w_j.(pobj(w_e, w_h) \wedge prep(w_j, w_e) \wedge nsubj(w_j, w_i)) \\ \Rightarrow Triple(w_i, w_j, ?w_h) \end{aligned} \tag{3.2}$$

   Como exemplo, a questão "*Who killed Ceasar*" se adequa a regra descrita na Equação (3.1) e assim a tripla extraída gerada é (*Ceasar,killed,?who*). Para a questão "*In which country does the nile traverse?*" não há entidades procedendo o verbo *traverse*

e desta forma aplica-se a regra descrita na Equação (3.2) e então, extrai-se a tripla (*Nile,traverse,?continent*). O processo exato implementado no *Parot* sem as adaptações para determinar-se a relação ou frase relacional *r* e formação da tripla contida na questão é descrito em Fader, Soderland e Etzioni (2011) e Ochieng (2020);

Figura 3.2 – Padrão *Pos-Tagging* para Obtenção das Relações ou Frases Relacionais

$V \mid VP \mid VW^*P$
$V =$ verb particle? adv?
$W =$ (noun | adj | adv | pron | det)
$P =$ (prep | particle | inf. marker)

Fonte: Fader, Soderland e Etzioni (2011)

b) Consultas (sentenças) baseadas em frases não relacionais: na reimplementação são empregadas as regras descritas nas Equações (3.3) e (3.4) para obtenção de triplas não relacionais que contenham as preposições *of* ou *in*, respectivamente:

$$\forall w_u, w_x, w_y, w_z.(nsubj(w_u, w_x) \wedge prep(w_x, w_y) \wedge pobj(w_y, w_z) \wedge (w_y = "of")) \Rightarrow Triple(w_z, w_x\_w_y, ?k), \tag{3.3}$$

$$\forall w_u, w_x, w_y, w_z.(nsubj(w_u, w_x) \wedge prep(w_x, w_y) \wedge pobj(w_y, w_z) \wedge (w_y = "in")) \Rightarrow Triple(w_z, has\_w_x, ?k). \tag{3.4}$$

A tripla retornada para a regra descrita na Equação (3.4) é uma adaptação completamente voltada ao emprego do *MINERVA* uma vez que a tripla que seria gerada pelo *Parot* neste caso é muito dependente do *Lexicon*. Para exemplificar a extração de triplas através das regras descritas pelas Equações (3.3) e (3.4), aplicando-se a primeira à questão "*What is the population of Texas*" e a segunda à questão "*Which is the highest mountain in Germany*" então as triplas retornadas são respectivamente (*Texas,population_of,?k*) e (*Germany,has_mountain,?k*).

Para extrair triplas de questões não relacionais genitivas, o *Parot* aplica a regra descrita por:

$$\forall w_u, w_x, w_y, w_z.(nsubj(w_u, w_x) \wedge poss(w_x, w_y) \wedge possessive(w_y, w_z)) \Rightarrow Triple(w_y, has\_w_x, ?k). \tag{3.5}$$

Para o caso genitivo, novamente a tripla retornada é uma adaptação voltada para a integração com o *MINERVA* uma vez que a tripla original gerada pelo *Parot* não possui a relação definida, e demandava que o *Lexicon* fosse empregado para sua definição. Aplicando-se a regra contida na Equação (3.5) para a questão "*What is Angela's birth name?*", por exemplo, obtém-se a tripla (*Angela,has_name,?k*);

c) **Observação**: as questões, relacionais ou não relacionais, podem ainda ser compostas e, caso sejam, o *Parot* possui regras específicas para quebrá-las em questões simples, e então, aplica as regras descritas para extração de triplas das questões simples geradas. As questões compostas, portanto, necessitam da geração de mais de uma tripla para suas representações, de maneira que o *MINERVA* é incapaz de respondê-las em sua implementação atual. Com isto em mente, questões compostas são ignoradas na implementação atual do *Parot* para composição de um assistente fim-a-fim com o *MINERVA*. Foram reimplementadas regras voltadas para questões-*Wh*, ou seja, questões iniciadas com *what, when, where, who, whom, which, whose, why, and how*. O *Parot* estabelece ainda variantes das regras para questões-*non-Wh*, ou seja, questões que não inciam com palavras *Wh*.

São geradas ainda algumas triplas adicionais na etapa de identificação de triplas formadas por adjetivos. É realizada a seguinte reimplementação desta etapa baseado no *Parot* considerando-se duas classes de adjetivos:

a) Adjetivos escalares: são reimplementadas as seguintes regras de extração de triplas contendo adjetivos escalares:

$$\forall w_u, w_x, w_y.(amod(w_x, w_u) \wedge (w_u \equiv JJS) \Rightarrow Triple(w_x, root(w_u), ?k), \tag{3.6}$$

$$\forall w_u, w_x, w_y.(advmod(w_x, w_u) \wedge amod(w_y, w_x) \Rightarrow Triple(w_y, root(w_x), ?k). \tag{3.7}$$

Para exemplificar a aplicação das regras contidas nas equações (3.6) e (3.7), considerando-se respectivamente as questões "*Which is the longest river in America?*" e "*Which is the most populated state in America?*", que se adequam respectivamente às equações, obtém-se através da aplicação das regras a tripla complementar (*river,root(longest),?k*) para a primeira questão, e a tripla complementar (*state,root(populated),?k*) para a segunda. Como se tratam de triplas complementares

e portanto a questão terá mais de uma tripla, em geral estas questões não se adéquam ao *MINERVA*, mas extrair tais triplas permite identificar tais tipos de questões.

b) Adjetivos não-escalares: são reimplementadas duas regras para extração de triplas deste caso, sendo elas:

$$\begin{aligned} &\forall w_u, w_x, w_y.(amod(w_x, w_u) \wedge amod(w_x, w_y)) \\ &\Rightarrow (Triple(w_x, ?k, w_u) \wedge Triple(w_x, ?k, w_y)) \end{aligned} \tag{3.8}$$

e

$$\forall w_u, w_x.(amod(w_x, w_u)) \Rightarrow Triple(w_x, ?k, w_u) \tag{3.9}$$

Para exemplificar a extração de triplas adjetivas não escalares utilizando-se as regras descritas pelas Equações (3.8) e (3.9), considerando-se a questão "*Which female German chemist won the Nobel prize?*" para a primeira equação e a questão "*Which German chemist won the Nobel prize?*" para a segunda, as triplas adjetivas retornadas são respectivamente (*Chemist,?k,German*)∧(*Chemist,?k,female*) e (*Chemist,?k,German*). Se tratando de questões com adjetivos não escalares, elas sempre necessitam de mais de uma tripla para representação de sua informação, assim como para adjetivos escalares, de modo que o *MINERVA* não é capaz de lidar na atual abordagem, e portanto, tais triplas geradas não foram adaptadas para o *MINERVA*. Desta forma, são utilizadas apenas como uma estatística a respeito do banco de dados.

Por fim, a última etapa neste processo no *Parot* é o *Lexicon*. Na reimplementação realizada, considerando-se as características do *MINERVA* para a consulta à *KB* e as adaptações empregadas nas etapas de extração de triplas visando o modelo composto *Parot-MINERVA*, esta etapa pode ser desconsiderada contanto que se aborde alguma forma diferente para correspondência entre entidades encontradas na questão e entidades existentes na *KB*. Considerando-se o banco *WikiMovies* a correspondência entre entidades contidas na sua *KB* e entidades contidas em suas questões pode ser feita simplesmente empregando-se *string matching*.

Com as etapas de implementação descritas até aqui, o modelo é capaz de extrair triplas para a grande maioria das questões que o *Parot* cobre que são possíveis de serem respondidas pelo *MINERVA*. São consideradas apenas questões simples, portanto se alguma tripla é extraída da questão em alguma das etapas, o processo de extração de triplas para a dada questão é encerrado. A prioridade de extração de triplas é Triplas Relacionais > Triplas Não Relacionais

> Triplas Adjetivas. Os resultados da aplicação da reimplementação do *Parot* às questões do banco de dados *WikiMovies* são apresentados na Seção 4.2.

3.1.3 Modelo para consulta a KBs baseado em triplas (MINERVA)

Após o levantamento de informações em relação a modelos para consultas à *Knowledge Bases* baseados em *queries*, considerando-se as diferentes abordagens contidas no trabalho de Gao, Galley e Li (2018), selecionou-se como modelo para consultas à *KBs* para o modelo fim-a-fim composto o *MINERVA* (DAS et al., 2018), por sua capacidade de lidar com questões *multi-hop* e pelo modelo não exigir que as relações contidas nas *queries* tenham correspondência direta com as relações contidas na *KB*, uma vez que o *MINERVA* é capaz de inferir as correspondências implicitamente.

