Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma aplicação para perguntas e respostas no domínio de inteligência computacional

William Gustavo Queiroz Simião

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Ladeira

Coorientador
Prof. Dr. Claus de Castro Aranha

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma aplicação para perguntas e respostas no domínio de inteligência computacional

William Gustavo Queiroz Simião

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Jorge Carlos Lucero
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
CIC/UnB

Prof. Dr. Claus de Castro Aranha
University of Tsukuba

Prof. Dr. Edison Ishikawa
Coordenador do Bacharelado em Ciência da Computação

Brasília, 08 de Março de 2019

Dedicatória

Dedico este trabalho à Rita, ao Alcides e à Cibelle, pelo amor e suporte sem igual.

Agradecimentos

Agradeço a UnB pela experiência de aprendizado, aos meus orientadores pela ajuda, aos professores pela diposição em ensinar e aos meus amigos pelo apoio.

Resumo

Estudantes de graduação tem como hábito a busca de um termo desconhecido em seu vocabulário acadêmico em mecanismos de busca como o *Google*, para então selecionar alguma página que contenha a definição do termo, que não necessariamente será precisa. O objetivo primário desse trabalho é desenvolver uma aplicação que permita ao usuário explorar o conteúdo de um domínio por meio de perguntas escritas em linguagem natural. Como domínio de prova de conceito escolhemos a área de Inteligência Computacional. Propomos responder à perguntas sobre temas do domínio através da seleção automática do texto relevante ao assunto da pergunta, utilizando o serviço de mensagem instantânea *Telegram*, a ferramenta Babelfy para reconhecimento entidades nomeadas, uma base de textos do domínio e o modelo *Bi-directional Attention Flow* (BiDAF) para predição da resposta. Adicionalmente abordamos um problema muito similar, buscar informações pontuais em documentos extensos, pois não se conhece previamente a localização da informação desejada no documento. Aliada a proposta ao problema de exploração de conteúdos, propomos o pré-processamento e anotação manual do documento, como passos prévios à geração de uma aplicação de busca das informações desejadas através de perguntas em linguagem natural. Os resultados obtidos indicam que o uso de reconhecimento de entidades nomeadas para determinação do assunto de uma frase possui limitações como: múltiplas entidades e problemas no reconhecimento de termos técnicos. Além disso, o modelo BiDAF apresentou respostas corretas e concisas para perguntas simples e factuais, mas exibiu respostas incompletas ou insatisfatórias para perguntas complexas.

Palavras-chave: Processamento de Linguagem Natural, Pesquisa de Respostas, Inteligência Computacional, Chatbot

Abstract

Undergrad students have a habit of searching for an unknown term to their academic vocabulary in search engines such as *Google*, then select some page that contains the definition of the term, which will not necessarily be accurate. The primary purpose of this research is to develop an application that allows the user to explore the contents of a domain through questions written in natural language. As a proof-of-concept domain we chose the area of Computational Intelligence. We propose to answer the questions about domain themes by automatically selecting the text relevant to the subject of the question, using the *Telegram* instant message service, the Babelfy tool for recognition of named entities, a domain text base and the *Bi-directional Attention Flow* (BiDAF) for prediction of response. In addition, we approach a very similar problem, to seek specific information in extensive documents, we do not know the location of the information desired in the document. Allied to the proposal to the problem of content exploration, we propose the pre-processing and manual annotation of the document, as steps prior to the generation of a application for searching the information desired through natural language questions. The results indicate that the use of recognition of entities named to determine the subject of a sentence has limitations such as: multiple entities and problems in the recognition of technical terms. In addition, the BiDAF model presented correct and concise answers to simple and factual questions, but exhibited incomplete or unsatisfactory answers to complex questions.

Keywords: Natural Language Processing, Question Answering, Comptacional Inteligence, Chatbot

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo descrevemos qual o problema a ser resolvido, a relevância dele e a proposta de solução.

1.1 Definição do objeto de estudo

O uso de um sistema de *Question Answering* para exploração de um novo assunto se diferencia de uma busca em *search engines*, como o *Google*, devido o retorno de uma resposta em linguagem natural, ao invés de uma coleção de documentos que podem ser relevantes para a pergunta.

Uma aplicação que fornecesse respostas diretas e específicas sobre o conteúdo de um domínio permitiria que o processo de familiarização com um novo assunto ocorresse de forma mais interativa. A proposta principal deste trabalho é desenvolver uma aplicação que permita esta exploração dinâmica do conteúdo para domínios fechados por meio de perguntas em linguagem natural, como de prova de conceito escolhemos o domínio de Inteligência Computacional. Complementarmente, propomos uma solução para um problema similar: desenvolver uma aplicação capaz de responder perguntas de um domínio escolhido pelo usuário, por meio da seleção de um documento PDF.

As soluções voltadas a responder perguntas sobre um conjunto de assuntos específicos são ditas de domínio fechado, enquanto as soluções que buscam responder perguntas de forma geral são ditas de domínio aberto.

Mas é importante atentar para diferença entre domínio fechado e a definição de hipótese do mundo fechado, termo comum no assunto de Inteligência Artificial, que se refere à premissa de que o conhecimento não presente na base do sistema deve ser assumido como falso. Logo, a adoção domínio fechado explicita que a fim de obter-se uma resposta válida o sistema requer uma pergunta pertencente ao conhecimento do domínio para o qual foi

projetado. Assim, se esse requisito não for satisfeito o resultado pode ser uma resposta equivocada ou um erro de ausência de conhecimento na base da aplicação.

Portanto, na etapa de pesquisa bibliográfica focou-se em trabalhos no tema de *Question Answering* voltadas à domínios fechados, visto que o domínio aberto diverge de nosso objetivo.

1.2 Justificativa

A vontade de contribuir para processo de aprendizagem autônoma vivenciado pela maioria dos estudantes de um novo conteúdo ou área do conhecimento foi o principal estímulo para a escolha do tema. Entretanto, o objetivo da ferramenta proposta na primeira etapa não é servir de fonte ou ensinar conceitos e definições aos estudantes, mas sim propiciar a consulta de sobre o conteúdo estudado de maneira mais dinâmica e refinada.

Ao encarar o desafio de aprender um novo conteúdo e familiarizar-se com novos conceitos, com os quais ideias mais complexas são construídas, é uma das maiores barreiras a se superar. A possibilidade de perguntar sobre estes conceitos em linguagem natural pode ajudar a consolidar o conhecimento básico requerido para o aprofundamento do estudo.

O domínio de Inteligência Computacional foi escolhido como prova de conceito devido sua relevância para cursos de graduação em computação. O domínio abrange variados conceitos como: Algoritmos Genéticos, Visão Computacional, Redes Neurais e etc. Como a ferramenta deve ter uma interface primariamente textual com o usuário, o idioma escolhido deve facilitar esta interação, logo o português seria mais indicado, todavia, consideramos dois fatores para a escolha do idioma: o público alvo e os ganhos da pesquisa com o uso do inglês:

- O público alvo da proposta primária deste trabalho, estudantes de graduação da área tecnológica, que em grande parte já possuem experiência com o inglês por demandas do próprio curso.
- A exploração da literatura no campo de *Question Answering* mostrou que em grande parte dos trabalhos de ponta investem no inglês como único ou principal idioma. Assim grande parte de algoritmo, bases de dados e *benchmarks* encontram-se neste idioma.

Assim, visto que o uso do inglês não representa um empecilho para o publico alvo e para fazer uso da maior variedade de recursos disponíveis para o inglês (bibliotecas e ferramentas) optamos por usar este idioma como interface entre estudante e aplicação.

A vontade de tornar a aplicação desenvolvida na primeira etapa, flexível para domínios retratados em documentos existentes, motivou a segunda etapa do projeto. Para situa-

ções onde uma informação particular está encoberta dentro de vários parágrafos de um documento, a possibilidade de simplesmente perguntar pela informação seria um grande facilitador.

1.3 Estrutura do documento

- Capitulo 2 : aborda os principais tópicos de estudo usados no desenvolvimento da pesquisa, fornecendo um embasamento para a compreensão dessa pesquisa.
- Capitulo 3 : descreve de forma gradual os passos e decisões tomadas para atingir os objetivos da pesquisa, bem como as dificuldades apresentadas durante o processo.
- Capitulo 4 : apresenta a análise da aplicação concluída e detalhamento de seu modo de uso para usuários.
- Capitulo 5 : apresenta as conclusões e as perspectivas de trabalho futuro.

Capítulo 2

Fundamentação Teórica

Este capítulo contém o embasamento teórico necessário para o desenvolvimento da pesquisa, definimos as técnicas usadas, deixando claro seu propósito e relevância para essa pesquisa.

São introduzidos os termos usados com frequência ao longo dos próximos capítulos, além de descrever o funcionamento dos algoritmos estudados, para aplicar na solução do problema de responder perguntas sobre um domínio fechado.

2.1 Estado da Arte

A síntese da arquitetura de um sistema de QA (*Question Answering*), elaborada por Allan e Haggag [1], a partir da análise dos sistemas citados na literatura da área, apresenta três módulos típicos dos sistemas de QA:

- Processamento da Pergunta: o propósito deste módulo é obter maiores detalhes sobre a pergunta proposta de forma a identificar o tipo de conhecimento necessário para respondê-la. Técnicas como *part-of-speech tags* e classificação de perguntas são utilizadas nos artigos de Derici *et al*, [2] e de Cuteri [3] para identificação do foco da pergunta.
- Processamento do Documento: este módulo é implementado apenas por aplicações que realizam extração de informações em documentos para obter dados para formulação da resposta, como é o caso da pesquisa de Lende e Raghuwanshi [4] . Em alternativa a extração de informações de documentos, as outras aplicações fazem uso das informações obtidas na etapa de *Question Processing* para mapear a pergunta feita aos termos presentes na base de conhecimento da aplicação, quando estejam associados a conceitos ou informações úteis para responder a pergunta.

- Processamento da Resposta: engloba a funcionalidade de refinar a resposta em linguagem natural fazendo uso da informação obtida na etapa anterior e responder de forma direta com a maior especificidade possível a pergunta de entrada. Suponha um sistema de perguntas e respostas cujo domínio seja questões meteorológicas locais, como o apresentado no artigo de *et al* [5], assim se a pergunta de entrada for: "*Will it rain today ?*"ou "*Will I need an umbrella today ?*", ambas as perguntas requerem o mesmo conhecimento (se o dia estará chuvoso ou não) mas as respostas apropriadas a cada uma delas diferem. Em cenários onde a interface do sistema é um diálogo, deseja-se que a aplicação formule uma resposta especificamente para a questão, ao invés de retornar uma parte de texto que contenha a informação relevante para a pergunta. Claramente esta funcionalidade requer um nível razoável de inferência sobre os termos, além de conhecimento modelado pelo sistema. Dessa forma, aplicações que utilizam bases de conhecimento estruturado, isto é, formatado como tabelas num banco de dados relacional, tem essa etapa facilitada.

Nos artigos de Derici, Cuteri, Chung *et al* e de Lende e Raghuwanshi foram evidenciadas correspondências com a arquitetura de três módulos, descrita no *survey* de Allan e Haggag [1]. Essas publicações abordam soluções para QA de forma tradicional, fazendo uso de heurísticas sobre o domínio em questão [3], análise sintática e outras práticas frequentes no campo de Processamento de Linguagem Natural. No entanto, os trabalhos mais citados dos últimos anos vêm usando algoritmos de Aprendizado de Máquina treinados com dados de perguntas e respostas correspondentes para fornecer a resposta de uma pergunta, como as publicações de Vikas *et al* [6] e de Kumar *et al*[7]. Seus dados de treinamento, de forma geral, são compostos por conjuntos que incluem: uma pergunta, um texto sobre o tema da questão e a correspondente resposta. Esses trabalhos não processam a informação da mesma maneira, portanto não se encaixam na arquitetura de três módulos.

Entretanto, semelhanças com a arquitetura de módulos descritas podem ser percebidas nos componentes internos de algoritmos de aprendizado de máquina, como o modelo Bi-DAF [8] desenvolvido por Seo *et al*, onde há etapas do processo responsáveis por identificar quais partes do texto contendo a resposta são mais relevantes de acordo com a pegunta. Ou seja, análogo a componente de processamento da pergunta dessa arquitetura.

No *survey* de sistemas de QA [1] também são analisados e comparados trabalhos que descrevem implementações de sistemas que dependem fortemente de técnicas de NLP (do Inglês *Natural Language Processing*) e de exploração de informação estruturada em uma base de conhecimento (KB, do inglês *Knowledge Base*). Nenhum deles utiliza métodos baseados em grande quantidades de dados para elaborar sua solução.

As abordagens, não baseadas em algoritmos de aprendizado de máquina, utilizam uma base de conhecimento elaborada para um domínio específico, isto é, o conhecimento sobre um assunto delimitado é organizado de forma estruturada ou faz-se uso de ontologias muito mais abrangentes, que podem abarcar mais de um domínio. Neste último caso, essas tendem a ser menos relacionadas e mais flexíveis, já que em geral seu conhecimento é obtido da Internet por meio da exploração de formas de representação do conhecimento como Web Semântica ou de *crawlers*, como feito por Chung *et al* [5].

As bases de conhecimento estruturado possuem alto custo para elaboração e manutenção, visto que todo novo conhecimento deve ser definido de maneira formal, estabelecendo sua relação com outros conceitos relevantes já contidos na base. Outra limitação do uso de bases de conhecimento estruturado é a cobertura de perguntas, pois questões sobre temas não definidos na KB não podem ser respondidas.

Todavia existem vantagens na adoção de uma KB estruturada para desenvolvimento de um sistema de QA, como a possibilidade de utilizar regras de inferência, na elaboração de resposta à domínios específicos como feito por Chung *et al* [5]. O escopo reduzido de assuntos das perguntas favorece a precisão das respostas [4], comparado a sistemas de QA que se propõem a responder quaisquer perguntas.

2.2 Chatbots

O termo chatbot é usado para referir a diferentes softwares, mas em geral são aplicações que utilizam processamento de linguagem natural para fornecer serviços através de uma interface similar a uma conversa com alguém numa aplicação de troca de mensagens, tornando o usuário mais engajado na experiência de uso.

Diversas empresas investem em tecnologias similares à proposta do chatbot, apesar da Mídia e o do *Marketing* das empresas terem preferido o termo "assistentes virtuais". Mas a proposta é análoga: fornecer serviços por meio de uma interface facilite a interação com o usuário.

Dentre os benefícios diretos do uso de chatbots, em detrimento de interfaces tradicionais, estão: menor curva de aprendizagem com a interface; múltiplos serviços, que podem ser ofertados através de uma interface em comum; e favorecer o engajamento de clientes, que passam mais tempo com troca de mensagens do que em mídias sociais [9]. Assim, visto que a entrada do usuário é primariamente escrita, a ferramenta desenvolvida nesta pesquisa tem muito a se espelhar na interface de chatbots.

2.3 *Question Answering*

Responder perguntas em linguagem natural sobre o conhecimento presente em um banco de dados, texto ou uma coleção de documentos é uma tecnologia com diversas aplicações possíveis. Onde for necessário obter um fragmento de informação contido em um conjunto textual muito maior, o custo de busca desse fragmento poderia ser reduzido drasticamente comparado à busca feita por qualquer pessoa.

QA está relacionada à outros campos de pesquisa, como: NLP, *information retrieval* e *machine comprehension*, visto que compartilham o objetivo de desenvolver um agente inteligente para diálogo [1].

De acordo com Burges, *Machine comprehension* (MC) é quando uma maquina entende um texto e é capaz de dar uma resposta correta, considerando a opinião da maioria dos falantes do idioma, sem que esta possua informações desnecessárias para a resposta da pergunta.[10]

Information Retrieval (IR) é um campo que se ocupa em recuperar, de um conjunto de documentos, apenas os mais relevantes para uma determinada informação. Vale destacar que no contexto de QA, a pergunta faz o papel dessa informação [11]. Em sistema que buscam responder perguntas sobre qualquer domínio, o uso de técnicas *IR* é útil para refinar a busca da resposta à alguns documentos ou mesmo parágrafos, [4]. Esses sistemas que visam cobrir perguntas sobre diversas áreas do conhecimento são classificados como *Open-domain*, isto é, não se restringem a um único domínio para as peguntas que é capaz de responder. Já os sistemas que focam em um só domínio, buscando aumentar a precisão, são chamados de *Close-domain*.