O *MINERVA* é capaz de realizar consultas à KB através de triplas no formato $(e_1, r, ?)$ onde e_1 é uma entidade contida na questão , denominada entidade tópico, da qual o modelo partirá para busca da resposta na KB, r é uma relação que expressa a informação solicitada sobre a entidade e_1 e ? é a entidade a ser determinada por meio de e_1 e r informados ao *MINERVA*.

Para determinação da resposta à questão (entidade ?), o MINERVA aprende a se deslocar sobre o grafo de conhecimento (*KB*) partindo-se da entidade tópico e_1, então o modelo pode dar h passos neste grafo, onde h é um número variável de acordo com a necessidade que o modelo houver para cada *query*, sendo tais passos direcionados pela relação r, até que se determine a resposta correspondente à *query* fornecida.

Para avaliar as capacidades do *MINERVA* e validar os resultados que são dispostos no artigo que propõe o algoritmo (DAS et al., 2018), o modelo é retreinado nos bancos de dados dispostos na Tabela 3.1, empregando-se no treino os hiper-parâmetros conforme descritos no artigo para cada banco de dados. Os resultados desta análise e suas discussões são dispostos na Seção 4.1.

Para avaliação do modelo composto *Parot-MINERVA* é empregado o banco de dados *WikiMovies*. Após aplicado o modelo *Parot* para obtenção das triplas das questões, é feita uma adaptação no banco de dados para viabilizar o treino e teste do *MINERVA* no banco de dados *WikiMovies*, de forma que para cada questão que haja uma *query* extraída pelo *Parot*, sejam geradas triplas para cada resposta correta pertinente a questão. As triplas geradas são embaralhadas e divididas em 80% para treino, 10% para validação e 10% para teste. Então, o *MINERVA* é treinado e testado nestes conjuntos, empregando-se os hiper-parâmetros ótimos

dispostos em seu artigo (DAS et al., 2018). Os resultados da avaliação do modelo composto *Parot-MINERVA* estão dispostos na Seção 4.2

3.2 Modelos Fim-a-fim Completos

Nesta seção serão apresentados modelos fim-a-fim completos, ou seja, modelos que por si só são capazes de funcionar como o módulo *soft-KB lookup* do assistente virtual. Na Subseção 3.2.1 é apresentado o banco de dados de referência para os modelos individuais e para o *Ensemble* descrito na Seção 3.3, e na Subseção 3.2.2 são nomeados os modelos completos candidatos.

3.2.1 Banco de Dados

O banco de dados de questões tomado como referência e empregado para avaliação dos modelos individuais e do *Ensemble* é o banco *WebQuestionsSP*, que é uma variação do banco de dados de questões *WebQuestions* de onde foram removidas algumas questões que não poderiam ser respondidas através de *SPARQLs* (YIH et al., 2015). Além disso, são atribuídos para cada questão um padrão *SPARQL* que pode ser usado como referência em modelos que utilizam-se de *SPARQLs* para consulta à *Knowledge Base* além de outras informações de análise semântica das questões (*Semantic Parsing (SP)*).

O *WebQuestionsSP* contém uma coleção de questões *multi-hop*, que são questões que necessitam de uma cadeia de triplas da *KB* para serem respondidas, sobre assuntos gerais, cujas respostas podem ser uma entidade ou um conjunto de entidades. Um análise das complexidades das questões é mostrada na Tabela 3.3. O banco possui 4737 pares de questões-respostas que são divididos em um conjunto de treino com 3097 questões, de onde em geral são reservadas uma parte das questões para avaliação dos modelos durante treino, e um conjunto de teste com 1639 questões (HUA et al., 2020).

Tabela 3.3 – Estatísticas de complexidade das questões contidas no *WebQuestionsSP*

Tipo de Questão	Percentual das Questões
1-hop s/ restrição	71,3%
1-hop c/ restrição	28,2%
2-hop s/ restrição	0,0%
2-hop c/ restrição	0,5%

Fonte: Adaptado de Lan e Jiang (2020)

As questões contidas no *WebQuestionsSP* são desenvolvidas para que possam ser respondidas com base em informações contidas na *Knowledge Base* denominada *Freebase* criada pelo *Google* e que contém cerca de 2,6 bilhões de triplas sobre aproximadamente 44 milhões de tópicos (YIH et al., 2015). Devido ao tamanho desta *KB*, em geral, o que é feito pelos modelos é uma extração de subgrafos que estejam em uma determinada vizinhança das entidades tópicos determinadas pelos modelos para as questões e então realiza-se a consulta nestes subgrafos.

3.2.2 Ferramentas

Os modelos completos são modelos de assistentes fim-a-fim que foram desenvolvidos desde o início por seus autores levando-se em consideração a questão em linguagem natural. Assim, tais modelos recebem as questões em linguagem natural e possuem seus próprios meios de extrair as informações nela contida, e com essas informações extraídas consultam à *KB* na tentativa de determinar a resposta adequada. Neste sentido, após uma revisão bibliográfica de tais modelos, a fim de verificar-se os resultados dispostos nos artigos e combiná-los em um *Ensemble*, analisa-se os seguintes modelos com base nas métricas dispostas na Subseção 3.4.2:

a) Modelos baseados em análise semântica tais como *GAnswer* (HU et al., 2017) e *Parot* (OCHIENG, 2020);

b) Modelos baseados em recuperação de informação tais como o *BAMnet* (CHEN; WU; ZAKI, 2019) e o *Neural State Machines teacher-student* (HE et al., 2021).

c) Modelos especiais, baseados em aprendizado por reforço, tais como o *STAGG* (YIH et al., 2015), o *Multihop* (LAN; JIANG, 2020) e o *NS-CQA* (HUA et al., 2020)

3.3 *Ensemble* (Comitê por Voto Majoritário)

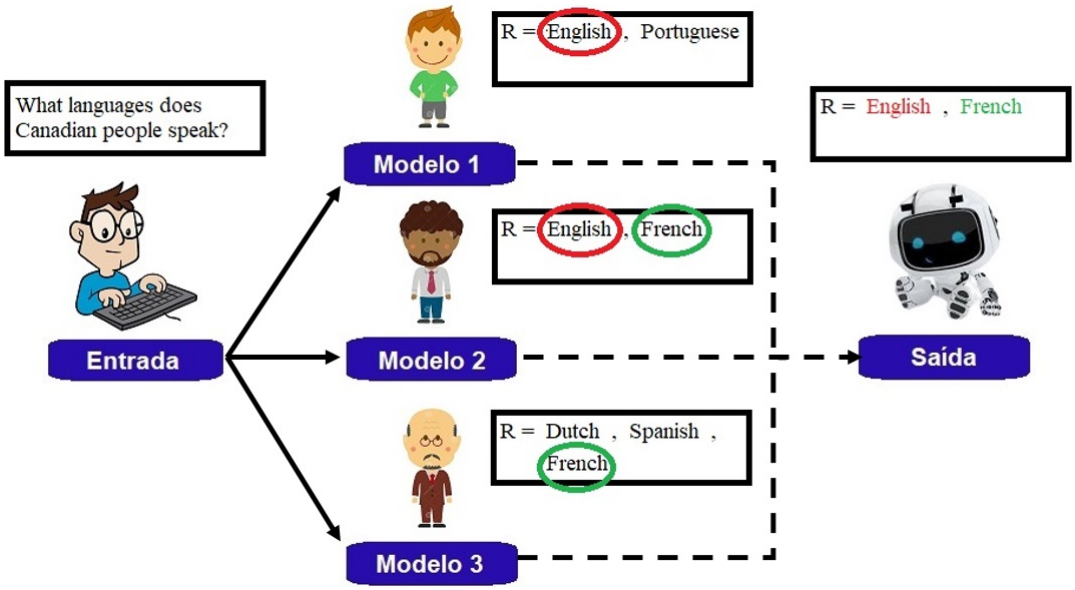
Diversas técnicas podem ser empregadas para combinação e aperfeiçoamento de modelos envolvidos nas áreas de aprendizado de máquina, e dentre essas técnicas, os *Ensembles* são considerados o maior avanço da área de *machine learning* devido às vantagens oferecidas em diversos tipos de situações, desde que os modelos independentes sejam suficientemente acurados e demonstrem diversidade (BAEZA-YATES; RIBEIRO-NETO, 2011).

Para desenvolvimento de um modelo aprimorado que possa ser aplicado como o módulo *soft-KB lookup* do Assistente Virtual, é proposta a aplicação da técnica de *Ensemble* por voto

majoritário neste trabalho. Desta forma após a análise do modelo composto desenvolvido neste trabalho e as análises dos modelos completos encontrados na literatura com reimplementação viável, são selecionadas distintas combinações de 3 modelos para composição de diferentes *Ensembles*, a fim de se avaliar qual combinação garante o melhor desempenho de acordo com as métricas descritas na Subseção 3.4.2, com relação ao banco de dados *WebQuestionsSP* descrito na Subseção 3.2.1.