Sistema de QA *Open-domain* tem um objetivo muito ambicioso, visto que qualquer assunto pode ser tema de uma pergunta. Então os sistemas com essa proposta não podem depender apenas de bases de conhecimento estruturado, visto que grande parte dos dados são gerados continuamente no âmbito corporativo e encontram-se não estruturados [12].

Alguns dos algoritmos de MC [8][7] são modelados para solucionar um problema similar à responder peguntas genéricas, porém significativamente mais simples, como encontrar a resposta a uma pergunta em um trecho de texto. Desta forma esses algoritmos delimitam a porção de um texto que contém a resposta para uma pergunta feita em linguagem natural.

Para responder uma pergunta sobre determinado texto, existem passos intermediários entre identificar qual a informação que é requerida pela pergunta e encontrá-la no texto, por exemplo: a reformulação da pergunta, seleção da fonte para extração da resposta. Informações externas ao texto, inferências e outras sutilezas intrínsecas à linguagem dificultam a obtenção de respostas satisfatórias para perguntas complexas. Por isso, os modelos atuais estão longe da definição proposta por Burges [10]. Contudo, esses algorit-

mos podem apresentar resultados promissores quando treinados sobre bases de dados com exemplos representativos de perguntas típicas como: questões que demandam indução a partir de fatos e perguntas que requerem contagem de elementos [13].

2.4 Reconhecimento de Entidades Nomeadas

O processo de identificação de entidades num texto, comumente abreviada para NER (do inglês *Named entity recognition*), é frequentemente usado em diversos campos de NLP [14], pois permite estabelecer quais palavras e trechos do texto tem maior relevância e até classificar a entidade em categorias que forneçam mais informações sobre o cerne da sentença analisada.

Essas classificação das entidades são chamadas *Tags*, que correspondem a um conjunto de categorias que depende da biblioteca usada. A Figura 2.1 apresenta *Tags* de quatro entidades em uma frase, esse exemplo foi feito pela CoreNLP [15], que faz parte da biblioteca NLTK.

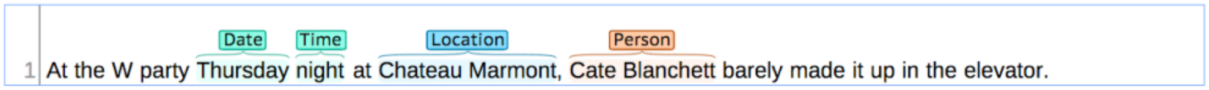


Figura 2.1: Exemplo de reconhecimento de entidades pela CoreNLP .

O reconhecimento de entidades nomeadas é usado, por exemplo, na geração de informação estruturada a partir de textos, bem como auxilia no campo de QA com a identificação de potenciais focos para que possam sugerir qual a informação requisitada numa pergunta.

O termo entidade pode levar alguém não familiar com o campo de pesquisa a pensar que apenas substantivos podem servir como entidades. Porém a definição usada por uma das bibliotecas utilizadas nessa tarefa, NLTK (*Natural Language Tool Kit*), abrange também termos como datas, horários, quantidades e lugares. Inclusive o NLTK após identificar as entidades, também as classifica nas categorias: Organização, Pessoa, Lugar, Data, Horário, Dinheiro, Porcentagem, instalação, GPE (*Geopolitical Entities*). Por isso é frequente referir-se a ferramentas que realizam essa identificação e classificação de entidades como *Named Entity Taggers*.

Desta forma, é possível ver a tarefa de um NER *Tagger* como duas sub-tarefas: identificar entidades e classificá-las. Em geral, ambas as sub-tarefas são tratadas por algoritmos que são treinados a partir de grandes quantidades de exemplos para identificar e categorizar corretamente as entidades numa frase. Outra abordagem usada é o uso de regras fixas baseadas em padrões de formação. Contudo, ambas as abordagens são dependentes

do idioma, logo algumas línguas carecem em variedade de ferramentas para a tarefa de reconhecimento de entidades. No caso do NLTK, a implementação proposta por Manning *et al* [15] foi encorporada à plataforma e é treinada sobre a base de dados CoNLL-2003 [16], apresentando bons resultados para pessoas, organizações e lugares.

Apesar de exibir bons resultados para essas três categorias, o desempenho do classificador do NLTK cai significativamente quando palavras técnicas ou de domínios específicos fazem o papel das entidades da frase, cenário que faz parte da esfera desse trabalho.

Em teste preliminares identificou-se duas ferramentas que tiveram melhor desempenho para entidades não pertencentes as categorias de pessoas, organizações e lugares foram: Babelfy [17] e o NER *Tagger* do grupo AllenNLP [18].

2.4.1 NER AllenNLP e Babelfy

Este modelo de reconhecimento de entidades desenvolvido pelo Instituto Allen, descrito por Peters *et al* [19], teve média da métrica F1 acima de 90%, superando os modelos comparados no mesmo artigo de Peters *et al*. Assim como o CoreNLP, também foi treinado usando a base de dados CoNLL 2003 para tarefa de reconhecimento de entidades.

Babelfy é uma ferramenta desenvolvida pela mesma equipe que desenvolve o Babelnet, uma rede que relaciona termos e sua definições além de ser usada para desambiguação de termos [20]. Por sua relação com o Babelnet, que possui enfoque na semântica, o Babelfy obteve melhor performance em teste que exploram entidades que não são frequentes em textos de âmbito geral, uma vantagem tanto para domínio de Inteligência Computacional quanto para o domínio de documentos PDF.

No próximo capítulo são descritas as particularidades das duas ferramentas e uma bateria de testes comparativos, pelo qual foi determinada a mais apropriada para o desenvolvimento deste trabalho.

2.5 Algoritmos de aprendizado para *Question Answering*

Abordar o problema de QA com algoritmos de aprendizado de máquina permite fazer uso de grandes quantidades de dados gerados por usuários de serviços online, como redes sociais e avaliações de produtos de lojas online. Mas existem diferentes modelos que propõem variadas soluções. Nesta seção, vamos analisar duas implementações amplamente citadas: o modelo BiDAF e o *Dynamic Memory Networks* (DMN). Discutiremos ainda a proposta de cada e exploraremos suas particularidades, a fim de escolher o mais cabível a uma ferramenta voltada a responder perguntas de domínios técnicos.

Antes de explorar os algoritmos de aprendizados propriamente, é preciso explicitar uma técnica que fundamenta suas implementações: a incorporação de palavras.

2.5.1 Incorporação de palavras

Traduzido do termo *Word Embedding*, é uma técnica para representação de palavras na forma de vetores e pode ser usada para relacionar palavras de acordo com a distância entre os vetores que as representam. Essa técnica apresenta duas principais vantagens sobre a alternativa mais simples usadas para modelar palavras, os N-grams. Primeiramente, ela permite estabelecer a similaridade entre as palavras. Visto que as palavras não são conceitos atômicos [21], a possibilidade de medir a proximidade semântica entre elas representa um grande potencial para sistemas de processamento de linguagem natural. A segunda vantagem desta abordagem é a escalabilidade do método comparada ao modelo de N-grams, que modela a linguagem como sequências de tamanho N compostas por N palavras contíguas de um vocabulário.

Ao representar as palavras na forma de vetores, abre-se a possibilidade de aplicação de operações algébricas sobre os vetores. No exemplo apontado em [21], ao executar a operação *vector('King') - vector('Man') + vector('Woman')* obteve-se um vetor próximo daquele representante da palavra *Queen*. A Figura 2.2, apresentada em [22], ilustra como essas operações podem revelar informações semânticas que podem ajudar em tarefas de NLP, a exemplo de inferências sobre QA.

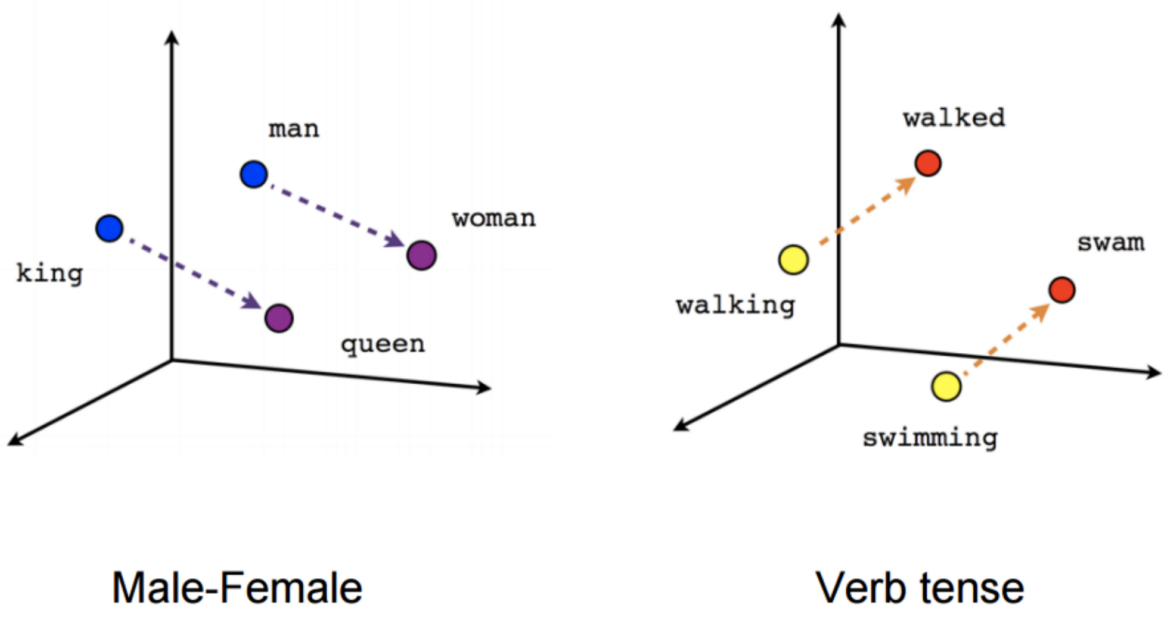


Figura 2.2: Exemplo de proximidade espacial entre vetores com relação semântica.

Os dois algoritmos de aprendizado voltados à QA, que serão descritos na próxima seção, fazem uso de modelos de *Word Embedding* que são treinados com *corpus* diversos, como artigos do Wikipédia e publicações do Twitter. Tanto o modelo BiDAF quanto o DMN utilizam o algoritmo GloVe proposto por Pennington *et all* [23].

2.5.2 *Dynamic Memory Networks*

É um modelo analisado por Kumar *et al* [7], onde é proposto que problemas como: tradução entre idiomas e análise de sentimento, podem ser modelados como um problema de QA. Ou seja, o algoritmo pode ser treinado para responder não só perguntas sobre o conteúdo de um texto de entrada mas também perguntas como: *What's the french translation ?* e *"What's the sentiment"*.

O algoritmo DMN é treinado sobre a base de dados bAbI do Facebook [13] proposta por Weston *et al*, que explora categorias de tarefas frequentes no desafio de responder perguntas sobre textos. Cada categoria explora uma habilidade necessária para responder perguntas em diversas situações. Dentre as 20 categorias, algumas são: *Yes or No Question*, *Two Supporting Fact*, *Indefinite Knowledge*, *Time Reasoning*, *Basic Deduction* e *Size Reasoning*. A Tabela 2.1 contém exemplos de duas categorias baseados no artigo de Kumar *et al* [7]:

Task 2: Two Supporting Facts	**Task 10: Indefinite Knowledge**
John is in the playground.	John is either in the classroom or the playground.
John picked up the football.	Sandra is in the garden.
Bob went to the kitchen.	Is John in the classroom? A:maybe
Where is the football? A:playground	Is John in the office? A:no

Tabela 2.1: Exemplo de categorias de tarefa abordadas na base de dados bAbI

A arquitetura do modelo é composta pelos módulos: *Input Module*, *Question Module*, *Episodic Memory Module*, *Answer Module*. Através de um mecanismo de atenção o *Episodic Memory Module* escolhe em quais partes do texto de entrada deve focar. Esse processo ocorre de forma interativa e a cada interação proporciona novas informações sobre o texto.

Os resultados apresentados no artigo de Kumar *et al* [7] mostram a ótima performance do modelo, atingindo acurácia maior que 95% em 18 tarefas.

2.5.3 *Bi-directional Attention Flow*(BiDAF)

É um modelo proposto por Seo, que deve receber uma pergunta e um contexto como entrada para gerar uma predição de resposta. O BiDAF emprega um mecanismo de atenção cujo objetivo é estabelecer a similaridade entre as palavras do contexto e da pergunta, determinando assim as palavras mais relevantes para responder à pergunta. É

treinado sobre a base de dados SQuAD que, assim como o bAbi, é composta por triplas (texto, pergunta, resposta), a Figura 2.3 obtida do site oficial do SQuAD [24] mostra um exemplo. Entretanto, o SQuAD tem dois grandes diferenciais. O primeiro é a elaboração a partir de artigos selecionados do Wikipédia, logo, o texto de entrada, ou contexto, são parágrafos desses artigos. O segundo, é que a resposta ou é composta de um trecho do contexto da tripla ou é vazia pela ausência de informação no contexto.

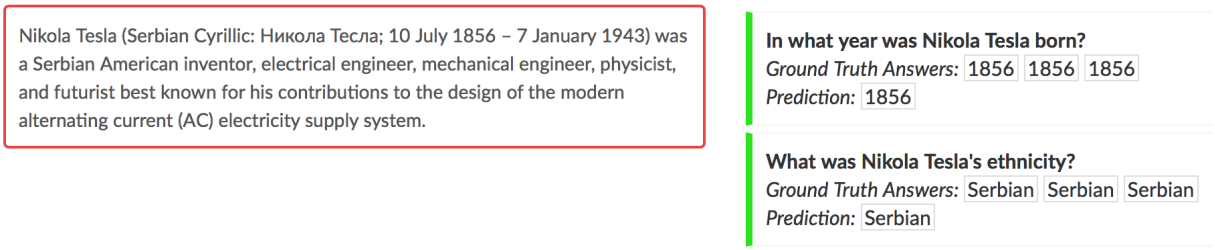


Figura 2.3: Exemplo da base de dados SQuAD.

A arquitetura do modelo é organizada em camadas e os autores apontam o seu diferencial como sendo dois fatores principais:

- A camada de atenção, que decide em quais partes da pergunta focar, calcula a atenção a cada passo do processo, o que reduz a perda de informação precoce, como ocorre em padrões de atenção anteriores.
- A computação da atenção é *memory-less*, isto é, em um instante de tempo a atenção é função apenas da pergunta e do contexto.

O modelo DMN usa um mecanismo interativo de atenção, ou seja, o cálculo da atenção dependia do seu valor anterior. Isso mostra como os modelos têm propostas diferentes.

Nos experimentos reportados no artigo de Seo *et al*[8] obteve-se resultados muito bons sobre o conjunto de teste do SQuAD, onde a predição feita pelo modelo BiDAF é comparada com a resposta verdadeira. As duas métricas usadas para a avaliação foram : a métrica *Exact Match*, na qual o modelo obteve 73.3; e a métrica F1, na qual obteve 81.1. Ambas as métricas são referências na avaliação de modelos de QA.

Capítulo 3

Metodologia

O primeiro objetivo deste trabalho foi desenvolver uma ferramenta para gerar resposta à perguntas sobre Inteligência Computacional. Posteriormente o estendemos objetivo para que o domínio fosse determinado pelo usuário a partir de um documento PDF, assim o desenvolvimento deste trabalho passou por duas fases. Na primeira, desenvolvemos os requisitos propostos para a ferramenta voltada para exploração do domínio de Inteligência Computacional. Na segunda, exploramos o potencial de criar uma ferramenta flexível cujo domínio fosse decidido pelo usuário. Ao invés de ser focada em responder perguntas sobre um domínio fixo e implementar funções que auxiliassem a contínua exploração da base de conhecimento, o objetivo passou ser responder perguntas sobre um documento no formato PDF. Dessa forma, novas demandas de implementação surgiram, por exemplo: leitura e segmentação do documento e criação da nova base de conhecimento.

Apesar de muitas mudanças necessárias entre as etapas, o cerne da proposta é o mesmo: utilizar uma associação entre palavras-chave e textos. As palavras-chave são identificadas na pergunta recebida do usuário e nos textos que compõem a base de conhecimento, referidos aqui por diante como contextos, os quais são usados para extrair a resposta da pergunta.

Na etapa do domínio de Inteligência Computacional, foram cobertos apenas alguns conceitos populares e introdutórios, e na segunda etapa foi utilizado como documento para validação um guia para pesquisadores bolsistas da JSPS (*Japan Society for the Promotion of Science*).

A seguir descrevemos partes comuns à ambas as fases e nas seções seguintes detalhamos as diferenças de implementação e escolhas específicas de cada etapa, ressaltando a causa dessas diferenças.

3.1 Escolha de algoritmos

No Capítulo 2 apresentamos dois algoritmos treinados com exemplos de triplas (pergunta, contexto e resposta) para identificar a resposta a partir de uma pergunta e um contexto. Também apresentamos ferramentas para identificação de entidades nomeadas em uma sentença, que ajudam a determinar o assunto da frase. A ferramenta desenvolvida, tanto nesta etapa quanto na que será discutida a diante, busca aliar as duas categorias de algoritmo para ampliar a gama de perguntas que podem ser respondidas, selecionando automaticamente o contexto de acordo com as entidades identificadas na pergunta.

Nesta seção, explicitamos as escolhas feitas para as duas categorias de algoritmo de aprendizado de máquina que compõem a ferramenta proposta, bem como a motivação por trás destas escolhas.

3.1.1 Algoritmo para NER

Esta tarefa é crucial para a identificação da resposta, pois é o primeiro passo deste processo. Assim, se o produto desta etapa for equivocado, todo o restante da sequência de ações realizadas (reescrita da pergunta, seleção do contexto, separação do trecho resposta) não terão como minimizar o erro cometido neste momento inicial. Os passos seguintes dependem tanto do reconhecimento de entidades sobre a pergunta, pois é neste onde essencialmente ocorre a escolha de qual será considerado o tema central da pergunta.

O algoritmo que realiza o reconhecimento de entidades deve ser capaz de reconhecer múltiplas entidades em uma frase, bem como uma variedade ampla o suficiente para cobrir o domínio sobre o qual a ferramenta se propõem a atuar. Várias bibliotecas com a função de reconhecimento de entidades nomeadas foram consideradas, como a *Open Calais* [25], um produto robusto, porém voltado a análise de documentos; dentre os serviços disponibilizados pelo Grupo *LX-Center* [26], o reconhecedor de entidade nomeadas só era acessível por meio de uma demostração no site do grupo; e *Rembrandt* [27], que não identificou corretamente entidades do domínio de Inteligência Computacional. Após os testes eliminatórios para verificar se a biblioteca era adequada para reconhecimentos de entidade de domínios técnicos, como o de Inteligência Computacional, chegou-se a duas alternativas cabíveis: o Bebelfy [17] e o NER do AllenNLP, biblioteca do Instituto Allen [18].

O Babelfy é uma API (*Application programming interface*) de reconhecimentos de conceitos e entidades nomeadas proposta pelos mesmo autores do BabelNet, uma API com uma proposta mais ampla de ser um dicionário enciclopédico multi-idiomático junto a uma rede semântica. São oferecidas versões pagas do Babelfy, entretanto o número de requisições é o único limitante da versão gratuita, 1000 requisições diárias. Este limite não

é um fator impeditivo para uso do Babelfy no caso deste trabalho visto que a princípio focamos na qualidade das resposta fornecidas e não no número de usuários. Contudo, a API possui um limite referente ao número de caracteres, são permitidos 5000 caracteres por requisição realizada. Este limite sim, trás impactos para o plano de uso API, visto que o reconhecimento de entidade em porções largas de texto será uma necessidade na segunda etapa deste trabalho, onde capítulos inteiros de um documento PDF passarão por uma análise.

O serviço do Babelfy pode ser acessado por meio de um API com interface para a linguagem Java ou por meio de uma API HTTP. A primeira forma de acesso permite tratar o retorno da API diretamente como objetos Java. Porém, a segunda permite o uso de qualquer linguagem para realizar a requisição, que retorna um arquivo JSON com as análises feitas sobre a frase fornecida como parâmetro. Esse arquivo contém informações que incluem a posição de início e fim de cada termo da frase identificado como "*Concept*" ou "*Named Entity*". Esses termos são usados na documentação para diferenciar entidades mais abstratas como *Artificial Intelligence* de entidades mais concretas como Allan Turing. O exemplo da Figura 3.1, extraido do site do Babelfy [17] ilustra o reconhecimento de termos em uma frase.

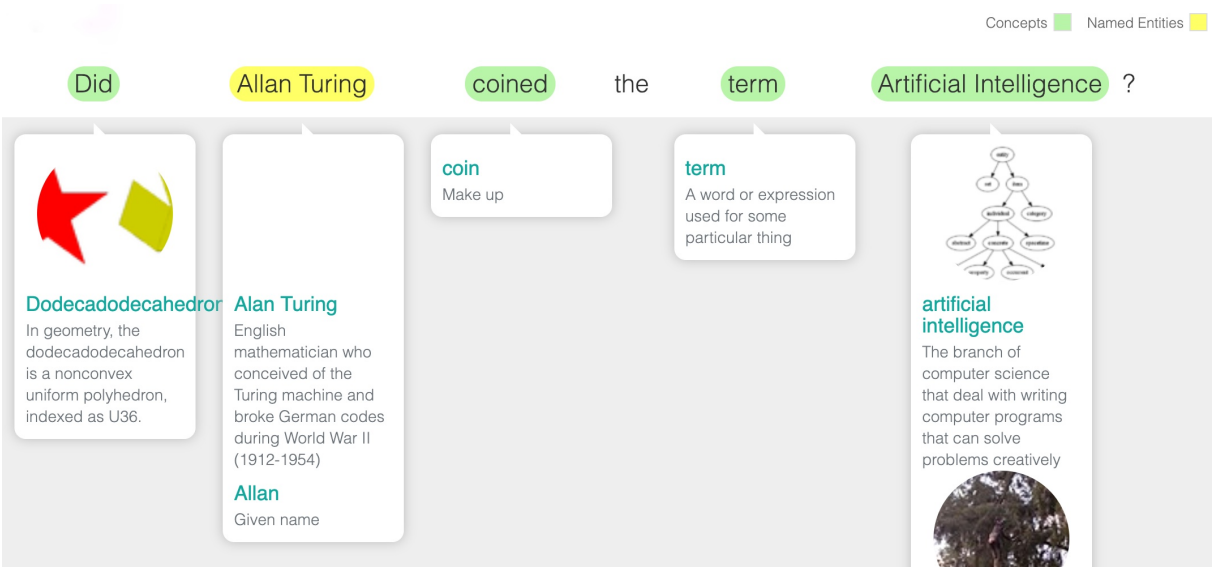


Figura 3.1: Exemplo de reconhecimento de entidades nomeadas pela biblioteca Babelfy.

O inconveniente do uso do Babelfy está relacionado ao que é considerado um "*Concept*", pois a definição usada engloba tanto verbos quanto os termos que são de interesse para o objetivo de identificar o tema central da pergunta. Para remediar a inclusão de verbos na lista de potenciais entidades da frase, foi realizada a remoção dos verbos da pergunta antes de passá-la como parâmetro para a API. Para essa identificação e remo-

ção dos verbos usamos o tokenizador e *Tagger* de POS (do inglês *Parts of Speach*) da biblioteca NLTK.

Com esta solução, o conjunto de entidade retornado após o processamento do Babelfy, que incluem tanto *"Concepts"* quanto *"Named Entities"*, possui uma quantidade menor de termos que não seriam usados como tema central da pergunta. Todavia, essa solução não impede que palavras como *"term"* na Figura 3.1 sejam considerados como possíveis focos da pergunta.

Uma alternativa ao Babelfy que foi o identificador de entidades nomeadas disponibilizado pelo instituto Allen para Inteligência Artificial através de uma biblioteca para a linguagem *Python*. Esta solução tem a vantagem de não requerer acesso a servidores e não possuir limites de uso. Assim como o Babelfy, a sua incorporação ao projeto requer pouco código e é bem direta. Desta forma partimos para teste comparativos, a fim de decidir a melhor solução para este trabalho.

A bateria de testes executados consistiu de perguntas em inglês sobre os tópicos do domínio de Inteligência Computacional. As perguntas elaboradas foram divididas em dois grupos, aqueles com apenas uma entidade no texto da pergunta e aqueles com duas entidades. Além de disso, buscou-se explorar nos testes alguns aspectos pertinentes sobre as perguntas:

- Inclusão de termos usados fora do domínio de Inteligência Computacional sem o sentido de uma entidade, como *generation*.
- Entidade constituídas de 1, 2 e 3 palavras.
- Uso de letra maiúscula e minúscula no início de entidades.

A Tabela 3.1 apresenta alguns dos testes realizados para a comparação do Babelfy e do modelo de reconhecimento de entidade da AllenNLP. Se a entidade foi reconhecida com sucesso, então a célula correspondente é marcada com ✓, caso contrário é marcado com ✗. Os termos em negrito são as entidades de cada sentença.

Das opções consideradas, o Babelfy parece a mais adequada, pois além de apresentar melhores resultados nos testes realizados, o uso dessa ferramenta facilitaria no caso de desejar-se integrar futuramente a API BabelNet para acessar definições adicionais de termos, o que pode auxiliar na desambiguação de palavras presentes na pergunta. Possibilitaria ainda o uso de outros conteúdos, como imagens, traduções e relacionamentos com outros termos dentro da base de conhecimento.

3.1.2 Algoritmo de Predição

A pesquisa do estado da arte de formas de responder perguntas de domínios fechados mostrou diferentes soluções que aplicavam processamentos sobre a pergunta e sobre os

Perguntas	Babelfy		AllenNLP	
	E1	E2	E1	E2
What is **Machine Learning** ?	✓		✓	
What are **Genetic Algorithms** ?	✓		✓	
What is **Artificial Intelligence** ?	✓		✓	
What is **activation function** ?	✓		✗	
What is **gradient descent** ?	✓		✗	
What is a **generation** ?	✓		✗	
What **Neural Networks** are used for ?	✓		✓	
How do **Convolutional Neural Networks** work ?	✓		✓	
What is **activation function** on **Neural Networks** ?	✓	✓	✗	✓
What is **gradient descent** on **Neural Networks** ?	✓	✓	✗	✓
Is **generation** related to **Genetic Algorithms** ?	✓	✓	✗	✓
Is **Genetic Algorithms** a subfield of **Artificial Intelligence** ?	✓	✓	✓	✗
Is **Neural Networks** a subfield of **Machine Learning** ?	✓	✓	✓	✗
Is **Data Minig** related to **Natural Language Processing** ?	✓	✓	✓	✗
Is **Generation** the same as **Epoch** ?	✓	✓	✗	✗

Tabela 3.1: Teste de reconhecimento de entidades pela ferramenta Babelfy e pelo *Tagger* da AllenNLP.

documentos usados como fonte para extração das respostas, tal como o sistema implementado por Derici *et al* [2], o sistema de Lende *et al* [4] e a solução de Cuteri [3]. Outras soluções encontradas na literatura aplicavam modelos de aprendizado treinados a partir de extensas bases que permitiam gerar predições de respostas para perguntas específicas de um domínio, se junto a pergunta fosse fornecido um texto que contém as informações necessárias para conclusão da resposta.

Duas abordagens para QA foram consideradas: a aplicação de modelos de aprendizado e o processamento explicito da pergunta e texto que abarca a resposta. As que usam modelos de aprendizado podem ser aplicadas a diferentes domínios, basta que se forneça, juntamente à pergunta, um contexto que contenha informação sobre a resposta. Enquanto que os sistemas que utilizam técnicas de análise e processamento de texto exclusivamente, tendem a modelar particularidades do domínio em questão na sua implementação, o que elimina a necessidade de prover um contexto para a pergunta, mas que dificulta adaptação do sistema para outros domínios. Como o sistema proposto neste trabalho foca inicialmente no domínio de Inteligência Computacional, mas com a perspectiva de aplicação para outros assuntos, optou-se pelo uso do modelo de aprendizagem.

No âmbito de algoritmos de aprendizado voltados ao *Question Answering*, a base de dados criada pelo grupo de NLP de Stanford, o SQuAD [28], é frequentemente tomada como métrica de performance para os modelos propostos. Além da qualidade da base, outro fator que contribui para sua adoção, como um padrão no campo, é o ranking

dos modelos submetidos pelos pesquisadores no próprio site do SQuAD. Analisando os modelos presentes neste ranking, selecionamos o mais bem classificado que disponibiliza a implementação do modelo abertamente, o BiDAF do instituto Allen para Inteligência Artificial.

O instituto Allen disponibiliza diversos modelos para processamento de linguagem natural por meio de uma biblioteca própria feita para a linguagem *Python*. Para utilizar a biblioteca basta instalar usando um gerenciador de pacotes do *Python* como o `pip`. O comando `pip install allennlp` dará acesso a todos os modelos desenvolvidos, como a possibilidade usar as versões pré-treinadas ou treiná-los usando dados próprios. O BiDAF é modelo de *Machine Comprehension* que realiza a predição da resposta baseado no texto de entrada e na pergunta.

Question Answering. O modelo DMN proposto por Kumar *et al* [7] faz uso da base de dados bAbi e possui algumas implementações explicadas de forma concisa como em Hewitt[29]. Por isso, comparamos os resultados apresentados nos respectivos artigos de cada modelo e analisamos as qualidades que os distinguem. No capítulo 2 detalhamos as particularidades que os autores apontam como os diferenciais de seus modelos. Na Tabela 3.2 fazemos uma breve comparação entre os dois modelos, indicando suas vantagens e desvantagens mais relevantes.

Modelo	**Base de treinamento**	**Vantagem**	**Desvantagem**
BiDAF	SQuAD	Conjunto de treinamento similar a estrutura da base de conhecimento (artigos da Wikipédia)	A resposta deve ser um trecho do contexto
DMN	bAbI	Resposta podem ser compostas com termos além dos presentes no contexto	Treinada principalmente com perguntas sobre encadeamentos de fatos.

Tabela 3.2: Vantagens e desvantagens dos algoritmos BiDAF e DMN

Por fim, a escolha do modelo foi em favor do BiDAF. O principal fator contribuinte para esta decisão foi o fato desse modelo ter sido treinado com parágrafos de artigos do Wikipédia, portanto o contexto associado a pergunta é, em geral, maior que os usados pelo bAbI. Tendo em vista que os textos usados como entrada para a predição da resposta na ferramenta que propomos terão em geral extensão superior a três parágrafos, então pressupomos que o modelo DMN perderia em performance visto os resultados apresentados por Kumar *et al* [7]. Os resultados mostram que para as tarefas de *Single Supporting Fact*, que conta com apenas uma frase factual como contexto; *Two Supporting Facts*, contexto com duas frases factuais; e *Three Supporting Facts*, que conta com três frases factuais; ocorre a perda de acurácia com o aumento do tamanho do contexto.

3.2 Proposta e Arquitetura geral

Alguns sistemas de QA usam reconhecimento de entidade como uma forma de pré-selecionar documentos para aplicação de técnicas de extração de informação, a exemplo dos artigos de Derici *et al* [2] e de Cuteri [3]. O foco deste trabalho é aliar a predição de resposta feitas pelo modelo BiDAF à seleção de um contexto adequado a pergunta feita, de acordo com as entidades identificadas nessa pergunta. Ou seja, dado uma pergunta dentro do escopo de Inteligência Computacional como *"What are layers in a neural network ?"* podemos seguir uma sequência de passos para chegar a resposta:

1. Identificar que as entidades presentes são *layers* e *neural networks*.
2. Selecionar dentre os temas cobertos pela ferramenta qual o mais relevante para as entidades identificadas. Cada tema possui um pequeno texto associado, chamado de contexto, que aborda o tema.
3. Escolhido o contexto com mais chances de conter a resposta da pergunta, então este e a pergunta são passados para o modelo pré-treinado BiDAF que identifica um trecho do contexto que pode ser uma resposta para a pergunta, como no exemplo da Figura 2.3.

Também podemos adicionar a essa sequência ações que podem ser tomadas para tentar melhorar a qualidade dessas respostas. A reescrita de perguntas feitas antes da identificação de entidades, é um exemplo de artifício usado para melhorar a identificação correta de entidade e, consequentemente, a qualidade da resposta. A seguir descrevemos os principais componentes usados para desenvolvimento desses passos.

3.2.1 Interface

Ao pensar na interface deste programa tomamos como orientação aplicações de *chatbot* que usam aplicativos de troca de mensagens para conectar o serviço fornecido pela aplicação ao usuário. O *Telegram* é um desses aplicativos de comunicação entre usuários, ele possui uma uma larga comunidade ativa e tem código aberto. Sua interface é simples, intuitiva e similar a muitos outros aplicativos de troca de mensagens, como *WhatsApp* e o *Facebook Messenger*.