Uma ilustração da estrutura do *Ensemble* por voto majoritário desenvolvido pode ser vista na Figura 3.3. Previamente, os modelos são treinados individualmente para que possam responder as questões contidas no banco de dados *WebQuestionsSP* (Subseção 3.2.1). Desta forma, cada modelo individual pode ser visto como uma das pessoas a responder a questão na Figura 3.3. O funcionamento do *Ensemble* é dado da seguinte forma:

Figura 3.3 – Estrutura do Ensemble (Exemplo de Funcionamento para Uma Questão)



Fonte: Do Autor (2022)

a) 1º passo: o usuário fornece uma questão ao sistema;

b) 2º passo: a questão é respondida pelos modelos individuais separadamente;

c) 3º passo: são verificados quais das respostas são comuns entre a maioria dos modelos, neste caso, entre 2 ou mais. Na implementação é necessário verificar a existência de consenso para cada resposta distinta gerada pelos modelos, e como algumas métricas podem ser influenciadas pela ordem das respostas (vide Subseção 3.4.2), a ordem de

prioridades das votações das respostas é dada em ordem decrescente de acordo com os *F1-scores* dos modelos individuais;

d) 4º passo: as respostas com consenso são então selecionadas como o conjunto de respostas final.

O modelo descrito até aqui é denominado neste trabalho como *Ensemble* simples ou simplesmente *Ensemble*. Esta abordagem, no entanto, permite ainda que haja muitas lacunas de questões não respondidas pelo modelo, uma vez que podem haver questões em que não haja consenso de respostas entre os modelos individuais. Para lidar com isto, faz-se duas abordagens para reduzir o número de questões sem respostas pelo modelo:

a) Contramedida simples: nesta abordagem é feita uma reavaliação dos modelos individuais para as questões em que não há consenso e então, seleciona-se o conjunto de respostas do modelo com a maior *F1-score* nestes casos como o conjunto de respostas finais destas questões;

b) Contramedida total: no caso de contramedida simples, o modelo selecionado como contramedida para o caso de não consenso das respostas pode também ser incapaz de responder a algumas questões, fornecendo resposta vazia. A abordagem de contramedida total visa minimizar ao máximo a existência de respostas vazias, desta forma, de acordo com a reavaliação dos modelos individuais para as questões sem consenso de respostas, é criada uma lista de prioridades dos modelos para responder as questões sem consenso com base neste *F1-score*. Assim, quando não há consenso entre os modelos, primeiramente o modelo individual com maior *F1-score* nestes casos é selecionado para responder as questões, caso ele não seja capaz, seleciona-se o segundo melhor modelo com base no F1-score nestes casos e assim sucessivamente até que não haja opções de modelos que possam responder as questões dentre os modelos que compõem o *Ensemble*.

Os resultados do melhor *Ensemble* com contramedida simples, também denominado como *Ensemble* com contramedida, e do melhor *Ensemble* com contramedida total são apresentados na Seção 4.3.

3.4 Ferramentas de Análise Estatística

Para analisar os desempenhos dos modelos serão utilizadas diversas métricas amplamente empregadas na literatura. Na Subseção 3.4.1 são apresentadas as ferramentas de análise

estatística voltadas principalmente para análise de modelos baseados em triplas e na Subseção 3.4.2 são apresentadas as principais ferramentas estatísticas empregadas na avaliação de modelos voltados para a tarefa *Knowledge Base Question Answering (KB-QA)* empregadas neste trabalho, respectivamente, para avaliação do modelo composto e dos modelos completos + *Ensembles* gerados.

3.4.1 Ferramentas de Análise do Modelo Fim-a-fim Composto

O *MRR* é a média de todas as classificações recíprocas para os candidatos reais sobre o conjunto de teste. Assim,

$$MRR = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{r_i} \tag{3.10}$$

em que m é o número de questões de teste e r_i é o *rank* da resposta correta para a questão i.

A métrica $H@k$ é a taxa de entidades corretas que aparecem entre as primeiras-k predições de resposta para uma questão. Este número pode ser maior que 1 se as questões possuírem mais de uma resposta correta. Assim,

$$H@k = \sum_{i=1}^{m}\frac{c_i}{m} \tag{3.11}$$

sendo m o número de questões de teste e c_i o número de respostas corretas que aparecem entre as primeiras k predições de respostas para questão i.

3.4.2 Ferramentas de Análise dos *Ensembles* e dos Modelos Fim-a-fim Completos

A precisão é a métrica que indica o percentual de respostas que são corretas dentre as respostas retornadas pelo modelo para uma questão. Desta forma, a precisão média de um modelo pode ser definida como,

$$Precisão = \frac{1}{m}\left(\sum_{i=1}^{m}\frac{c_i}{t_i}\right) \tag{3.12}$$

em que m é o número de questões avaliadas, c_i é o número de respostas corretas retornadas pelo modelo para a questão i e t_i é o número total de respostas retornadas pelo modelo para a questão i.

O *Recall* demonstra o percentual das respostas corretas que o modelo é capaz de recuperar para uma questão. Assim, o *Recall* médio de um modelo é definido como,

$$Recall = \frac{1}{m}\left(\sum_{i=1}^{m}\frac{c_i}{n_i}\right) \tag{3.13}$$

em que m é o número de questões avaliadas, c_i é o número de respostas corretas retornadas pelo modelo para a questão i e n_i é o número de respostas corretas existentes para a questão i.

A principal métrica de avaliação de desempenho empregada nesta pesquisa é o *F1-score*, que é uma ponderação entre a Precisão e o *Recall* médios, também denominada Macro *F1-score*, definida como,

$$F1\text{-}score = \frac{1}{m}\left(\sum_{i=1}^{m}\frac{2*\frac{c_i}{t_i}*\frac{c_i}{n_i}}{\frac{c_i}{t_i}+\frac{c_i}{n_i}}\right) \tag{3.14}$$

em que m é o número de questões avaliadas, c_i é o número de respostas corretas retornadas pelo modelo para a questão i, t_i é o número total de respostas retornadas pelo modelo para a questão i e n_i é o número de respostas corretas existentes para a questão i.

Já a métrica Micro *F1-score* é uma medida da acurácia do modelo considerando-se as respostas individualmente, independente dos conjuntos de respostas das questões. Assim ela é definida como,

$$Micro\ F1 = \left(\frac{2*\frac{c_t}{t_t}*\frac{c_t}{n_t}}{\frac{c_t}{t_t}+\frac{c_t}{n_t}}\right) \tag{3.15}$$

em que c_t é o número de respostas corretas retornadas pelo modelo considerando-se todas as questões, t_t é o número total de respostas retornadas pelo modelo considerando-se todas as questões e n_t é o número de respostas corretas existentes considerando-se todas as questões.

A métrica *H@1* é importante para verificar-se a confiabilidade da primeira resposta retornada pelo modelo para as questões e é definida como,

$$H@1 = \frac{c}{m} \tag{3.16}$$

onde m é o número de questões avaliadas e c é o número de questões para as quais a primeira resposta retornada pelo modelo é uma resposta correta.

A métrica Acurácia Real é uma medida utilizada para verificar a capacidade dos modelos de acertarem completamente os conjuntos de respostas das questões. Assim, é definida como,

$$Acurácia\ Real = \frac{q}{m} \tag{3.17}$$

onde m é o número de questões avaliadas e q é o número de questões para as quais o modelo é capaz de acertar todo o conjunto de respostas, ou seja, o número de questões para as quais o modelo atinge *F1-score=1*.

Por fim, para analisar a capacidade dos modelos de não retornarem respostas erradas para as questões, implementou-se uma métrica que aqui é denominada de Taxa de Questões com Apenas Verdadeiro Positivos (TP), definida como,

$$TP = 100 * \frac{q}{m} \tag{3.18}$$

em que m é o número de questões avaliadas e q é o número de questões para as quais dentre o conjunto de respostas retornados pelo modelo para a questão, não haja nenhuma resposta errada.

4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados do trabalho e da pesquisa realizados, e as devidas discussões de tais resultados. Referente ao modelo composto desenvolvido, são dedicadas a Seção 4.1, que mostra os resultados e análises do modelo para consultas à *KB (MINERVA)* de forma independente, e a Seção 4.2, que apresenta os resultados e análises do modelo composto completo para o banco de dados *WikiMovies*. As análises e resultados referentes aos *Ensembles* desenvolvidos são apresentados na Seção 4.3.