A Figura 3.2 mostra a troca de mensagens privadas entre dois usuários comuns. As mensagens com fundo verde claro são as enviadas pelo remetente e as com fundo branco são enviadas pelo destinatário. O ícone em forma de clipe no canto inferior esquerdo é usado para enviar arquivos e mídias, um recurso que foi explorado para a inserção de novos conceitos na base de conhecimento do domínio de Inteligência Computacional.

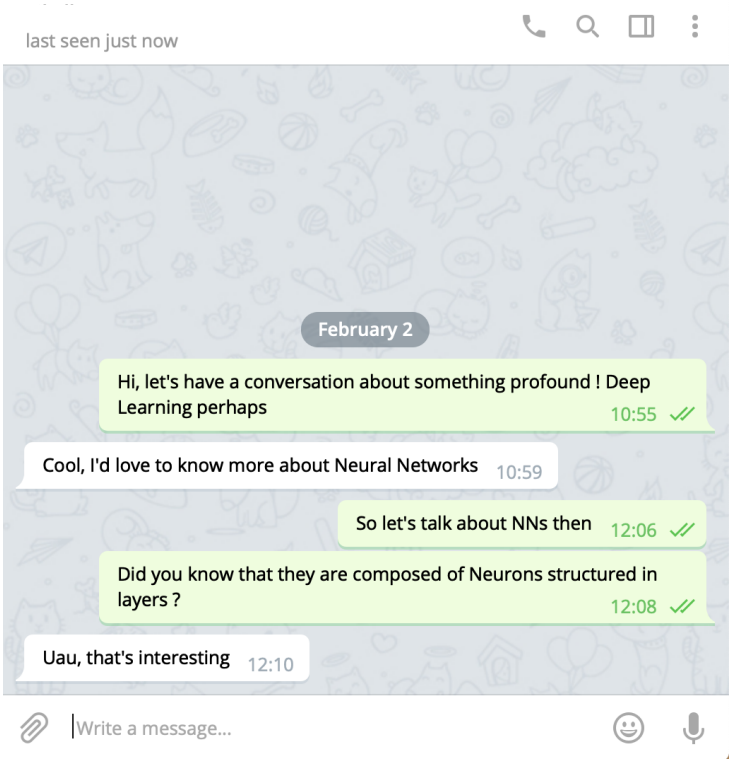


Figura 3.2: Interface do Telegram na troca de mensagens entre usuários.

A API do *Telegram* permite criar um programa que se comunique com usuários dessa plataforma de troca de mensagens. Isso habilita o programa a gerenciar as mensagens recebidas de diversos recipientes e possibilita o envio e recebimento de mensagens multimídia e arquivos.

A escolha também permitirá focar mais tempo de desenvolvimento no objetivo central de responder as perguntas do domínio, deixando a implementação da interface a cargo da aplicação Telegram, que possui a vantagem de estar disponível para os sistemas operacionais iOS, Android, Windows, Linux e MacOS. Logo a ferramenta terá um alcance maior.

Ao ganhar em facilidade de implementação perdemos em poder de customização da interface com a escolha do *Telegram*. Qualquer formas de interação com o usuário além da entrada textual, como botões, são limitadas aos meios permitidos pela aplicação. Por exemplo, a inserção de novos conceitos na base de conhecimento da ferramenta seria facilitada por uma interface na forma de um formulário, onde campos de "nome", "contexto"e outros poderiam ser preenchidos mais facilmente. Entretanto a API possibilita a inserção de botões como parte da mensagem enviada pelo *bot*.

No exemplo da Figura 3.3, uma das utilidade para botões em nossa aplicação é obrigar o usuário a selecionar uma das opções oferecidas. Isso pode ser útil para facilitar a leitura

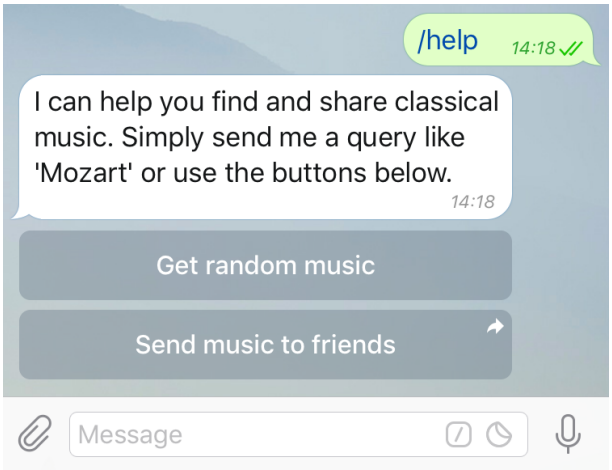


Figura 3.3: Exemplo da interface na troca de mensagem com botões entre usuário e um *bot* para para encontrar músicas clássicas.

de entradas do usuário que não sejam perguntas, como o seleção de entidade foco da pergunta e realizar *feedback* das respostas fornecidas.

A API do *Telegram* possui outro limitante, o número máximo de caracteres permitidos por mensagem. Qualquer mensagem com mais de 4095 caracteres perde os caracteres excedentes antes do seu enviada . Felizmente este limitante não prejudica nossa aplicação pois o objetivo é justamente fornecer resposta concisas.

3.2.2 Seleção de contexto e trecho resposta

Um grande empecilho para a processo de associação entre a entidade e o contexto que trata do tema dessa entidade é a variabilidade de nomes pelos quais algumas entidades podem ser chamadas. A entidade *Neural Networks*, por exemplo, possui pelo menos dois nomes associados além do comumente usado, *Artificial Neural Networks* e NN.

Se uma pergunta referente a uma entidade usar um nome diferente daquele usado para identificá-la em nossa aplicação, então por mais que o nome usado tenha sido reconhecido como uma entidade pelo Babelfy, a associação entre esta entidade e o contexto adequado será impossível, devido a ausência de uma conexão entre o seu nome e este contexto. Além disso, nos testes realizados, o modelo BiDAF não mostrou a capacidade de responder corretamente as perguntas que referenciassem um termo inexistente no contexto fornecido. Por isso a reescrita da pergunta, processamento realizado apenas para o domínio de Inteligência Computacional, e a seleção do contexto são processos que dependem de uma solução para o problema de múltiplas formas de referenciar a uma mesma entidade.

Então para evitar perguntas que seriam respondidas corretamente sejam falsamente classificadas como não referentes a nenhuma entidade na base de conhecimento, imple-

mentamos uma solução bem simples: a criação de um índice de sinônimos, estabelecendo a conexão entre nomes e entidades.

Após o Babelfy retornar a lista de entidades, realizamos uma busca no índice por esses nomes. Se o nome for encontrado, imediatamente sabemos o contexto correspondente a entidade referida.

Entretanto, no caso de perguntas que referenciam múltiplas entidades, existe a chance de mais de uma destas possuir um contexto associado, logo é preciso decidir qual delas usar para selecionar o contexto. Por falta de mais informações que permitissem escolher automaticamente o contexto mais adequado, decidimos atribuir ao usuário essa escolha. Por meio de botões, um para cada entidade referenciada, na interface do *Telegram*, o usuário deve selecionar a que melhor representa o foco de sua pergunta.

3.2.3 Módulos do software

A API do *Telegram*, a biblioteca AllenNLP e o NLTK possuem interfaces com a linguagem *Python*, disponibilizadas por meio de pacotes facilmente instaláveis. Isso contribuiu para a escolha do *Python* como linguagem de desenvolvimento do projeto, juntamente ao fato da linguagem ser amplamente usada para aplicações de Inteligência Artificial.

As principais APIs e bibliotecas usadas foram:

- pacote `NLTK`: é uma interface para *Python* da biblioteca de processamento de linguagem natural. Ela foi usada para a tarefa de remoção dos verbos das perguntas. Para isso utilizamos o *tokenizer* e *tagger* de partes do discurso.
- pacote `allennlp`: permitiu ter acesso direto aos modelos do Instituto Allen, que incluem modelos para extração de informação e reconhecimento de entidades, por exemplo. Porém, utilizamos apenas o modelo o BiDAF.
- pacote `telegram`: possibilitou uma interface com o *Python* para nosso sistema receber mensagens e arquivos de usuários, bem como construir menus com botões e enviar mensagens de resposta.
- pacote `urllib`: usado para fazer requisições HTTP à API do Babelfy. Permitiu encodar os parâmetros, como o texto da pergunta e a chave de acesso da API, para envio aos servidores da Babelfy.
- pacote `JSON`: após o recebimento da requisição HTTP e processamento pelos serviço Babelfy, o retorno é enviado no formato JSON, que é estruturado por meios de pares de chave e valor. A leitura e conversão do seu conteúdo numa estrutura de dicionário manejável pelo *Python* é facilitada com este pacote.

Outros softwares que ajudaram na implementação deste trabalho foram o *Git*, para versionamento e o *MySQL*, que foi sistema gerenciador de banco de dados escolhido para gerir a base de conhecimento do sistema.

As componentes descritas nas seções 3.2.1 e 3.2.2 elucida algumas funcionalidades necessárias para chegarmos à aplicação proposta, mas não entramos em detalhes da estrutura do software. A estrutura do software da ferramenta diz respeito ao agrupamento de partes que possuem competências similares. Por exemplo: um que método identifica os verbos de uma frase e outro método remove as ocorrências de uma palavra numa frase, ambas fazem parte do processo de remoção dos verbos realizado antes do reconhecimento de entidades.

Portanto, essas partes, ou módulos, do software foram organizadas de forma a agrupar o código de funções que realizam uma única tarefa, mas que juntas são responsáveis por funcionalidades chave no sistema. A elaboração desses módulos também levou em conta a possível evolução da ferramenta e mudanças de bibliotecas e APIs.

A seguir são descritos o papel e funcionamento geral de cada módulo que compõe o software da ferramenta, tanto para o domínio de Inteligência Computacional quanto para o domínio de documentos PDF.

- `text_processing`: Esse módulo faz uso do *tokenizer* e do *tagger* de partes do discurso disponíveis no NLTK para realizar a remoção dos verbos da pergunta do usuário, evitando assim que o Babelfy reconheça alguns verbos como entidades. No caso do domínio de Inteligência Computacional, esse módulo também é encarregado com a reescrita da pergunta, isto é, a substituição das entidades na pergunta pelo nome padrão usado pela ferramenta (este processo será descrito da seção 3.3.6).

- `ner`: É encarregado de reconhecer as entidades presentes na pergunta. O módulo tem apenas um método visível para o restante do programa, `get_entities(message)`, recebendo como parâmetro o texto da pergunta, ele deve retornar uma lista de entidades reconhecidas. Parte dessa tarefa é realizada pelo Babelfy, mas um processamento adicional é necessário, pois as informações obtidas do JSON retornado pela API do Babelfy apenas informam os índices, inicial e final, das entidades reconhecidas. Além disso, algumas entidades compostas de mais de uma palavras, como "*Computer Vision*", apresentam-se como três entidades na lista retornada, *Computer*, *Vision* e *Computer Vision*. Dessa forma, foi necessário um tratamento para selecionar apenas a entidade que não pode estendida . Mas esse tratamento é transparente para os chamadores do método, o que possibilita que a política de extração das entidades mude sem a necessidade de alterações fora deste módulo.

- `kb_interface`: Módulo responsável pela comunicação com o banco de dados, centraliza todo acesso a base de conhecimento em seus métodos. Implementamos métodos básicos para acesso aos atributos do tipo entidade e do tipo contextos, mas também criamos métodos para consultas que tinham uso frequente, por exemplo: retornar o contexto associado a uma entidade ou o relacionamento entre duas entidades existentes na base.

- `prediction`: Aqui reside a interface com o pacote do Instituto Allen, onde usamos o modelo de BiDAF. Tomando como entrada apenas o contexto e a pegunta, a biblioteca retorna uma estrutura de dicionário extremamente extensa, contendo o valor computado da atenção entre cada palavra do contexto e cada palavra da pergunta, e mais informações. No entanto, desejamos apenas extrair o trecho do contexto com a melhor probabilidade de responder a pergunta. Este trecho é acessado no dicionário por meio da chave "best_span_str".

- `bot`: Esse módulo é o que integra a API do *Telegram* ao projeto, nele reside o método executado sempre que o usuário envia uma mensagem a aplicação; o método responsável pela criação do menu de botões para certas situações; os métodos executados quando comandos são enviados pelo usuário, como \feedback ou \insertion; e inclui o método que envia mensagens para o usuário. E é neste módulo que são centralizadas as chamadas aos demais, isto é, aqui ocorre a delegação de tarefas aos outros módulos e união dos resultados retornados.

Estes módulos contém todo o código para realizar a predição de respostas às perguntas de cada domínio. Contudo, a preparação da base de conhecimento foi um processo a parte que requiriu etapas manuais e de pré-processamento, sendo diferentes entre os dois domínios.

Um descrição geral do fluxo de chamada entre os módulos, válida em ambos os domínios, seria: ao receber uma pergunta, o método de recebimento de mensagens do módulo `bot` é executado automaticamente por meio da API do *Telegram*. Então o texto da mensagem é passado como parâmetro para método de remoção dos verbos presente no módulo `text_processing`. O resultado obtido é direcionado para o método de extração de entidades da pergunta, que identifica a lista de entidades presentes. Dessa lista é preciso filtrar as entidades que constam na base de conhecimento. Essa consulta é realizado pelo módulo de `kb_interface`, que recebe a lista como parâmetro e retorna o subconjunto das entidade na base. Em seguida ocorre a seleção de contexto, a lógica desse módulo foi descrita na seção 3.2.2 para o domínio de Inteligência Computacional e será descrita para o domínio de documentos PDF na seção 3.4.3. Após a seleção do contexto apropriado, a dupla (contexto, pergunta), com os verbos presentes, é usada como argumento para a

função `predict` do módulo de predição. Após o processamento a função deve retornar a resposta, que depois será enviada ao usuário.

Essa sequência de eventos descreve como ocorre a geração da resposta à pergunta de entrada do usuário. Todavia, no caso do domínio de Inteligência computacional pode ocorrer a sugestão de temas relacionados a pergunta logo após a resposta. Neste caso o usuário é livre para escolher entre digitar uma nova pergunta, ignorando os temas sugeridos, ou selecionar um deles e receber sugestões de perguntas no tema.

3.3 Domínio de Inteligência Computacional

O domínio de Inteligência Computacional abrange temas bem diversos, como Visão Computacional, Redes Neurais, Algoritmos Evolutivos e muitos outros modelos que são usados para diferentes propósitos. O objetivo desta etapa da ferramenta não é ensinar o conceito aos usuários, visto que essa tarefa representa um desafio fora do escopo deste trabalho e demandaria um arcabouço pedagógico. Por outro lado, permitir que o estudante faça perguntas em relação ao tema sobre o qual tem pouca ou nenhuma noção, possibilita que este explore o tema e a sua relação com outros conceitos pertencentes ao tema.

3.3.1 Requisitos

A ênfase, e principal atrativo de usuários da ferramenta nesta etapa, é o aspecto exploratório. Assim, propomos uma ferramenta capaz de responder perguntas em inglês sobre o temas presentes numa base de conhecimento sobre o domínio de Inteligência Computacional, além de ser hábil a realizar sugestões de temas relacionados ao da pergunta feita.

O conjunto inicial de temas presentes na base foram pré-selecionados por sua popularidade. A lista a seguir mostra as entidades que foram incluídas:

- neural networks
- artificial intelligence
- computer vision
- activation function
- genetic algorithms
- data mining
- machine learning

- unsupervised learning
- supervised learning
- pattern recognition

Visto que o número de entidades cadastradas na base de conhecimento é pequeno, implementamos a funcionalidade de inserção de novas entidades e seus atributos por meio de um arquivo CSV. Essa implementação é detalhada na seção 3.4.2 que trata da preparação da base de conhecimento.

Quanto a interface da aplicação, as versões do *Telegram*, para *Desktop* e *Smartphones*, já satisfaz uma padronização. Então, resta a nossa ferramenta coordenar a forma com que os textos são mostrados, para que seja sempre claro quais os próximos passos que o usuário pode tomar. O uso de marcadores para tornar o texto da mensagem negrito e o uso de *emojis* foram meios auxiliares usados comunicar com o usuário.

A possibilidade de realizar-se uma avaliação sobre as respostas fornecidas pelo sistema foi vista como uma maneira de aprimorá-lo futuramente. Por isso, implementamos uma forma não intrusiva do usuário, que sentir-se apto, realizar o *feedback*.

3.3.2 Modelagem de Conhecimento do Domínio

Como o domínio de Inteligência Computacional é um campo acadêmico com muitos conceitos abstratos que estão amplamente relacionados, decidimos modelar esse vínculo entre os conceitos de maneira mais concreta, a fim de fornecer melhores sugestões de entidades relacionadas ao tema central da pergunta.

Para isso foi necessário que especificássemos como duas entidades se relacionam no Domínio de Inteligência Computacional. Consideramos algumas opções de ferramentas para auxiliar nesse processo, como *Protégé* [30], que é um editor de ontologias, onde é possível organizar e gerenciar conhecimento textual. Esse e outros editores de ontologia similares utilizam a linguagem OWL para a representação do conhecimentos e seus relacionamentos. Contudo, após investigar essas ferramentas, vimos que devido a robustez dessa abordagem, a gerência e modelagem do conteúdo do domínio de Inteligência Computacional, por meio dessas aplicações, necessitaria de mais tempo, enquanto a familiaridade com bancos de dados relacionais proporcionariam um progresso mais ágil na sua implementação.

Pudemos realizar a gerência do conhecimento de forma direta e eficiente usando um sistema gerenciado de banco de dados, em específico o *MySQL*. Por meio do uso de *scripts* na linguagem SQL realizamos não só a criação, deleção e atualização da base de conhecimento, mas também a inserção de novas entidades na base pelo usuário, sem que este tenha qualquer interface com o restante da base de conhecimento.

Para a modelagem das relações entre as entidades presentes na base, optamos por usar nomes que identifiquem o tipo de relacionamento entre duas entidades, isto é, criar triplas compostas por uma entidade fonte, uma relação e uma entidade destino. Por exemplo, o relacionamento entre as entidades *Neural Network* e a entidade *Activation Function* seria: `Neural Network-has-Activation Function`. A entidade fonte da tripla é a que rege o relacionamento.

Este formato e as categorias de relacionamento usados foram baseadas em padrões no formato de arquivo RDF, usado na modelagem de conhecimento textual. As classes foram adaptadas ao domínio de Inteligência Computacional.

A seguir mostramos as classes de relacionamento usadas para modelar o vínculo entre entidades da base de conhecimento.

- `isA`: relação de pertencimento da entidade destino à entidade fonte. Usada indicar que uma entidade é um subtipo de outra.
- `partOf`: indica uma relação pertencimento.
- `has`: usada para indicar a composição da entidade fonte com a entidade destino. Por exemplo: Redes Neurais possuem funções de ativação *Neural Networks* `has` *Activation Function.*
- `relatedTo`: esta categoria mais ampla serve para estabelecer um vínculo entre entidades que possuem conceitos ou métodos em comum.

São quatro categorias de relacionamento, que agregam mais informações ao conhecimento da base, pois permite sugerir temas semelhantes ao foco da pergunta para a exploração do usuário. Além disso abre a oportunidade para formas mais sofisticadas de seleção do contexto para perguntas complexas.

A figura 3.4 ilustra o resultado da modelagem dos relacionamentos para as entidades padrões do banco de dados.

Apesar da categoria "isA"não ter sido utilizada, ela foi incluída, pois antevimos o crescimento da base de conhecimento possuindo entidades que são sub-tipo de outras. É possível ver também na Figura 3.4 que a categoria `relatedTo`, mesmo sendo vaga em detalhes, ajuda assinalar relação entre entidades como *Machine Learnig* e *Genetic Algorithms*, que não corresponde a nenhuma outra categoria.

A escolha do contexto correspondente a cada entidade inserida é a outra parte da tarefa de modelagem da base. Selecionar um conjunto de parágrafos que descrevessem um conceito deste domínio acadêmico e que preferivelmente não fizesse citações externas ou requeresse familiaridade com outros termos técnicos, pareceu um objetivo difícil a princípio. Porém, notou-se que os textos que serviram de fonte para elaboração da base

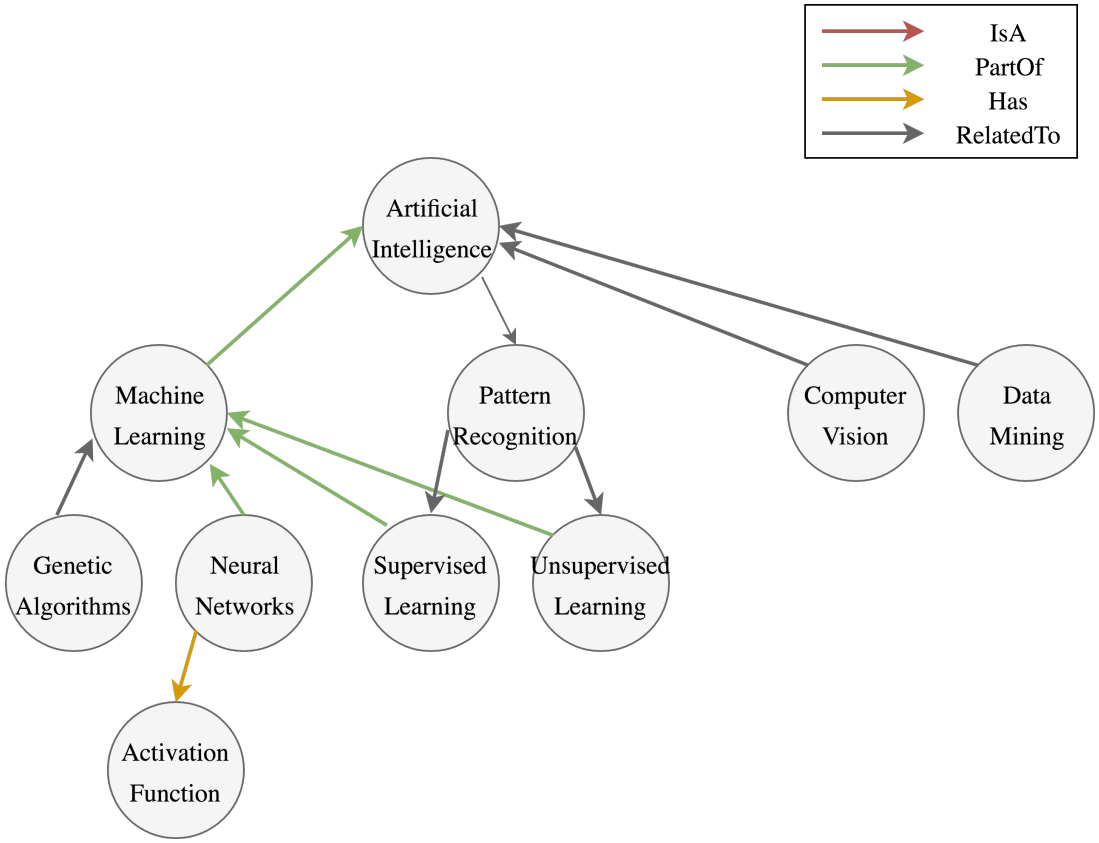


Figura 3.4: Exemplo de relacionamento de entidades na base de conhecimento.

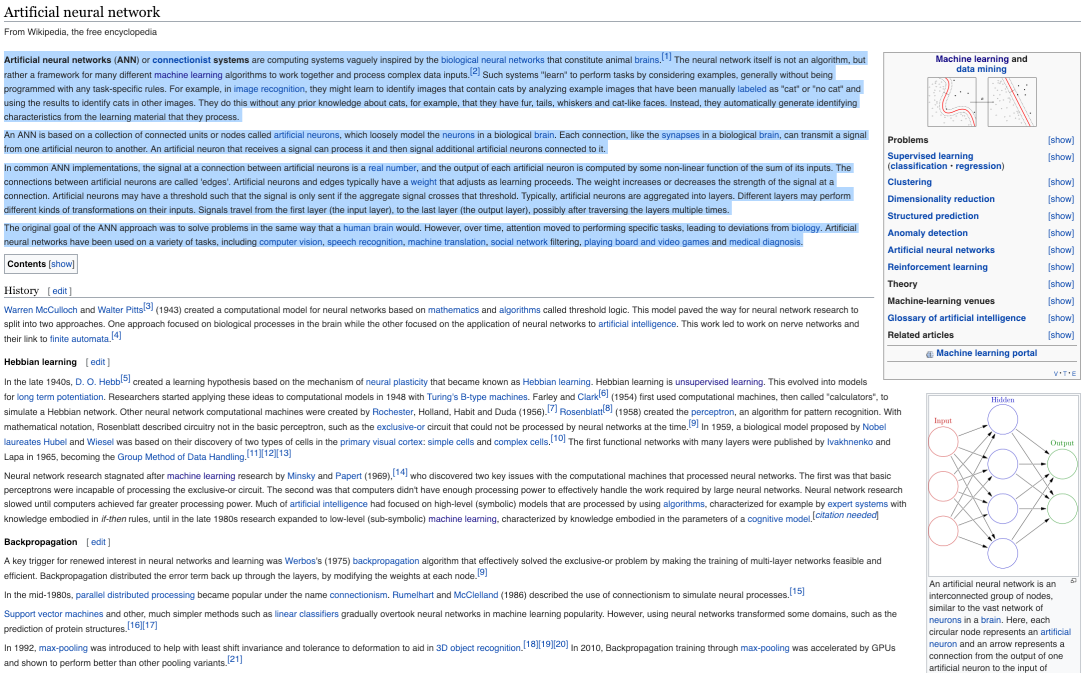
SQuAD foram parágrafos de artigos bem avaliados disponíveis no Wikipédia. Logo, usar como contexto parágrafos do artigo do Wikipédia, próprio de cada entidade, foi uma solução simples que trás o benefício de utilizar contextos similares em estrutura aos usados no treinamento do BiDAF.

Assim, foi necessário selecionar os parágrafos de cada artigo para elaborar os contextos. Esse processo foi realizado manualmente, visto que as entidades de referência são apenas as listadas no início da seção. Na seleção dos parágrafos mais relevantes de cada artigo, buscamos aqueles que resumissem aspectos importantes da entidade em questão, bem como aqueles que mencionavam a ligação entre duas entidades da base.

Também foi necessário remover as formatações e link presentes no texto do artigo, pois além comprometer a extração da resposta também poderiam aparecer no trecho resposta retornada ao usuário.

3.3.3 Arquitetura específica ao domínio

Na seção 3.2 descrevemos uma proposta de arquitetura geral com funcionalidades comuns a ambos os domínios, omitindo implementações particulares de cada domínio. A


Artificial neural network

From Wikipedia, the free encyclopedia

Artificial neural networks (**ANN**) or **connectionist systems** are computing systems vaguely inspired by the biological neural networks that constitute animal brains.[1] The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs.[2] Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Contents [show]

History [edit]

Warren McCulloch and Walter Pitts[3] (1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model paved the way for neural network research to split into two approaches. One approach focused on biological processes in the brain while the other focused on the application of neural networks to artificial intelligence. This work led to work on nerve networks and their link to finite automata.[4]

Hebbian learning [edit]

In the late 1940s, D. O. Hebb[5] created a learning hypothesis based on the mechanism of neural plasticity that became known as Hebbian learning. Hebbian learning is unsupervised learning. This evolved into models for long term potentiation. Researchers started applying these ideas to computational models in 1948 with Turing's B-type machines. Farley and Clark[6] (1954) first used computational machines, then called "calculators", to simulate a Hebbian network. Other neural network computational machines were created by Rochester, Holland, Habit and Duda (1956).[7] Rosenblatt[8] (1958) created the perceptron, an algorithm for pattern recognition. With mathematical notation, Rosenblatt described circuitry not in the basic perceptron, such as the exclusive-or circuit that could not be processed by neural networks at the time.[9] In 1959, a biological model proposed by Nobel laureates Hubel and Wiesel was based on their discovery of two types of cells in the primary visual cortex: simple cells and complex cells.[10] The first functional networks with many layers were published by Ivakhnenko and Lapa in 1965, becoming the Group Method of Data Handling.[11][12][13]

Neural network research stagnated after machine learning research by Minsky and Papert (1969),[14] who discovered two key issues with the computational machines that processed neural networks. The first was that basic perceptrons were incapable of processing the exclusive-or circuit. The second was that computers didn't have enough processing power to effectively handle the work required by large neural networks. Neural network research slowed until computers achieved far greater processing power. Much of artificial intelligence had focused on high-level (symbolic) models that are processed by using algorithms, characterized for example by expert systems with knowledge embodied in *if-then* rules, until in the late 1980s research expanded to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a cognitive model.[*citation needed*]

Backpropagation [edit]

A key trigger for renewed interest in neural networks and learning was Werbos's (1975) backpropagation algorithm that effectively solved the exclusive-or problem by making the training of multi-layer networks feasible and efficient. Backpropagation distributed the error term back up through the layers, by modifying the weights at each node.[9]

In the mid-1980s, parallel distributed processing became popular under the name connectionism. Rumelhart and McClelland (1986) described the use of connectionism to simulate neural processes.[15]

Support vector machines and other, much simpler methods such as linear classifiers gradually overtook neural networks in machine learning popularity. However, using neural networks transformed some domains, such as the prediction of protein structures.[16][17]

In 1992, max-pooling was introduced to help with least shift invariance and tolerance to deformation to aid in 3D object recognition.[18][19][20] In 2010, Backpropagation training through max-pooling was accelerated by GPUs and shown to perform better than other pooling variants.[21]

Machine learning and data mining

Problems [show]
Supervised learning (classification • regression) [show]
Clustering [show]
Dimensionality reduction [show]
Structured prediction [show]
Anomaly detection [show]
Artificial neural networks [show]
Reinforcement learning [show]
Theory [show]
Machine-learning venues [show]
Glossary of artificial intelligence [show]
Related articles [show]

Machine learning portal

v • t • e

An artificial neural network is an interconnected group of nodes, similar to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of

Figura 3.5: Seleção do contexto da entidade *Neural Networks*.

diferença dos domínios motivou a elaboração de requisitos específicos para os respectivas implementações desenvolvidas, permitindo melhorar a usabilidade da aplicação e o número de respostas corretas.

O recorrente vínculo entre os conceitos do domínio de Inteligência Computacional, motivou a elaboração de uma estrutura com relacionamentos entre entidades da base de conhecimento. Assim como o caráter didático do domínio motivou a inclusão da funcionalidade *feedback* e inserção de novas entidades. Todavia, essas funcionalidades são adicionais ao principal objetivo: explorar conceitos do domínio por meio de perguntas.

A reescrita da pergunta, desambiguação da entidade principal e sugestão de temas relacionados a última pergunta foram as funcionalidades que ajudaram a aprimorar particularmente a ferramenta do domínio de Inteligência Computacional. Mas antes que possamos detalhar estas implementações, é pertinente apresentar uma descrição da arquitetura da ferramenta, contabilizando as funcionalidades específicas que diferem o software deste domínio do software para o domínio de documentos PDF.

O diagrama da Figura 3.6 foi elaborado para segmentar o fluxo de dados em etapas que se assemelham às funções descritas na seção 3.2.3 sobre os módulos do software. No entanto, omitimos do diagrama manipulações e processamentos do software que já foram descritos anteriormente. Assim, a descrição da arquitetura será guiada pelo diagrama, facilitando o entendimento do caminho dos dados, que começa apenas com a entrada da pergunta pelo usuário.