4.1 Resultados *MINERVA*

O *MINERVA* foi executado nos bancos de dados apresentados na tabela 3.1 e os resultados obtidos são apresentados na tabela 4.1. É possível notar que o desempenho decai de acordo com o tamanho da *knowledge base*. Seus resultados são muito próximos ou superiores aos dos modelos mais recentes encontrados na literatura (vide Das et al. (2018)), e apesar de seu desempenho decair com o tamanho da *KB*, esta ocorrência também se verifica para outros modelos, uma vez que o tamanho da base está diretamente relacionado à dificuldade do problema. Seu desempenho também se mostra atrelado à generalização de caminhos, por exemplo, na *KB* FB15K-237 que possui menor quantidade de caminhos generalizáveis, o modelo obteve os menores valores para ambas as métricas de desempenho.

Tabela 4.1 – Desempenho do *MINERVA* nas diferentes *KBs*

	HITS@1	HITS@3	HITS@10	MRR
COUNTRIES	0,7083	1,0000	1,0000	0,8472
UMLS	0,6778	0,8790	0,9259	0,7854
KINSHIP	0,4721	0,7709	0,9302	0,6363
WN18RR	0,4666	0,7108	0,7967	0,6002
NELL-995	0,4276	0,4697	0,5284	0,4598
FB15K-237	0,1968	0,3027	0,4311	0,2727

Fonte: Do Autor (2022)

Para verificar se há a dependência do modelo ao conjunto de treino para gerar respostas para cada tipo de pergunta, adicionou-se ao conjunto de treino, validação e teste da *Knowledge Base COUNTRIES*, questões relacionadas à vizinhança dos países, ou seja, questões com a relação *'neighborOf'* extraídas de seu grafo. As amostras do tipo '*neighborOf*' a serem adicionadas para o conjunto de treino foram selecionadas de forma que o país entidade tópico houvesse ao menos três vizinhos distintos, de forma que pudesse ser adicionada ao menos uma

amostra correspondente aquele país entidade tópico para o conjunto de treino, uma amostra para o conjunto de validação e uma para o conjunto de teste. Os conjuntos de treino, validação e teste originais da *KB* só possuíam questões relacionadas à localização, ou seja, apenas amostras com relação do tipo *'locatedIn'*. Na Tabela 4.2 são mostrados as informações dos conjuntos de treino, validação e teste do banco de dados *COUNTRIES* original e do banco com adição das novas amostras, bem como o incremento percentual do número de amostras com relação ao banco *COUNTRIES* original.

Tabela 4.2 – Dados dos conjuntos do banco *COUNTRIES* original e do banco *COUNTRIES* com questões de vizinhança dos países

Conjuntos	COUNTRIES	Novo COUNTRIES	Incremento
Treino	118	150	27,1%
Validação	24	36	50,0%
Teste	24	36	50,0%

Fonte: Do Autor (2022)

Observe que o novo banco de dados *COUNTRIES* aumenta a complexidade da tarefa em relação ao *COUNTRIES* original, devido a inserção de um novo tipo de questão e incremento do tamanho do banco de dados com amostras de um novo tipo. O resultado comparativo do desempenho do *MINERVA* na *KB* original e após a inserção das novas questões é sintetizado na tabela 4.3.

Tabela 4.3 – Desempenho do *MINERVA* na COUNTRIES original e na COUNTRIES com questões de vizinhança dos países

	COUNTRIES (original)	COUNTRIES (questões adicionadas)
HITS@1	0,7083	0,4167
HITS@3	1,0000	0,5833
HITS@5	1,0000	0,6667
HITS@10	1,0000	0,8056
HITS@20	1,0000	0,9167
MRR	0,8472	0,5392

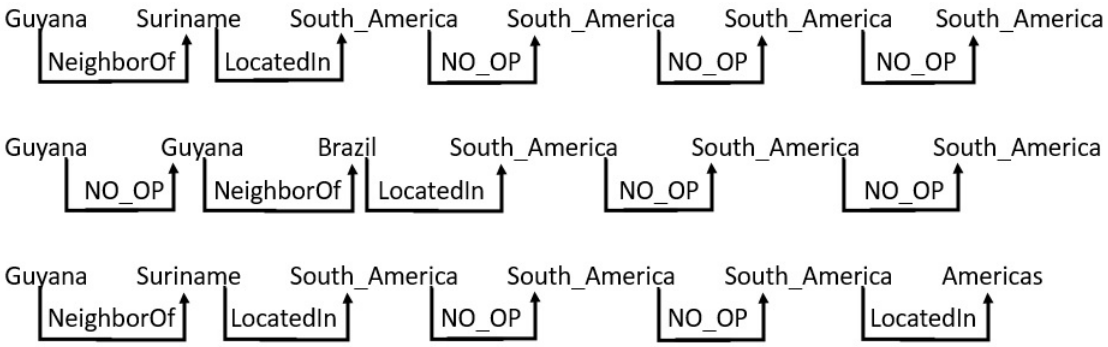
Fonte: Do Autor (2022)

Analisando-se os resultados dispostos na Tabela 4.3 do modelo para a *KB COUNTRIES* original e com adição de *queries* do tipo *'neighborOf'* verifica-se que há uma redução do desempenho do modelo em geral. Isto ocorre devido ao aumento da dificuldade da tarefa, devido a inclusão de um novo tipo de questão e, principalmente, devido às questões do tipo *'neighborOf'* possuírem mais de uma resposta correta, enquanto em treino, desenvolvimento e teste apenas uma destas é considerada na tripla a cada vez. Como o modelo não possui meios de distinguir

qual é a esperada para cada tripla, o retorno de uma resposta correta pode ainda contar como um erro, acarretando em uma redução de desempenho.

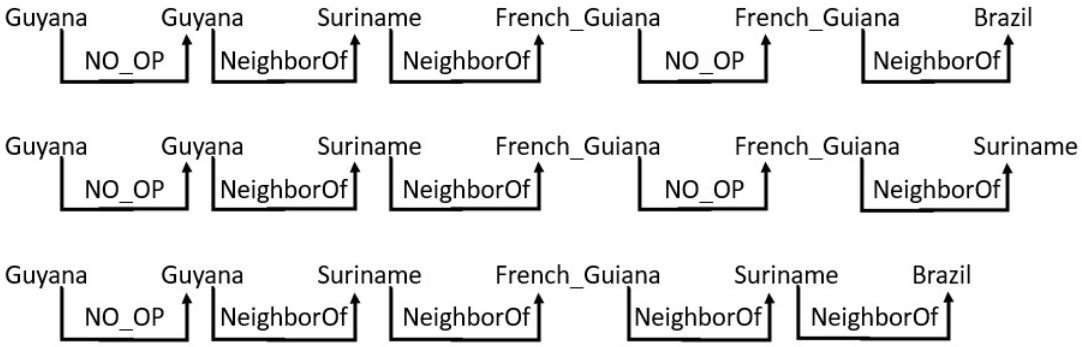
O modelo *MINERVA* disponibiliza na saída os caminhos tomados para encontrar as respostas às questões (*queries*). Os caminhos tomados para solucionar a *query* (*Guyana*, *neighborOf*, ?), antes de treinar com questões do tipo *'neighborOf'* e após treinar com questões deste tipo, podem ser observados, respectivamente, nas Figuras 4.1 e 4.2.

Figura 4.1 – Caminhos para solução da *query* (*Guyana*, *neighborOf*, ?) antes da adição deste tipo de questão ao treino



Fonte: Do Autor (2022)

Figura 4.2 – Caminhos para solução da *query* (*Guyana*, *neighborOf*, ?) após adição deste tipo de questão ao treino



Fonte: Do Autor (2022)

Por fim, observando-se os caminhos desenvolvidos para responder a *query* (*Guyana*, *neighborOf*, ?) antes da inclusão de questões de vizinhança no treino do modelo, é possível verificar na Figura 4.1 que em nenhuma das 3 *rollouts* um vizinho correto de *Guyana* foi retornado, sendo as entidades finais retornadas *South_America, South_America* e *Americas*, respostas associadas a query *'locatedIn'* que eram as únicas inclusas no treino. Já após a inclusão de questões do tipo *'neighborOf'* no treino, analisando-se os caminhos resultantes para a busca do modelo na Figura 4.2, verifica-se que o modelo consegue retornar respostas corretas

para a *query* (*Guyana*, *neighborOf*, ?) nos 3 *rollouts*, sendo as entidades finais retornadas em cada *rollout*, *Brazil, Suriname* e *Brazil*, respectivamente. Com isso, comprova-se que a capacidade do modelo de responder a um determinado tipo de questão (relação) está atrelada ao treino do modelo. Ainda além, verifica-se que o modelo é capaz de diversificar os caminhos e obter respostas diferentes para uma mesma questão, o que é um ponto positivo visando uma aplicação real.