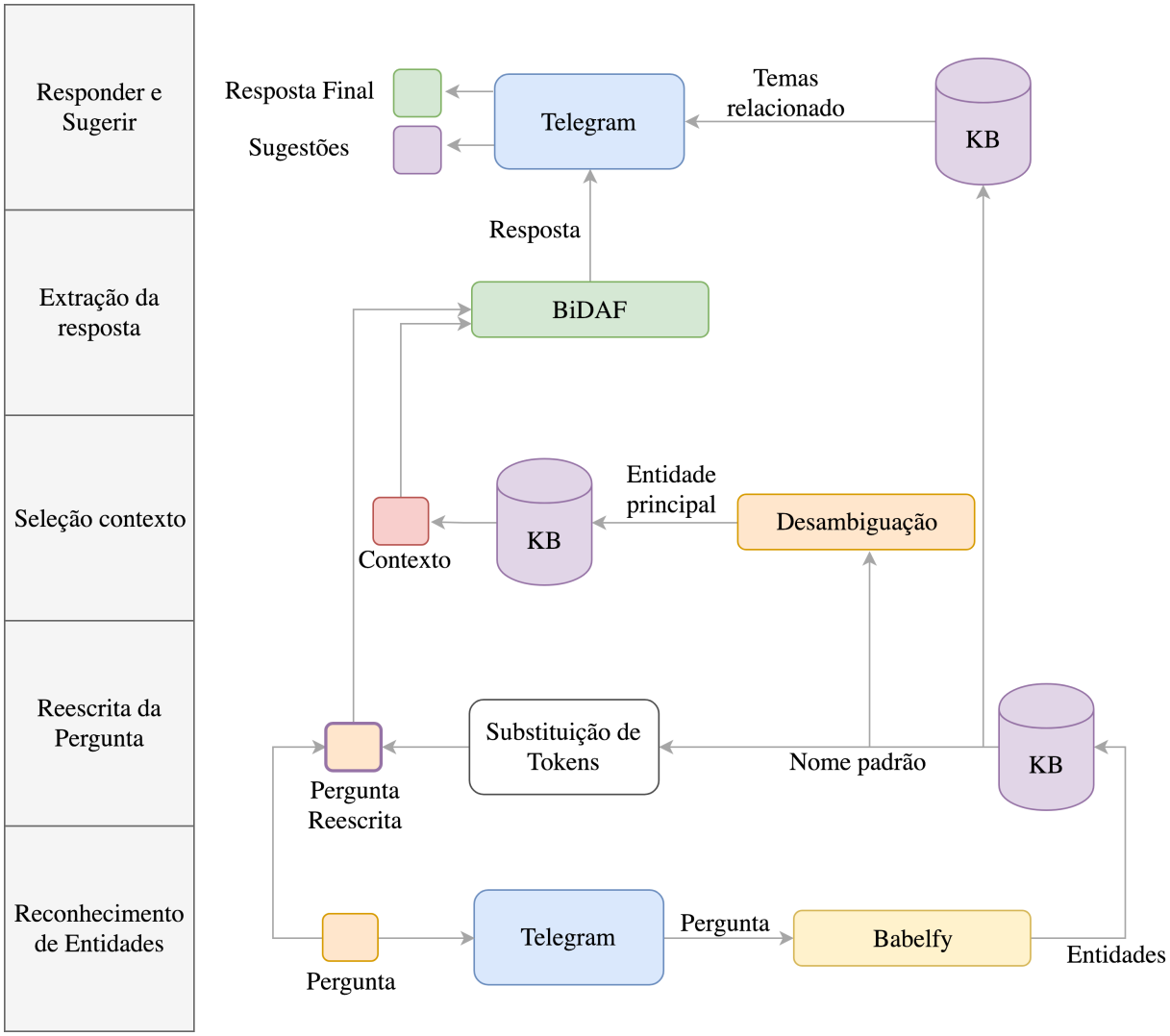


Figura 3.6: Diagrama da Arquitetura do domínio de Inteligência Computacional.

1. Na primeira etapa a pergunta é enviada pela interface do *Telegram* e recebida por nossa ferramenta, que realiza a remoção de verbos e o reconhecimento de entidades fazendo uma chamada à API do Babelfy. Em seguida, processamos a lista de entidades para que inclua apenas as entidades na sua forma mais longa.

2. Esta lista resultante é passada para o módulo de processamento de texto, que verifica se as entidades estão presentes na base de conhecimento, referida do diagrama como KB (do inglês *Knowledge Base*). As entidades encontradas são substituídas no texto original da pergunta pelo nome padrão da entidade adotado pela aplicação.

3. Antes da seleção contexto, se houver mais de uma entidade da pergunta existente na KB, é preciso selecionar qual será seu contexto usado para a extração da resposta. Portanto, o usuário deve selecionar dentre o menu de opções a entidade que julgar

de maior relevância para ser a entidade foco da pergunta.

4. Para completar a etapa de seleção do contexto, o nome da entidade foco da pergunta é usado para recuperar o seu contexto associado da KB.

5. A extração da resposta ocorre com o fornecimento da pergunta e do contexto como entrada ao modelo BiDAF, cujo retorno usado para extrair a resposta final, composta de um trecho do contexto. A resposta é então enviada por uma mensagem do *Telegram*.

6. Após o envio da mensagem de resposta, ocorre uma consulta na base a partir do nome padrão da entidade foco, que fornece a lista de entidades com as quais ela está relacionada. Com essa lista é gerado um menu de botões na interface do *Telegram* para que o usuário selecione um tema que deseja explorar. Em seguida, ele deve escolher uma das opções de pergunta padrão sobre esse tema. A pergunta escolhida passa por todo o processo descrito nos itens acima até o recebimento da resposta.

O processo de *feedback*, por ser opcional e ativado por um comando, é descrito posteriormente neste capítulo por um fluxo particular de ações do usuário.

3.3.4 Inserção na base de conhecimento

A fim de possibilitar que a ferramenta ajude na exploração do conteúdo do domínio, a função de inserir novas entidades na base de conhecimento mostrou-se pertinente. Apesar da política de permitir à qualquer pessoa inserir conhecimento na base parecer arriscada, visto que até um engano da fonte usada pode espalhar uma definição errada ou uma relação inexata, optamos por deixar essa função aberta a todos e usar formas alternativas de evitar presença de conteúdo errado na base. Incentivar o *feedback* da resposta dada à uma pergunta sobre uma entidade inserida é uma forma de determinar quais conteúdos precisam ser de corrigidos.

Primeiramente, pensou-se em realizar o processo de inserção diretamente pela interface do *Telegram*, usando mensagens envidas ao usuário para indicar qual informação sobre a nova entidade ele deve fornecer e usar as mensagens recebida para montar os dados.

Mas ao realizar testes com esse formato de inserção percebemos que ele é propenso a falhas, pois o usuário não poderia editar uma informação já enviada.

Por isso, optamos por realizar a inserção por meio de um arquivo CSV. Um arquivo modelo conterá um formulário não preenchido e será enviado ao usuário ao iniciar-se o processo de inserção. O usuário deverá baixar o arquivo, preencher os dados de cada campo do formulário, sendo que alguns poderão ter mais de uma entrada e ao concluir deve salvar o arquivo e enviá-lo de volta pela conversa com o *bot* da ferramenta no *Telegram*.

A Figura 3.7 mostra a inserção da entidade *overfitting*. Todos os campos são obrigatórios, sendo realizada a verificação se todos estão preenchidos durante a leitura do arquivo. Se todos os campos foram preenchidos, então os dados são inseridos na base de conhecimento por meio de um *script* SQL. Concluído o processo, o usuário recebe uma mensagem de sucesso, caso contrário é enviada uma simples mensagem de falha.

	A	B	C	D	E
1	**NAME**	**Contexto**	**Entidade relacionada**	**Tipo de ralacionamento**	**Apelidos**
2	**overfitting**	is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably	neural networks	relatedTo	overfit
3			genetic algorithms	relatedTo	overfitt
4					superfit

Figura 3.7: Exemplo de inserção da entidade *overfitting* utilizando arquivo CSV..

3.3.5 Estrutura e gerência da Base de conhecimento

Para gerenciar as entidades, as relações entre elas, o tipo dessas relações e contexto de cada entidade utilizamos um banco de dados relacional juntamente com gerenciador *MySQL*, que possui uma integração simples com a linguagem *Python*.

Já mencionamos na subseção 3.2.3 que o acesso ao banco foi encapsulado em um módulo da aplicação. Contudo, a estrutura das tabelas do banco e as consultas mais frequentes da aplicação serão descritas nesta seção.

Criamos tabelas distintas para entidades e para os contextos associados. Embora pudéssemos usar uma só tabela para os dois, vimos que o uso de duas tabelas adicionaria flexibilidade para mudanças futuras, como no caso de desejar-se entidades que se relacionassem com mais de um contexto. Chamamos essas duas tabelas de `entities` e `contexts`. O relacionamento entre as tabelas é explicitado pelo campo de chave estrangeira presente na tabela `contexts`.

Para melhor ilustrar organização das tabelas, elaboramos o modelo entidade-relacionamento para o banco, um instrumento recorrente para modelagem de bancos de dados relacionais.

Na Figura 3.8 são evidenciados os relacionamentos entre cada tabela e sua cardinalidade. É necessário ainda explicitar o papel da cada tabela e seus campos.

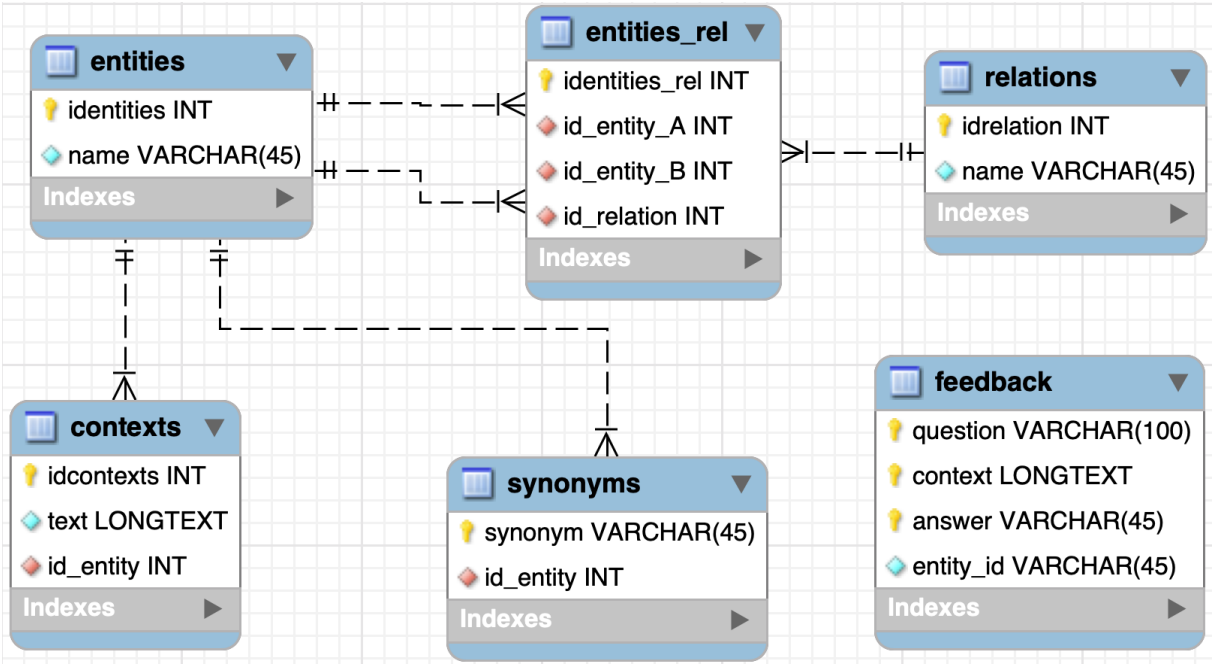


Figura 3.8: Modelo entidade-relacionamento para o banco de dados do domínio de Inteligência Computacional.

Utilizamos uma tabela de nome `synonyms` para organizar os múltiplos nomes associados as entidades. Assim, o relacionamento entre `entidades` e `synonyms` é de 1:N, pois uma entidade pode ser referida por seu nome padrão, o campo *name* da tabela `entidades`, ou sua sigla ou por outra forma do seu nome.

Assim, a tabela `synonyms` contém um campo para o sinônimo de uma entidade, *synonym*, e o identificador da entidade a qual ele se refere, *id_entity*.

A tabela `relation` apenas cataloga os tipos de relacionamentos possíveis entre duas entidades (*isA*, *partOf*, *has* e *relateTo*),apresentados na subseção 3.3.2.

Esses tipos de relacionamentos são referenciados pela tabela `entities_rel`, que tem a função de modelar relacionamentos de cardinalidade N para N entre as entidades. Assim, para todo relacionamento entre duas entidade, os campos `id_entity_A`, `id_entity_B` e `id_relation` devem indicar respectivamente: o indentificador da entidade fonte, indentificador da entidade destino e o ID do tipo de relacionamento.

A tabela de `contexts` tem os campos: `text`, referente ao próprio texto do contexto e `id_entity`, uma chave estrangeira referente a entidade associada ao contexto.

A tabela `feedback` não possui relacionamento com as restantes, pois desejamos que ela seja independente da inserção ou remoção de entidades da KB. A tabela contém os campos necessários para conhecer o cenário que motivou o *feedback* do usuário. Portanto, é vital saber a pergunta feita; o contexto selecionado, que pode mudar se for implementado

outra política de escolha do contexto; e a resposta gerada. Para tanto, a tabela possui um campo para cada um desses itens.

3.3.6 Reescrita da pergunta

Após o processamento realizado pelo Babelfy, a lista de nomes reconhecidos como entidades é retornada e analisada para determinar quais nomes estão presentes no índice de sinônimos. Os nomes encontrados no índice, serão substituídos na pergunta original pelo nome padrão respectivo, usado para referenciar a entidade internamente na ferramenta, resultando uma pergunta reescrita com entidades conhecidas.

É importante ressaltar que a escolha do nome principal usado na tabela *entities* de cada entidade foi baseada no nome mais usado para referir-se à entidade no seu respectivo contexto. Isto é, se o contexto da entidade "Artificial Neural Networks"se refere a ela na maior parte do texto como "Neural Networks", então o último será seu nome padrão.

3.3.7 Avaliação da resposta pelo usuário e sugestão de temas relacionados

Tanto a avaliação da resposta pelo usuário quanto a exploração de entidades relacionadas são procedimentos que ocorrem por meio da construção de um menu de botões que mostram ao usuário as opções que ele pode selecionar.

No caso do *feedback*, deve-se escolher entre a avaliação boa ou ruim. Mas ainda que esta avaliação feita pelo usuário ocorra de forma binária, ela pode gerar uma base de dados que pode possibilitar extensão desta pesquisa. Mas como o principal intuito desta etapa da ferramenta é a exploração do conteúdo, então determinamos que o *feedback* será uma etapa opcional ativada apenas através do envio do comando \feedback. Após a realização do *feedback* são armazenados de forma independente do restante das tabelas da KB: a pergunta original, o contexto usado e resposta obtida e se o *feedback* foi positivo ou negativo.

A sugestão de temas relacionados é outro artifício usado para estimular a exploração do conteúdo. Após o retorno da resposta, o sistema verifica quais as entidades que possuem relacionamento de qualquer tipo com a entidade foco da pergunta anterior. Essas entidades relacionadas são apresentadas como botões ao usuário, que deve selecionar aquela que tiver maior interesse. Em seguida são apresentadas perguntas construídas a partir de modelos como, "*What is* `entidade`"e "*Is* `entidade` *a kind of* `entidade_anterior. foco`?". Ao clicar em uma das opções, a pergunta é enviada ao sistema que deve respondê-la em seguida.

A avaliação da resposta e sugestão de temas relacionados são processos que acrescentam uma certa complexidade ao fluxo de uso da aplicação, mas isso é atenuado pela maneira como são acionados. A sugestão de entidades relacionadas, por exemplo, só ocorre após a resposta fornecida. O *feedback* pode ocorrer também após a resposta, sempre referente a última resposta fornecida. E em qualquer ponto de ambos os processos é possível iniciar uma nova pergunta e ignorar o processo inacabado. O diagrama da Figura 3.9 mostra como esses eventos podem ocorrer de forma cronológica e explicitando a parte do software responsável pela ação.

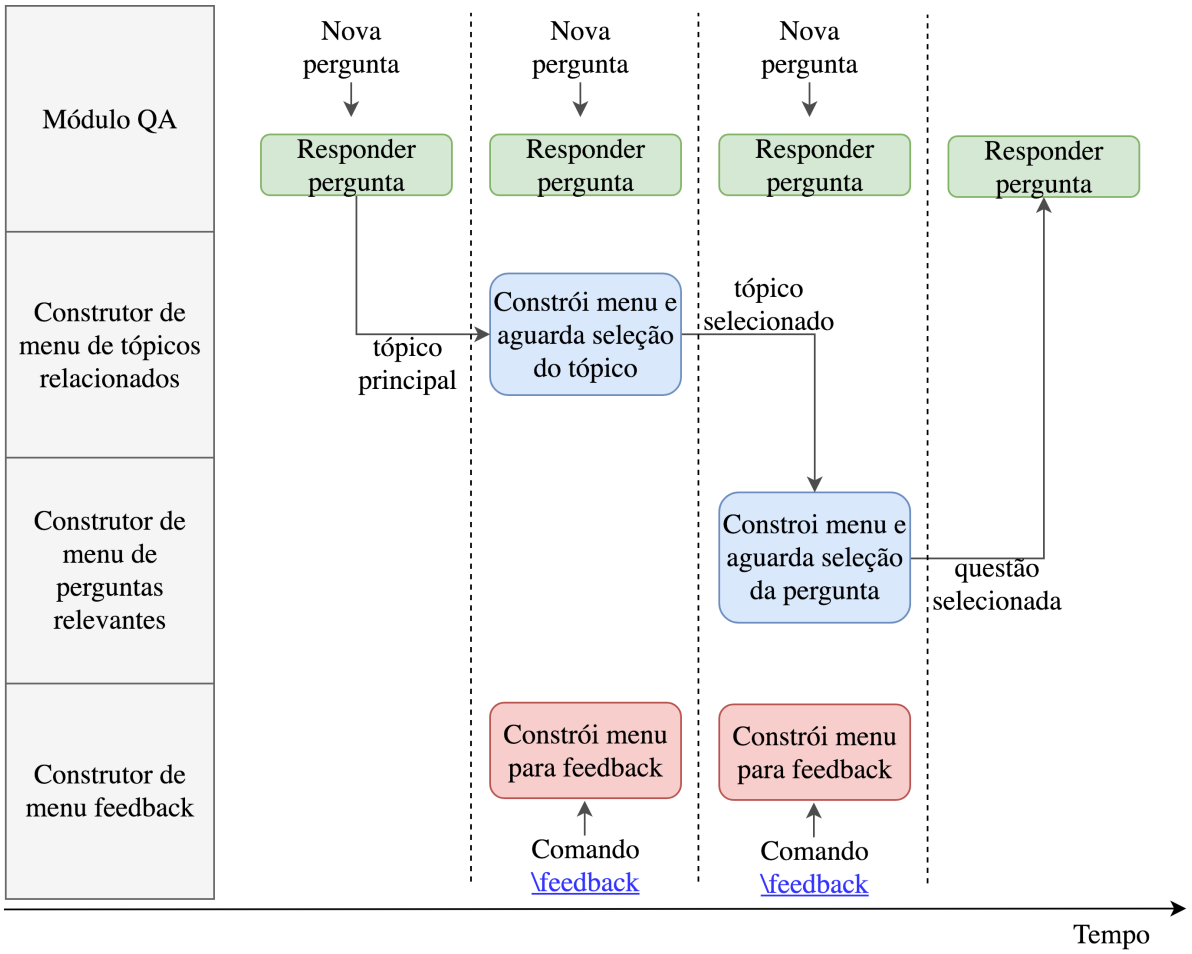


Figura 3.9: Diagrama das ações possiveis do usuário para os processos de *feedback* e *sugestão de temas relacionados à pergunta*.

3.3.8 Desambiguação da entidade foco

Esse procedimento é necessário quando a pergunta feita possui mais de uma entidade existente na base de conhecimento, gerando uma ambiguidade de qual delas deve ter seu respectivo contexto selecionado para a extração da resposta.

Esta tarefa é facilitada pelo uso de um menu de botões construídos de acordo com as opções de entidade na pergunta. O número de opções será o mesmo do número de entidades reconhecidas na pergunta e presentes na base de conhecimento. Logo que identificada a ambiguidade, o usuário irá receber como opção cada entidade e deve selecionar aquela que julgar de maior relevância para sua pergunta.

Essa delegação foi feita ao usuário visto que a ausência de informações adicionais dificulta realizar uma escolha respaldada. Além disso, esse procedimento motiva o usuário a testar a mesma pergunta em diferentes formas para verificar a diferença entre as respostas, o que incentiva a exploração do conhecimento da base.

3.4 Domínio de documento PDF

A proposta dessa etapa é possibilitar que a base de conhecimento da aplicação seja moldada pelo usuário a partir de um documento PDF de sua escolha, tornando possível realizar perguntas sobre o conteúdo do documento.

Portanto o domínio da aplicação seria reflexo do documento escolhido e totalmente decidido pelo usuário. Todavia, esse ganho em flexibilidade vem atrelado a necessidade de moldar o domínio, de forma que ele modele o relacionamento entre as entidades e seus contextos. Essa tarefa ocorre por meio de passos de pré-processamento realizados com a ajuda da entrada do usuário e execução de *scripts* elaborados para automatizar partes dos passos.

Como documento de prova de conceito, utilizamos um PDF escrito em inglês com um total de 20 páginas, sobre bolsas e dicas de vivência no Japão para pesquisadores estrangeiros.

3.4.1 Requisitos

Neste domínio os tratamentos realizados sobre a base de conhecimento precisam ser mais abrangentes em relação aos desenvolvidos para o domínio de Inteligência Computacional, visto que não é seguro assumir algo sobre o domínio de escolha, por mais que o documento de prova de conceito nos leve a crer que sim.

O usuário deve ser capaz de usar um documento PDF para gerar base de conhecimento, sendo necessário a marcação das seções do documento que devem ser encorporadas à base. Também deve ser possível ao usuário editar o índice de entidade antes da inserção.

Após a construção da base de conhecimento, os requisitos da aplicação devem ser similares aos objetivos básicos mencionados nos requisitos do domínio anterior: a interface deve permitir visualizar a sequência cronológica das mensagens e fornecer respostas às perguntas em inglês sobre o domínio do documento.

3.4.2 Preparação da Base de conhecimento

Para geração da base de conhecimento, a partir de um documento PDF, adotamos a seguinte estratégia: permitir que o próprio usuário anote o conteúdo do documento de forma a demarcar os intervalos de texto que tratam de um tema, gerando assim uma série de contextos onde cada um se refere a um tema diferente.

A aquisição das entidades representou um problema nesta fase devido o desconhecimento do domínio, mas propomos uma solução que une a contribuição do usuário à analise automática das entidades presentes no documento.

Como o caso de uso padrão da ferramenta foi pensado para documentos estruturados em capítulos, onde os contextos desejados são usualmente os capítulos desse documento, isso nos motivou a adotar o título do capítulo como um identificador para o contexto.

Para a anotação feita pelo usuário, foi adotado um padrão de marcação do PDF, usando o carácter `~`. Assim, o usuário deve inserir um "~"antes do título capítulo, após o título e no final trecho de texto. O exemplo a seguir é baseado nos capítulos do documento teste da JSPS.

`PDF_text.txt`

```
~1. Monthly Financial Support from JSPS~
  ...
~
~2. Temporary Absence from Japan~
  ...
~
  ...
~
~4. Fellowship Extension~
  ...
~
~5. Certification of Fellowship~
         ...  ...  ...
```

Proximidades entre os temas dos contextos marcados são esperadas e as ambiguidades associadas tratadas, contudo percebeu-se que quanto mais similar o vocabulário de dois contextos, maiores as chances de ocorrer uma seleção errada para o contexto que possui a resposta da questão informada pelo usuário. Por isso, no passo seguinte geramos mais um arquivo intermediário onde o usuário pode colaborar com a ferramenta.

Usando o arquivo `PDF_text.txt` anotado como entrada, o usuário deve executar um *script python* que faz uso da API do Babelfy para gerar o arquivo `indice.txt`, que

contém as entidades identificadas no documento separadas pelo título do capítulo ao qual pertencem, como no exemplo a seguir.

indice.txt

```
1. Monthly Financial Support from JSPS
housing
subsidy
cost
maintenance
allowance
family
japan
    ...
2. Temporary Absence from Japanreason
temporary
absence
research
activities
unavoidable
reasons
maintenance
allowance
    ...
3. National Health Insurancenational health insurance
plan
residents
japan
year
    ...
```

Esse arquivo de índices também pode ser editado pelo usuário, que pode adicionar entidades que possam não ter sido reconhecidas e também remover de um contexto as entidades que julgar não representar bem a temática desse trecho. Com esta contribuição a mais do usuário, a tarefa de seleção de contexto tende a ser mais precisa.

Por último basta executar o *script Python*, `importing_data`, responsável pela importação dos contextos a partir do arquivo `PDF_text.txt`, realizando também a inserção das entidades na base em que cada uma irá ser associada a um ou mais contextos. Concluído o *script*

Assim uma sequência de passos para esta etapa de preparação pode ser resumida da seguinte forma:

1. Passando o arquivo de documento PDF como entrada para um *script python* gera-se o arquivo `PDF_text.txt` com o conteúdo de texto do PDF.
2. O usuário marca neste arquivo um "~"no início do título do capítulo, ao término do título e no fim do texto relevante para o capítulo.
3. O arquivo anotado deve ser usado como entrada para outro *script* que salvará no disco um arquivo com as entidades de cada capítulo
4. Este arquivo de índice pode ser editado manualmente para diminuir a interseção entre os conjuntos de entidades de cada capítulo.
5. O *script importing_data.py* insere os dados dos arquivos intermediários nas tabelas do banco da base de conhecimento.

3.4.3 Estrutura da Base de conhecimento e seleção do contexto

O fato de não sabermos qual documento será usado para gerar a base de conhecimento da ferramenta faz com que a estrutura usada no domínio da etapa anterior não seja replicável aqui. No entanto, a fundamentação da estrutura permanece (a associação entre entidade e contextos).

Visto que para qualquer domínio escolhido as entidades inseridas na base de conhecimento serão extraídas automaticamente do documento inteiro, então as chances de entidades presentes em um capítulo também aparecerem em outros são grandes. Como no domínio anterior o contexto era diretamente vinculado a apenas uma entidade, um relacionamento do tipo 1:1, então a mesma política de seleção de contexto não era adequada.

Esse problema foi resolvido com a modelagem do banco de dados para dar suporte a relacionamentos de cardinalidade N:N entre entidades e contextos, com a mudança da política de seleção de contexto. A seleção do contexto passou a ocorrer da seguinte maneira: para uma entidade reconhecida na pergunta e na base de conhecimento verificamos os contextos associados e nestes incrementamos em uma unidade o contador próprio do contexto, que é zerado a cada nova pergunta. Isso é realizado para todas as entidades da pergunta pertencentes à base, ou seja, ao fim desta computação temos a frequência de referências a cada contexto. Logo avaliamos que a escolha do mais frequente aumenta as chances do contexto conter a resposta, tornando a escolha mais segura no caso de ambiguidade.

Como consequência das diferentes necessidades, o modelo entidade relacionamento resultante desta base de conhecimento apresentou menos tabelas, sendo necessárias apenas três:

- `entities`, que mantém referência do identificador único e nome de cada entidade

- `contexts` possui três colunas, o identificador; o "capítulo", que é nome do contexto usado para facilitar as etapas de preparação da base pelo usuário; e o texto de onde serão extraídas as repostas.
- `entities_contexts_rel` que modela o relacionamento N:N entre entidades e contexto

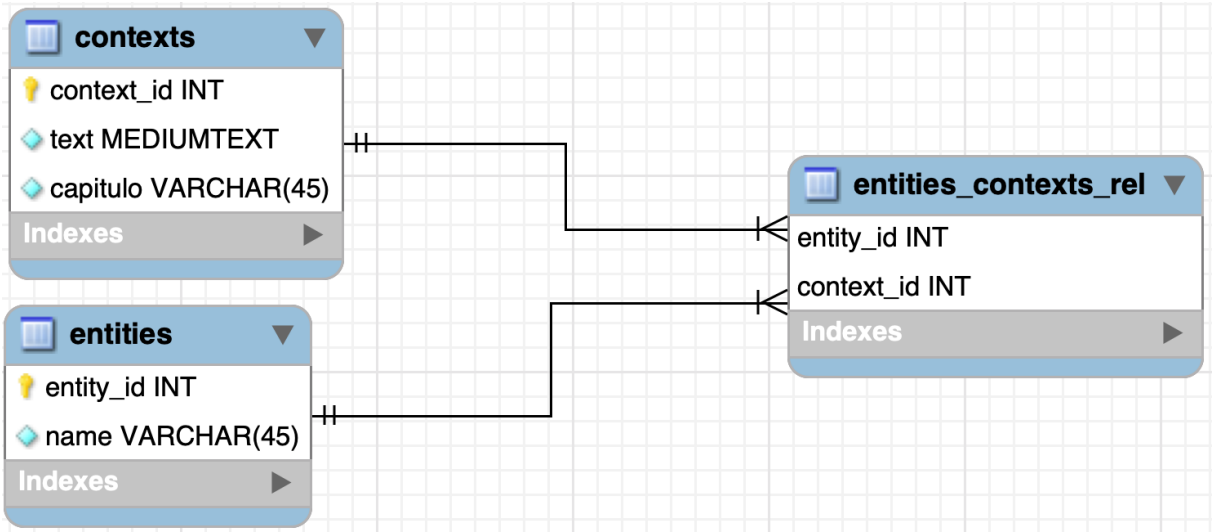


Figura 3.10: Modelo entidade-relacionamento para o banco de dados do domínio de documentos PDF.

3.4.4 Arquitetura específica do Domínio

Neste domínio, a aplicação também se encaixa nos moldes da arquitetura geral descrita no início deste capítulo, seguindo os passos: identificar entidades da pergunta, selecionar contexto mais relevante e passar a pergunta e o contexto para o modelo BiDAF realizar a extração do trecho resposta.

Mas em comparação com a arquitetura do domínio anterior, vários módulos perderam algumas funções e outros mudaram o método pelo qual atingem seu objetivo. O diagrama da Figura 3.11 replica a arquitetura do domínio de Inteligência Computacional e explicita as remoções de funções dos módulos, bem como o caminho alternativo dos dados para a nova configuração.

O procedimento de reescrita da pergunta foi removido por completo, visto que este requer um índice de nomes, ou sinônimos, associados a uma entidade, como ilustrado na Figura 3.8. O caráter flexível do domínio inviabiliza a elaboração de um índice similar. Dessa forma, após o reconhecimento de entidades na pergunta, ocorre o processo de

desambiguação, mas desta vez seguindo a política de frequência dos contextos, como descrito anteriormente.

Também pelo fato de não haver reescrita, a pergunta original é passada diretamente como entrada para BiDAF.

Visto que nesta etapa não há modelagem do relacionamento entre as entidades, então o processo de sugestão temas e perguntas relacionadas não é cabível. Portanto, as ações relacionadas estão marcadas em vermelho na seção "Responder e Sugerir"da Figura 3.11.

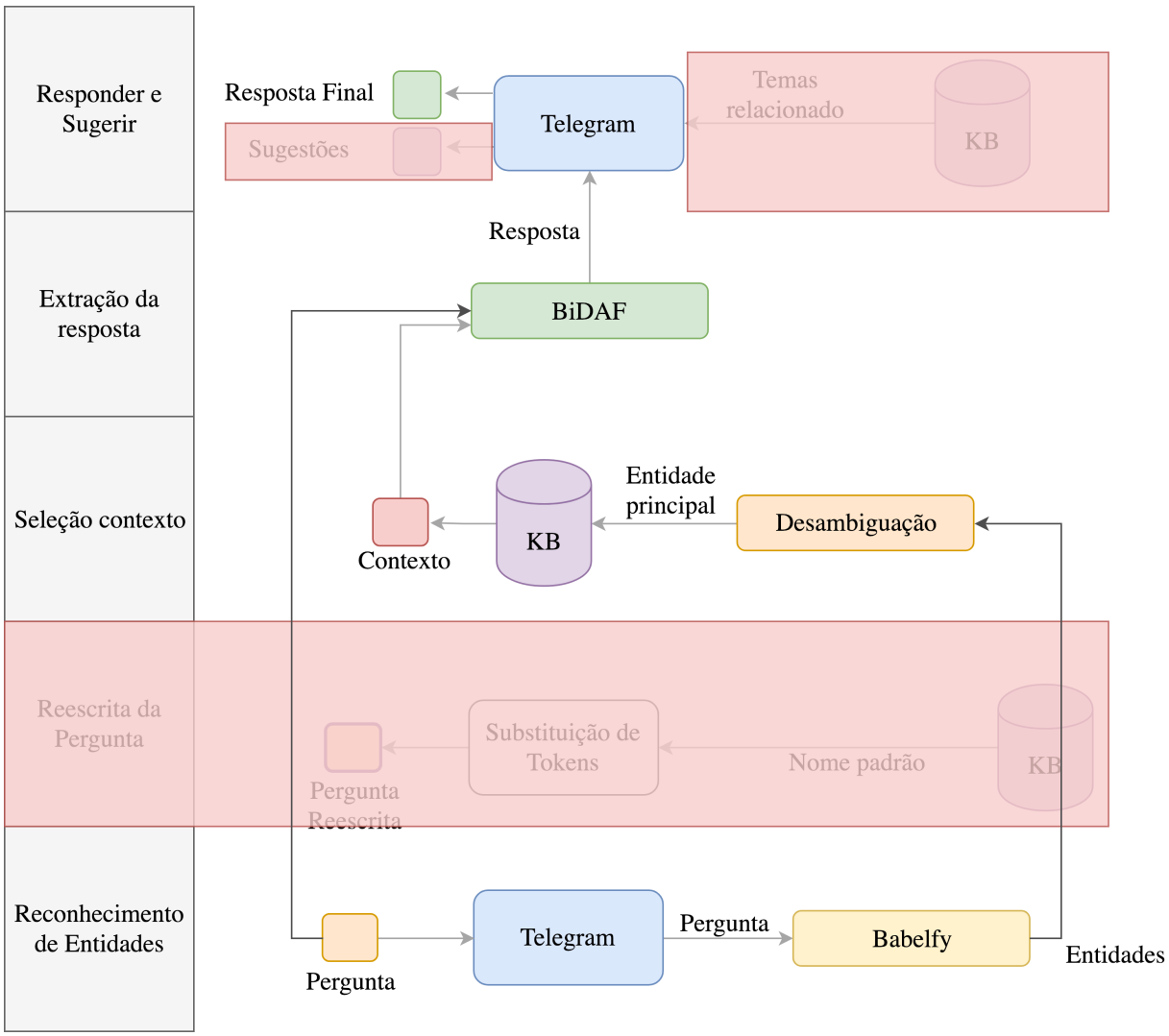


Figura 3.11: Diagrama da Arquitetura do domínio de documentos PDF.