4.2 Experimento e Resultados do Modelo *Parot-MINERVA*

Na análise do modelo composto *Parot-MINERVA* foi empregado o banco de dados *Wiki-Movies* (Subseção 3.1.1). Para avaliar o modelo, é necessário um procedimento de duas etapas. Na primeira etapa aplica-se a reimplementação do *Parot* para a extração de triplas das questões em linguagem natural. O resumo dos resultados obtidos para a primeira etapa é sintetizado na Tabela 4.4.

Tabela 4.4 – Resultados da Extração de Triplas via Implementação do *Parot*

Número de *queries* relacionais obtidas	4147
Número de *queries* não relacionais obtidas	1802
Número de *queries* relacionais impróprias	1354
Número de *queries* adjetivas	255
Número de questões sem extração de *queries*	2442

Fonte: Do Autor (2022)

Para a primeira etapa, a atual implementação do *Parot* consegue a extração de 5949 triplas adequadas ao *MINERVA* (triplas relacionais + triplas não relacionais da Tabela 4.4) que corresponde a 59,49% das questões empregadas. As triplas *queries* relacionais impróprias correspondem a 13,54% das questões. São consideradas *queries* impróprias as triplas relacionais extraídas em que ambas as entidades (e_1 e e_2) são consideradas variáveis pelo *Parot* e, portanto, não há como o *MINERVA* responder a *query*. Com a implementação atual, 2,5% das questões só geraram triplas adjetivas. Triplas adjetivas só são extraídas na implementação atual do *Parot* caso não fossem obtidas triplas relacionais ou não relacionais, uma vez que as triplas adjetivas não são solucionáveis pelo *MINERVA* e são sempre tratadas como triplas complementares. Por fim, para 24,42% das questões, não são obtidas triplas de nenhum tipo. A grande maioria ocorre por falha das regras implementadas e uma pequena parte das falhas de extração ocorre devido às questões não serem iniciadas com palavras *Wh*.

A segunda etapa refere-se a aplicação do *MINERVA* para consulta à *KB*. As questões que tiveram *queries* extraídas adequadas ao uso no *MINERVA* (5949 questões das 10000) são rearranjadas de maneira a formar uma tripla para cada resposta correta pertinente a sua respectiva questão. Com isso, para a segunda etapa é gerado um banco de dados com um total de 12511 *queries*, que após embaralhadas são divididas em 80% das amostras para treino (*train set*), 10% das amostras para validação (*validation set*) e 10% para teste (*test set*). Os resultados obtidos em treino e teste são mostrados na Tabela 4.5.

Tabela 4.5 – Desempenho do *MINERVA* em treino e teste

	Resultados Treino	Resultados Teste
HITS@1	0,1471	0,1367
HITS@3	0,1543	0,1471
HITS@5	0,1663	0,1551
HITS@10	0,1695	0,1655
HITS@20	0,1815	0,1703
MRR	0,1548	0,1444

Fonte: Do Autor (2022)

Os resultados iniciais obtidos mostrados na Tabela 4.5 para o modelo são bem abaixo dos resultados que são relatados em Yang, Yang e Cohen (2017) para a base de dados *WikiMovies*, onde os modelos mostram acurácias entre 78% e 95%. Os próprios autores do *MINERVA* também realizaram um experimento utilizando-se o *MINERVA* para o banco de dados *WikiMovies* e relatam uma acurácia de 96,7% em seu artigo (DAS et al., 2018), no entanto a implementação utilizada não foi disponibilizada. Outra informação importante de se destacar é que as questões contidas em *WikiMovies* são todas questões que podem ser respondidas via *1-hop* de busca na *KB*, onde já foi verificado que o *MINERVA* é capaz de solucionar questões de maior complexidade em experimentos anteriores.

Com base nestas informações, é possível sugerir que a maior fonte de erros está na formação das *queries* passadas para o *MINERVA*. Ao analisar-se as triplas formadas é possível verificar que para muitas questões, devido ao *Parot* nos casos de relações verbais primeiramente determinar as relações ou frases relacionais na questão antes de extrair as entidades, ocorre confusão entre a relação e a entidade, principalmente nos casos em que nomes de filmes contém verbos, e assim a real relação não é extraída pela implementação do *Parot* e consequentemente a própria entidade da questão pode acabar sendo identificada de forma errada. Além disso, verifica-se também que o método empregado na implementação para reconhecimento de entidades em alguns momentos são falhos, como por exemplo para a questão "*what is the primary*

language in the film Peyton Place?" onde a entidade reconhecida pelo modelo é "*film*" quando a real entidade que deveria ter sido reconhecida é "*Peyton Place*".

Além dos problemas maiores citados anteriormente, ainda ocorrem algumas confusões devido às regras implementadas em *Parot*, que apesar de gerar triplas no formato adequado para o *MINERVA*, em alguns casos não extraem a real informação necessária para a devida representação da questão. Portanto, é possível supor que com uma melhor extração de triplas, o desempenho do modelo composto pode ser de maior relevância. Assim, em trabalhos futuros podem ser buscadas novas formas de abordagem para representação das questões em forma de triplas para o desenvolvimento de um modelo composto, uma vez que é interessante o uso do *MINERVA* por ele não necessitar de fontes de dados externas à *KB* e sua capacidade de lidar com questões *Multi-hop* de maneira implícita.

4.3 Resultados do *Ensemble*

De acordo com a metodologia aplicada, foram reavaliados vários modelos voltados para a tarefa de *Knowledge Base Question Answering (KB-QA)* que pudessem compor o módulo *soft-KB lookup* de um assistente virtual, com abordagens de recuperação de informação ou de análise semântica. Alguns exemplos de modelos testados são *Neural State Machines teacher-student (NSM)* (HE et al., 2021), *STAged Query Graph Generation (STAGG)* (YIH et al., 2015), *Neural-Symbolic Complex Question Answering (NS-CQA)* (HUA et al., 2020) e *Multi-hop Complex KBQA (Multihop)* (LAN; JIANG, 2020).

Diversas abordagens foram tomadas para estruturação do *Ensemble*, considerando-se diferentes conjuntos de modelos. Dentre as configurações abordadas, a que apresentou melhor resultado em termos de *F1-score* foi o *Ensemble* por voto majoritário composto pelos modelos *Multihop* (LAN; JIANG, 2020), *NS-CQA* (HUA et al., 2020) e o *BAMnet* (CHEN; WU; ZAKI, 2019), que foram melhores avaliados individualmente de acordo com a reavaliação realizada, considerando também que alguns modelos reavaliados apresentaram resultados inconsistentes com os apresentados em seus artigos.

Os resultados de desempenho do *Ensemble* por voto majoritário e das reavaliações dos modelos selecionados com relação a *F1-score* são sintetizados na Tabela 4.6. Além disso, na mesma Tabela (Tabela 4.6) são apresentadas outras métricas auxiliares que permitem realizar uma análise mais ampla de vários aspectos do *Ensemble* que são importantes para sua implementação em um assistente virtual.

Tabela 4.6 – Desempenho dos modelos individuais selecionados para diferentes métricas relevantes

	Multihop	*NS-CQA*	*BAMnet*	*Ensemble*
F1-score	0,7312	0,7138	0,5722	0,7540
Precisão	0,7344	0,7130	0,6198	0,7703
Recall	0,7947	0,7391	0,6324	0,7803
Micro F1	0,5804	0,8509	0,2708	0,7661
Acurácia Real	0,6431	0,6425	0,3667	0,6480
H@1	0,7297	0,7175	0,6162	0,7657

Fonte: Do Autor (2022)

Observando-se os resultados da Tabela 4.6 é possível notar que o *Ensemble* é capaz de superar todos os três modelos individuais no aspecto de *F1-score*, que é a principal métrica de cobertura das respostas pelos modelos e principal métrica de desempenho adotada, bem como nas outras métricas auxiliares, com exceção da Micro *F1-score* onde o modelo *NS-CQA* consegue um melhor resultado.

O *Ensemble* cujos resultados são descritos na Tabela 4.6, no entanto, possui brecha para aprimoramento uma vez que possui 192 questões onde não ocorre consenso dentre os modelos para as respostas e, desta forma, não são respondidas. Seguindo como descrito na metodologia aplicada, através da reavaliação dos modelos individuais para as questões sem repostas são gerados os resultados da Tabela 4.7.

Tabela 4.7 – Desempenho dos modelos individuais selecionados para questões sem consenso

	Multihop	*NS-CQA*	*BAMnet*
F1-score	0,2443	0,2912	0,1259

Fonte: Do Autor (2022)

Através da análise da Tabela 4.7 são desenvolvidos os modelos *Ensemble* com contramedida, onde a contramedida é o modelo *NS-CQA*, e o *Ensemble* com contramedida total, onde de acordo com a Tabela 4.7, as prioridades são *NS-CQA* > *Multihop* > *BAMnet*. Observou-se que para este caso, considerando-se o banco de dados *WebQuestionsSP*, o *BAMnet* acaba não fornecendo respostas uma vez que os casos restantes em que o *NS-CQA* não fornece respostas são completamente supridos pelo *Multihop*, no entanto, isto pode não acontecer para outros bancos de dados. Os resultados de desempenho para ambos os novos *Ensembles* gerados em relação a *F1-score* e outras métricas auxiliares são mostrados na Tabela 4.8.

Observando-se os resultados dispostos na Tabela 4.8 dos modelos de *Ensembles* com contramedida e com contramedida total em comparação aos resultados do modelo de *Ensemble* sem nenhum tipo de contramedida, mostrado na Tabela 4.6, é possível verificar que utilizar-se

Tabela 4.8 – Desempenho dos modelos finais de *Ensembles* desenvolvidos para diferentes métricas relevantes

	Ensemble c/ contramedida	*Ensemble* c/ contramedida total
F1-score	0,7872	0,8143
Precisão	0,8034	0,8311
Recall	0,8138	0,8456
Micro F1	0,8579	0,8420
Acurácia Real	0,6803	0,7010
H@1	0,7993	0,8255

Fonte: Do Autor (2022)

contramedidas para os casos sem consenso garante uma melhora de desempenho para ambos os modelos, com contramedida e com contramedida total, em todos os aspectos das tabelas. Além disso, o modelo de *Ensemble* com contramedida total possui o desempenho em geral superior ao modelo com contramedida simples, com exceção do micro *F1-score*.

Com os resultados obtidos é possível destacar os seguinte benefícios visando o emprego dos *Ensembles* desenvolvidos no módulo *soft-KB lookup* proposto:

a) Considerando o *F1-score*: empregar o *Ensemble* com contramedida total garante uma maior cobertura em geral das respostas corretas das questões. É importante notar que o modelo *Multihop* é o modelo *KB-QA* estado da arte para o banco de dados *Web Questions SP*, com um *F1-score* de 74,0%, e o *Ensemble* com contramedida total atinge um *F1-score* de 81,43% (LAN; JIANG, 2020);

b) Considerando-se a Precisão: o *Ensemble* com contramedida total garante que dentre as respostas retornadas pelo modelo haja um maior número de respostas corretas, com um aumento de precisão de 9,67% em relação ao modelo individual com a maior precisão (Multihop);

c) Considerando-se o *Recall*: o *Ensemble* com contramedida total atinge um aumento no *Recall* de 5,09% com relação ao melhor modelo individual neste aspecto (Multihop), o que garante que maior número das respostas corretas sejam recuperadas pelo modelo;

d) Considerando-se o Micro *F1-score*: neste caso o único modelo capaz de superar o melhor modelo individual foi o *Ensemble* com contramedida simples, obtendo um desempenho de 85,79% superando o melhor modelo individual (*NS-CQA*) em 0,7%. Ainda assim, o *Ensemble* com contramedida total obtém um desempenho competitivo de 84,20%. A Micro *F1-score* é uma medida de acurácia a nível de respostas individuais do modelo.

É preciso observar que os valores mais elevados do *Ensemble* simples, do *Ensemble* com contramedida simples e do *NS-CQA* se dá devido ao número de questões as quais não são capazes de responder, e considerando-se isto em conjunto com as outras análises realizadas, verifica-se uma abrangência geral superior por parte do *Ensemble* com contramedida total;

e) Considerando-se a Acurácia Real: o *Ensemble* com contramedida total atinge uma Acurácia Real de 70,10%, o que representa uma melhora de desempenho de 5,31% em relação ao melhor modelo individual neste aspecto (Multihop). Isso mostra que o modelo é capaz de acertar um maior número de questões completamente;

f) Considerando-se *H@1*: o *Ensemble* com contramedida total possui um desempenho de 82,55%, uma superação de 9,58% em relação ao melhor modelo individual nesta métrica (Multihop). Existem situações em que o usuário não necessariamente necessita de um conjunto de respostas completo, mas apenas uma resposta correta, e nestes casos o modelo com maior *H@1* possui uma maior chance de suprir tal necessidade.

Para complementar a análise, na Tabela 4.9 são mostrados os resultados de desempenho obtidos pelos *Ensembles* gerados e pelos modelos individuais empregados, com relação a taxa de questões com apenas verdadeiros positivos (TP) como descrita na Subseção 3.4.2 e também com relação a taxa de questões com apenas verdadeiros positivos desconsiderando-se as questões sem respostas (TP*).

Tabela 4.9 – Desempenho dos modelos individuais selecionados e dos *Ensembles* com relação a métrica TP

Modelos	TP	TP*	#Questões sem respostas
Multihop	69,12%	69,12%	0
NS-CQA	93,41%	68,88%	402
BAMnet	48,62%	48,62%	0
Ensemble simples	85,05%	73,34%	192
Ensemble c/ contramedida	84,75%	76,63%	133
Ensemble c/ contramedida total	79,07%	79,07%	0

Fonte: Do Autor (2022)

De acordo com os resultados dispostos na Tabela 4.9, o melhor desempenho com relação à métrica TP dentre modelos individuais e *Ensembles* é o *NS-CQA* com uma taxa de 93,41%, que supera o segundo melhor modelo, o *Ensemble* simples, por uma taxa de 8,36%. Dentre as

três versões de *Ensembles* desenvolvidas, observa-se na Tabela 4.9 uma redução da TP a medida que são fornecidas respostas para mais questões.

Pensando-se por exemplo, em um assistente virtual voltado para atendimento de suporte em um *e-commerce*, um valor de TP mais alto traz uma maior garantia de que não será fornecida uma resposta errada ao cliente, por exemplo se a sugestão oferecida contrariar o que seria a solução, ou mesmo criar outro problema, o que seria inconveniente e poderia trazer prejuízos a empresa.

É importante notar, no entanto, que a maior taxa de questões com apenas verdadeiros positivos obtida pelo modelo *NS-CQA* se dá principalmente devido ao modelo não retornar respostas para uma grande quantidade de questões, sendo que 24,53% das questões não são respondidas pelo modelo *NS-CQA*, o que também é inconveniente pensando-se no caso hipotético de um assistente virtual para atendimento de suporte em um *e-commerce*, uma vez que o assistente seria incapaz de solucionar boa parte dos problemas, o que também geraria gastos para a empresa para contornar a situação. Além disso, pensando em um assistente virtual voltado para entretenimento, como a Cortana do Google ou Alexa da Amazon, um modelo que não é capaz de responder a muitas questões acaba sendo vago e entediante, o que acaba afastando os usuários (ADIWARDANA et al., 2020).

Desta forma, uma maior taxa de questões com apenas verdadeiros positivos *TP* acompanhado de uma grande quantidade de questões não respondidas, no geral, não satisfaria o problema. Com isto, uma análise mais interessante pode ser avaliar os modelos com relação a taxa de questões com apenas verdadeiros positivos desconsiderando-se as questões sem respostas dos modelos (TP*) dispostas na Tabela 4.9. Com base na TP*, verifica-se que todos os *Ensembles* conseguem aprimorar as taxas de questões com apenas verdadeiros positivos com respostas e que tomar contramedidas que permitam aos modelos que sejam capazes de responder a mais questões ainda os torna mais acurados neste sentido.

É notável que considerando-se a TP*, o melhor modelo de *Ensemble* desenvolvido, o *Ensemble* com contramedida total, consegue superar o melhor modelo individual nesta métrica, o *Multihop*, em 9,95%. Observa-se ainda que o modelo individual *NS-CQA* sofre uma redução abrupta de 24,53% considerando suas TP e TP* dispostas na Tabela 4.9, o que o torna inferior neste aspecto ao *Multihop* que era o segundo melhor modelo individual considerando-se a TP.

À luz dos resultados, análises e discussões apresentadas, supõe-se que a melhor situação seja conciliar maiores TP* e *F1-score*. Assim, apesar de não possuir a mais alta TP, o *Ensemble*

com contramedida total é capaz de agregar muitos benefícios em relação aos modelos individuais avaliados, o que o torna a melhor opção em geral para no futuro compor o módulo *soft-KB lookup* do Assistente Virtual.

5 CONCLUSÃO

Com a análise dos diferentes modelos para a tarefa de *Knowledge Base Question Answering (KB-QA)* e o desenvolvimento dos *Ensembles*, principalmente o *Ensemble* com contramedida total, atingiu-se um resultado antes ainda não obtido na literatura, com um incremento na *F1-score* de 7,43% em relação ao modelo *KB-QA* individual estado-da-arte, considerando-se o banco de dados *WebQuestionsSP*. Neste sentido, aplicar tal modelo como o módulo *soft-KB lookup* de um assistente virtual poderia apresentar resultados interessantes.

O desenvolvimento de um modelo *KB-QA* composto, no entanto, apresentou resultados com grandes falhas para o banco de dados *WikiMovies*, que comparado ao banco de dados *WebQuestionsSP* é relativamente simples, por não conter questões com restrições e que necessitem de mais de um passo de busca na *KB* para determinação da resposta. Grande parte da responsabilidade de tais erros, através das análises das extrações geradas e análise prévia do *MINERVA* individualmente, pode ser atribuída ao método de conversão da questão em linguagem natural para triplas, *Parot*, que gera triplas viáveis para apenas aproximadamente 60% das questões, e que após analise mais profunda dentre estas, verifica-se ainda muitos erros relacionados a extração de entidades e formação das relações.

Para trabalhos futuros, é possível tentar aprimorar ainda mais o modelo combinado via *Ensemble* buscando-se novos modelos para *KB-QA* individuais, em vista de que visando os 100% em F1-score, o modelo *Ensemble* com contramedida total, o melhor modelo gerado, ainda tem uma margem de aproximadamente 19% para melhora, além de que o modelo individual *BAMnet* empregado nos *Ensembles* é relativamente inferior aos outros dois modelos individuais empregados, com uma diferença em *F1-score* de aproximadamente 16% para o melhor modelo individual, o *Multihop*), e aproximadamente 14% para o segundo, o *NS-CQA*, o que nesta configuração tende a inferiorizar o resultado final na média, havendo um certo nível de diversidade de respostas dentre os modelos. Pode-se também aplicar diferentes configurações de *Ensembles*, seja empregando-se mais modelos em sua composição, ou que empreguem informações das saídas dos modelos individuais e informações de classificação das questões para seleção de respostas, à nível individual ou de conjunto.

Outra possibilidade para trabalhos futuros, considerando-se o modelo *KB-QA* composto, é tentar utilizar novas abordagens para a representação da questão em linguagem natural na forma de uma *query*, que é a maior fonte de falhas do modelo, o que pode resultar em um modelo que seja competitivo com os atuais modelos *KB-QA* fim-a-fim. Uma possibilidade seria utilizar

por exemplo, as informações contidas nas *SPARQLs* geradas pelo *Multihop*, o que necessitaria de algumas adaptações em vista de utilizar-se o *MINERVA* para realizar as consultas.

Além disso, em trabalhos futuros, visando o emprego de tais modelos como parte de um assistente virtual voltado para o *e-commerce*, é necessário o levantamento de bancos de dados que sejam interessantes e voltados para o comércio eletrônico, ou mesmo bancos gerados pelas próprias empresas, e a realização de análises e testes dos modelos para este nicho específico. No mais, são necessários o desenvolvimento dos outros módulos para um assistente virtual, para gerenciar o diálogo, gerar respostas em linguagem natural e lidar com outras características do diálogo, como opiniões e emoções apresentadas pelos usuários.

REFERÊNCIAS

ABCOMM. **Inteligência artificial, robôs e automação: O futuro do e-commerce**. 2020. Disponível em: <https://abcomm.org/noticias/inteligencia-artificial-robos-e-automacao-o-futuro-do-e-commerce/>. Acesso em: 03 ago. 2020.

ABCOMM. **O crescimento dos marketplaces em 2021**. 2021. Disponível em: <https://abcomm.org/noticias/o-crescimento-dos-marketplaces-em-2021/>. Acesso em: 13 abr. 2022.

ADIWARDANA, D. et al. Towards a Human-like Open-Domain Chatbot. 2020. Disponível em: <http://arxiv.org/abs/2001.09977>.

AUER, S. et al. Dbpedia: A nucleus for a web of open data. In: **The semantic web**. [S.l.]: Springer, 2007. p. 722–735.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval: The Concepts and Technology behind Search**. 2nd. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 9780321416919.

BLIP, T. **Chatbot: o que é, como funciona, benefícios e cases**. 2019. Disponível em: <https://www.take.net/blog/chatbots/chatbot/>. Acesso em: 25 jun. 2021.

BOLLACKER, K. et al. Freebase: A collaboratively created graph database for structuring human knowledge. In: **Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2008. (SIGMOD '08), p. 1247–1250. ISBN 9781605581026. Disponível em: <https://doi.org/10.1145/1376616.1376746>.

BORDES, A. et al. Large-scale simple question answering with memory networks. **arXiv preprint arXiv:1506.02075**, 2015.

BOUCHARD, G.; SINGH, S.; TROUILLON, T. On approximate reasoning capabilities of low-rank vector spaces. **AAAI Spring Symposium - Technical Report**, SS-15-03, p. 6–9, 2015.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research. **IEEE Computational intelligence magazine**, IEEE, v. 9, n. 2, p. 48–57, 2014.

CAO, N. D.; KIPF, T. Molgan: An implicit generative model for small molecular graphs. **ArXiv**, abs/1805.11973, 2018.

CHEN, Y.; WU, L.; ZAKI, M. J. Bidirectional attentive memory networks for question answering over knowledge bases. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 2913–2923. Disponível em: <https://www.aclweb.org/anthology/N19-1299>.

CHO, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference**, p. 1724–1734, 2014.

CLARK, P. et al. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. 2018. Disponível em: <http://arxiv.org/abs/1803.05457>.

COLLOBERT, R. et al. Natural language processing (almost) from scratch. **Journal of machine learning research**, v. 12, n. ARTICLE, p. 2493–2537, 2011.

CUI, W. et al. Kbqa: learning question answering over qa corpora and knowledge bases. **arXiv preprint arXiv:1903.02419**, 2019.

DALE, R. Gpt-3: What's it good for? **Natural Language Engineering**, Cambridge University Press, v. 27, n. 1, p. 113–118, 2021.

DAS, R. et al. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. **ArXiv**, abs/1711.05851, 2018.

DETTMERS, T. et al. Convolutional 2D knowledge graph embeddings. **32nd AAAI Conference on Artificial Intelligence, AAAI 2018**, p. 1811–1818, 2018.

DHINGRA, B. et al. Towards end-to-end reinforcement learning of dialogue agents for information access. **arXiv preprint arXiv:1609.00777**, 2016.

DO, K.; TRAN, T.; VENKATESH, S. Graph transformation policy network for chemical reaction prediction. **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**, 2019.

DUNN, M. et al. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. 2017. Disponível em: <http://arxiv.org/abs/1704.05179>.

E-COMMERCE_BRASIL. **ABComm projeta faturamento de R$ 169,5 bilhões no e-commerce em 2022**. 2022. Disponível em: <https://www.ecommercebrasil.com.br/noticias/abcomm-faturamento-e-commerce-2022/>. Acesso em: 13 abr. 2022.

ELMAN, J. L. Finding structure in time. **Cognitive Science**, v. 14, n. 2, p. 179–211, 1990. ISSN 03640213.

FADER, A.; SODERLAND, S.; ETZIONI, O. Identifying relations for open information extraction. In: **Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing**. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011. p. 1535–1545. Disponível em: <https://aclanthology.org/D11-1142>.

FU, B. et al. A survey on complex question answering over knowledge base: Recent advances and challenges. **CoRR**, abs/2007.13069, 2020. Disponível em: <https://arxiv.org/abs/2007.13069>.

GAO, J.; GALLEY, M.; LI, L. Neural approaches to conversational AI. **ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference Tutorial Abstracts**, p. 2–7, 2018.

GEHRING, J. et al. Convolutional sequence to sequence learning. In: PRECUP, D.; TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning**. PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 1243–1252. Disponível em: <https://proceedings.mlr.press/v70/gehring17a.html>.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with LSTM. **Neural Computation**, v. 12, n. 10, p. 2451–2471, 2000. ISSN 08997667.

GOODFELLOW, I. et al. Generative adversarial nets. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. Curran Associates, Inc., 2014. p. 2672–2680. Disponível em: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

GRAVES, A.; JAITLY, N. Towards end-to-end speech recognition with recurrent neural networks. **31st International Conference on Machine Learning, ICML 2014**, v. 5, p. 3771–3779, 2014.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. **Neural Networks**, v. 18, n. 5-6, p. 602–610, 2005. ISSN 08936080.

HANIN, B. Which neural net architectures give rise to exploding and vanishing gradients? **Advances in Neural Information Processing Systems**, v. 2018-December, n. iii, p. 582–591, 2018. ISSN 10495258.

HE, G. et al. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In: **Proceedings of the 14th ACM International Conference on Web Search and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2021. (WSDM '21), p. 553–561. ISBN 9781450382977. Disponível em: <https://doi.org/10.1145/3437963.3441753>.

HE, W. et al. DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications. p. 37–46, 2019.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997. ISSN 08997667.

HU, S. et al. Answering natural language questions by subgraph matching over knowledge graphs. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 30, n. 5, p. 824–837, 2017.

HUA, Y. et al. Less is more: Data-efficient complex question answering over knowledge bases. **Journal of Web Semantics**, v. 65, p. 100612, 2020. ISSN 1570-8268. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570826820300470>.

JAGVARAL, B. et al. Path-based reasoning approach for knowledge graph completion using cnn-bilstm with attention mechanism. **Expert Systems with Applications**, v. 142, p. 112960, 2020. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417419306785>.

JOSHI, N. **Yes, Chatbots And Virtual Assistants Are Different!** 2018. Disponível em: <https://www.forbes.com/sites/cognitiveworld/2018/12/23/yes-chatbots-and-virtual-assistants-are-different/#6691fd7a6d7d>. Acesso em: 01 ago. 2020.

KARPATHY, A.; FEI-FEI, L. Deep Visual-Semantic Alignments for Generating Image Descriptions. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 39, n. 4, p. 664–676, 2017. ISSN 01628828.

KEPUSKA, V.; BOHOUTA, G. Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home). **2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018**, v. 2018-January, n. c, p. 99–103, 2018.

KIM, Y. Convolutional neural networks for sentence classification. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1746–1751. Disponível em: <https://www.aclweb.org/anthology/D14-1181>.

KOČISKÝ, T. et al. The NarrativeQA Reading Comprehension Challenge. **Transactions of the Association for Computational Linguistics**, v. 6, p. 317–328, 2018. ISSN 2307-387X.

KOK, S.; DOMINGOS, P. Statistical predicate invention. **ACM International Conference Proceeding Series**, v. 227, p. 433–440, 2007.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LAI, G. et al. RACE: Large-scale ReAding comprehension dataset from examinations. **EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings**, p. 785–794, 2017.

LAN, Y. et al. A survey on complex knowledge base question answering: Methods, challenges and solutions. **CoRR**, abs/2105.11644, 2021. Disponível em: <https://arxiv.org/abs/2105.11644>.

LAN, Y.; JIANG, J. Query graph generation for answering multi-hop complex questions from knowledge bases. In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, 2020. p. 969–974. Disponível em: <https://aclanthology.org/2020.acl-main.91>.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: TOURETZKY, D. S. (Ed.). **Advances in Neural Information Processing Systems 2**. Morgan-Kaufmann, 1990. p. 396–404. Disponível em: <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf>.

LIU, S. et al. A recursive recurrent neural network for statistical machine translation. **52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference**, v. 1, p. 1491–1500, 2014.

MCCRAE, J.; SPOHR, D.; CIMIANO, P. Linking lexical resources and ontologies on the semantic web with lemon. In: SPRINGER. **Extended Semantic Web Conference**. [S.l.], 2011. p. 245–259.

MELAMUD, O.; GOLDBERGER, J.; DAGAN, I. context2vec: Learning generic context embedding with bidirectional LSTM. **CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings**, p. 51–61, 2016.

MIKOLOV, T. et al. Extensions of recurrent neural network language model. **ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings**, p. 5528–5531, 2011. ISSN 15206149.

MILLER, A. et al. Key-value memory networks for directly reading documents. **arXiv preprint arXiv:1606.03126**, 2016.

MOHAN, A. T.; GAITONDE, D. V. A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks. n. April, 2018. Disponível em: <http://arxiv.org/abs/1804.09269>.

MOREIRA, S. **Rede Neural Perceptron Multicamadas**. 2018. Disponível em: <https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9>. Acesso em: 15 jul. 2020.

NACAXE, I. **Assistentes virtuais: Por que o varejo está apostando nessa tendência?** 2019. Disponível em: <https://www.ecommercebrasil.com.br/artigos/assistentes-virtuais-varejo-tendencia/>. Acesso em: 01 ago. 2020.

NATHANI, D. et al. Learning attention-based embeddings for relation prediction in knowledge graphs. **arXiv preprint arXiv:1906.01195**, 2019.

NGUYEN, T. et al. MS MARCO: A human generated MAchine reading COmprehension dataset. **CEUR Workshop Proceedings**, v. 1773, p. 1–10, 2016. ISSN 16130073.

OCHIENG, P. Parot: Translating natural language to sparql. **Expert Systems with Applications: X**, v. 5, p. 100024, 2020. ISSN 2590-1885. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2590188520300032>.

OLAH, C. **Understanding LSTM Networks**. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 15 jul. 2020.

OU, S. et al. Automatic question pattern generation for ontology-based question answering. In: **Flairs Conference**. [S.l.: s.n.], 2008. p. 183–188.

PEREIRA, M. J. et al. Chatbots' greetings to human-computer communication. **arXiv e-prints**, p. arXiv–1609, 2016.

RAJPURKAR, P.; JIA, R.; LIANG, P. Know what you don't know: Unanswerable questions for SQuAD. **ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)**, v. 2, p. 784–789, 2018.

RAJPURKAR, P. et al. SQuad: 100,000+ questions for machine comprehension of text. **EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings**, n. ii, p. 2383–2392, 2016.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: A core of semantic knowledge. In: **Proceedings of the 16th International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2007. (WWW '07), p. 697–706. ISBN 9781595936547. Disponível em: <https://doi.org/10.1145/1242572.1242667>.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning**: An introduction. 2. ed. Cambridge, MA: MIT Press, 2018. ISBN 0262039249.

SUTTON, R. S. et al. Policy gradient methods for reinforcement learning with function approximation. In: **In Advances in Neural Information Processing Systems 12**. [S.l.]: MIT Press, 2000. p. 1057–1063.

TOUTANOVA, K. et al. Representing text for joint embedding of text and knowledge bases. **Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing**, n. September, p. 1499–1509, 2015.

TRISCHLER, A. et al. NewsQA: A Machine Comprehension Dataset. p. 191–200, 2017.

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

WALLACE, R. S. The anatomy of a.l.i.c.e. In: EPSTEIN, R.; ROBERTS, G.; BEBER, G. (Ed.). **Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer**. Dordrecht: Springer Netherlands, 2009. p. 181–210. ISBN 978-1-4020-6710-5. Disponível em: <https://doi.org/10.1007/978-1-4020-6710-5_13>.

WALTER, S.; UNGER, C.; CIMIANO, P. A corpus-based approach for the induction of ontology lexica. In: SPRINGER. **International Conference on Application of Natural Language to Information Systems**. [S.l.], 2013. p. 102–113.

WEIZENBAUM, J. Eliza—a computer program for the study of natural language communication between man and machine. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 9, n. 1, p. 36–45, jan 1966. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/365153.365168>.

XIONG, W.; HOANG, T.; WANG, W. Y. DeepPath: A reinforcement learning method for knowledge graph reasoning. **EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings**, p. 564–573, 2017.

YANG, F.; YANG, Z.; COHEN, W. W. Differentiable learning of logical rules for knowledge base reasoning. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 2316–2325. ISBN 9781510860964.

YIH, W.-t. et al. Semantic parsing via staged query graph generation: Question answering with knowledge base. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Beijing, China: Association for Computational Linguistics, 2015. p. 1321–1331. Disponível em: <https://aclanthology.org/P15-1128>.

YIN, W. et al. Simple question answering by attentive convolutional neural network. In: **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers**. Osaka, Japan: The COLING 2016 Organizing Committee, 2016. p. 1746–1756. Disponível em: <https://aclanthology.org/C16-1164>.

YOU, J. et al. Graph convolutional policy network for goal-directed molecular graph generation. In: **Proceedings of the 32nd International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2018. (NIPS'18), p. 6412–6422.

YOU, J. et al. Graphrnn: A deep generative model for graphs. **CoRR**, abs/1802.08773, 2018. Disponível em: <http://arxiv.org/abs/1802.08773>.

YOUNG, T. et al. Recent trends in deep learning based natural language processing [Review Article]. **IEEE Computational Intelligence Magazine**, IEEE, v. 13, n. 3, p. 55–75, 2018. ISSN 15566048.

ZHANG, S. et al. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. 2018. Disponível em: <http://arxiv.org/abs/1810.12885>.

ZHANG, Z.; CUI, P.; ZHU, W. Deep Learning on Graphs: A Survey. **IEEE Transactions on Knowledge and Data Engineering**, v. 14, n. 8, p. 1–1, 2020. ISSN 1041-4347.

ZHOU, L. et al. The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. **Computational Linguistics**, v. 46, n. 1, p. 53–93, 03 2020. ISSN 0891-2017. Disponível em: <https://doi.org/10.1162/coli_a_00368>.

ZOU, L. et al. Natural language question answering over rdf: A graph data driven approach. In: **Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2014. (SIGMOD '14), p. 313–324. ISBN 9781450323765. Disponível em: <https://doi.org/10.1145/2588555.2610525>.