Capítulo 4

Análise de resultados

Neste capítulo explicaremos o uso da aplicação para ambos os domínios, discutiremos os resultados dos testes realizados e abordaremos as principais diferenças entre a implementação das duas versões da aplicação.

4.1 Aplicação para o Domínio de Inteligência Computacional

A aplicação desenvolvida cumpre com os requisitos propostos e implementa as funcionalidades discutidas que aprimoram a sua usabilidade, como o *feedback*, sugestão de temas relacionados e inserção de novas entidades.

Como a interface da aplicação é um *bot* para o aplicativo *Telegram*, então antes de começar a interação, o usuário deve abrir o *Telegram*, em qualquer plataforma disponível e selecionar a conta associada a aplicação.

A seguir são descritos o fluxo de uso para as ações de iniciar o bot; enviar uma pergunta; selecionar um tema e pergunta sugeridos; realizar a desambiguação do contexto relevante para a pergunta; fornecer *feedback* sobre uma resposta; e realizar uma inserção na base de conhecimento. As descrições são acompanhadas de referências as figuras que representam o estado da aplicação para a ação descrita.

- Na primeira vez que selecionar a conversa com o *bot* é necessário inserir o comando `\start`. Em seguida o *bot* retorna com uma mensagem de saudação e sinaliza que o comando de avaliação da resposta pode ser usado. Vide Figura 4.1.
- O próximo passo é o usuário enviar uma pergunta, se esta contiver exatamente uma entidade coberta pela base de conhecimento, então a resposta será retornada, como na Figura 4.3. Imediatamente após a resposta, segue a sugestão de temas relacionados a entidade foco da pergunta.

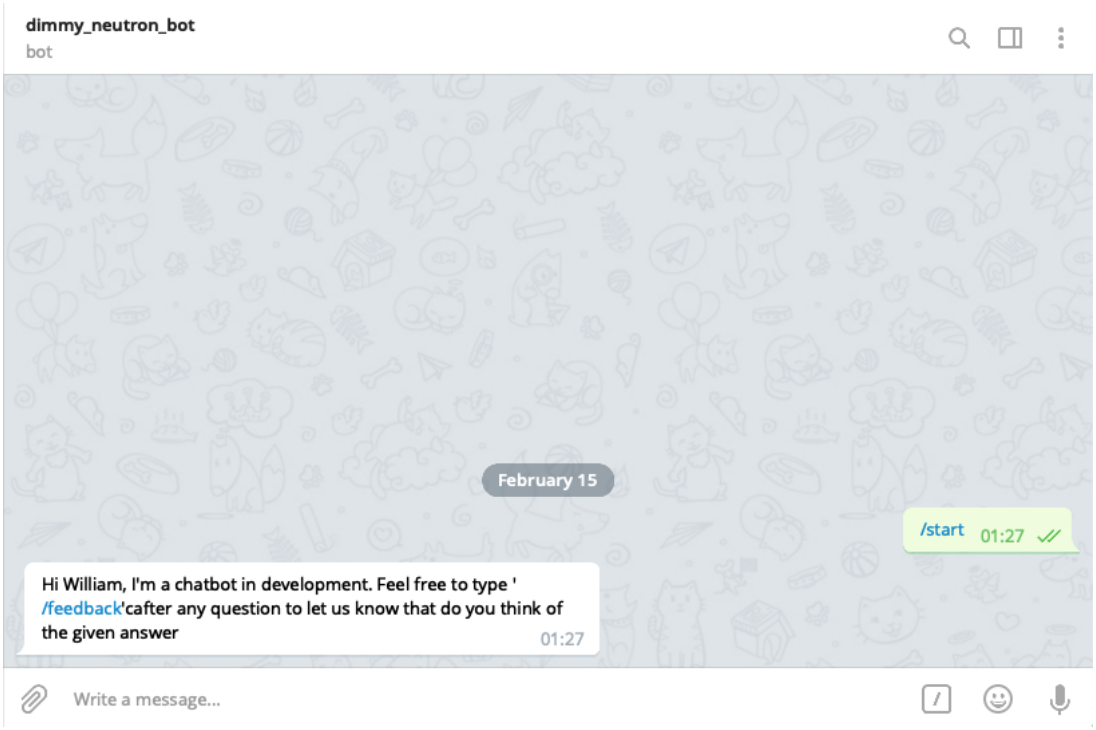


Figura 4.1: Iniciação do *bot*.

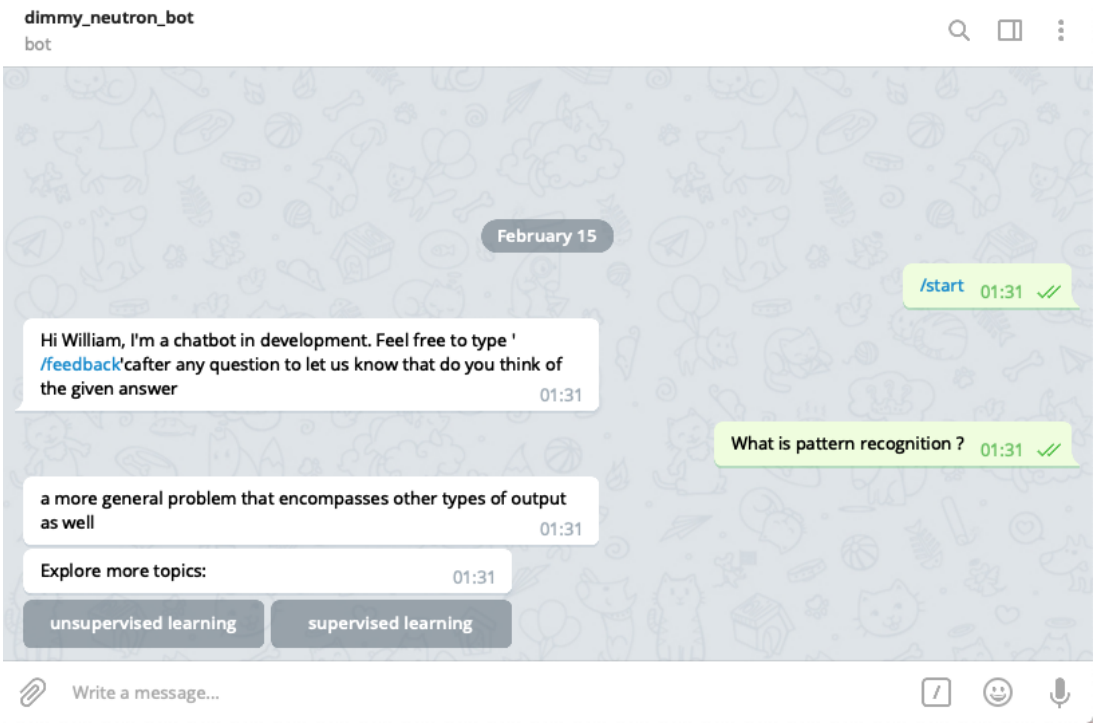


Figura 4.2: Exemplo de resposta não satisfatória seguida de sugestão de temas relacionados..

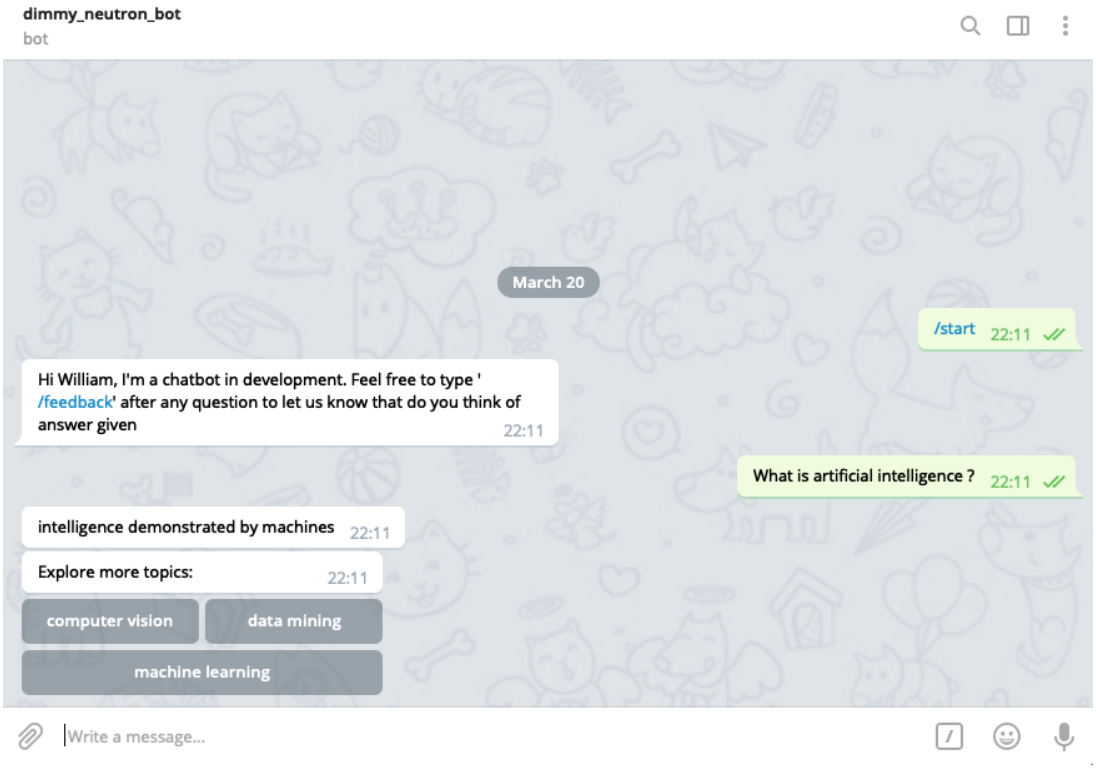


Figura 4.3: Exemplo de resposta satisfatória seguida de sugestão de temas relacionados..

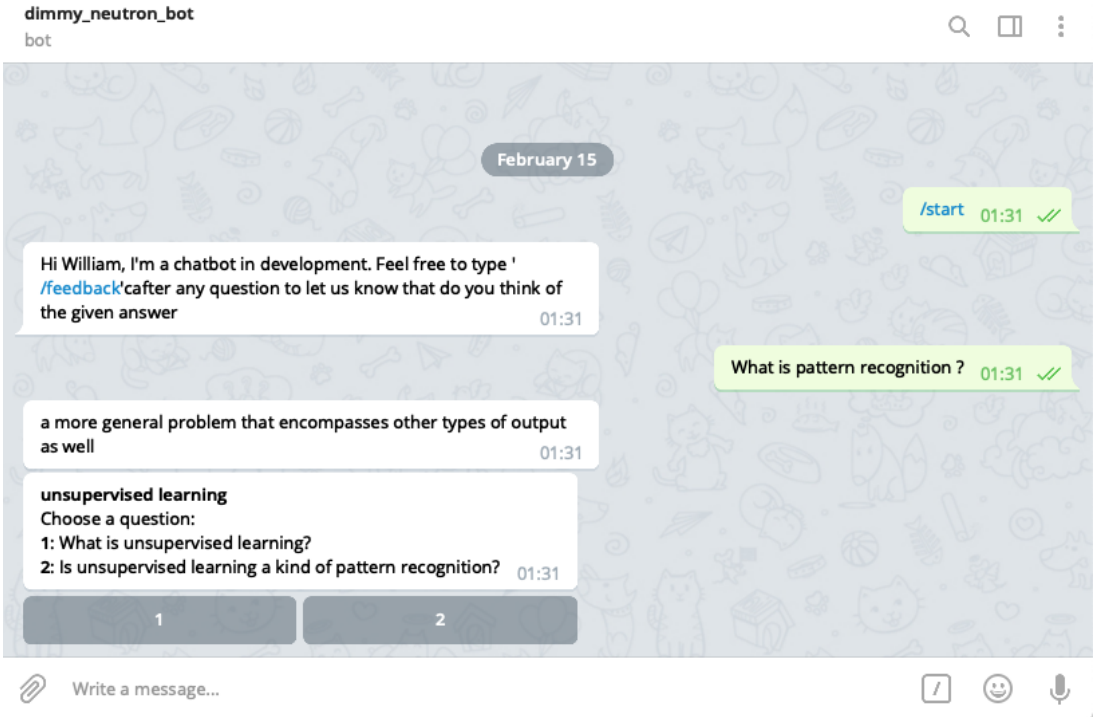


Figura 4.4: Sugestão de perguntas para o tópico selecionado.

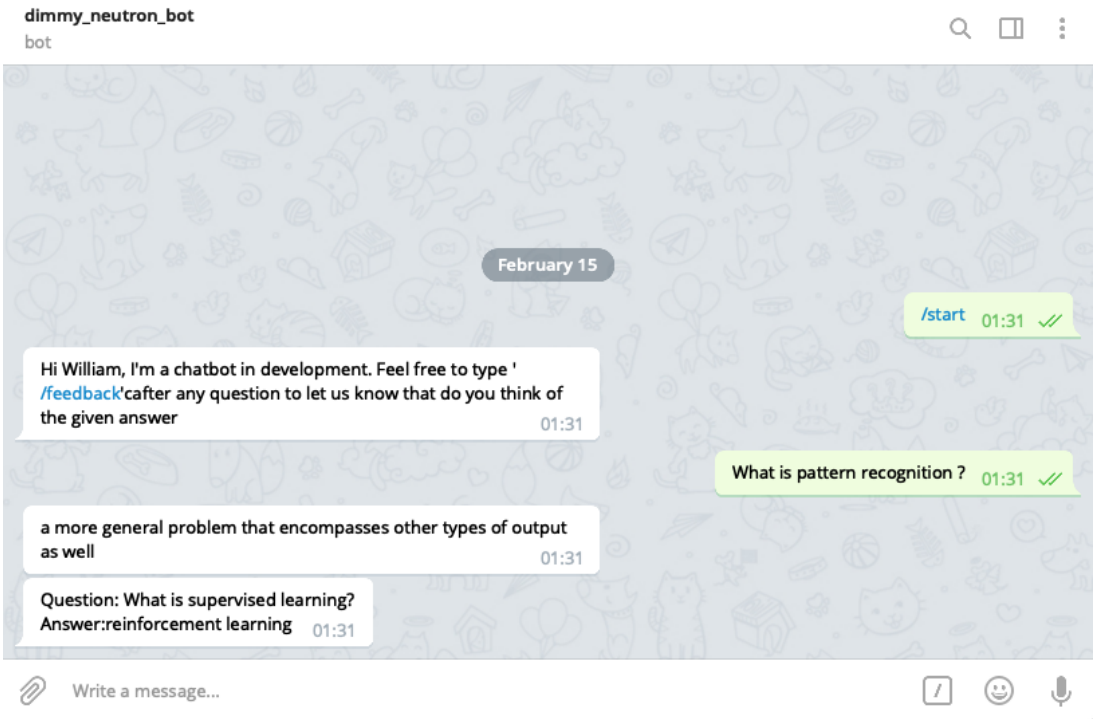


Figura 4.5: Evidenciação da pergunta selecionada e sua respectiva resposta.

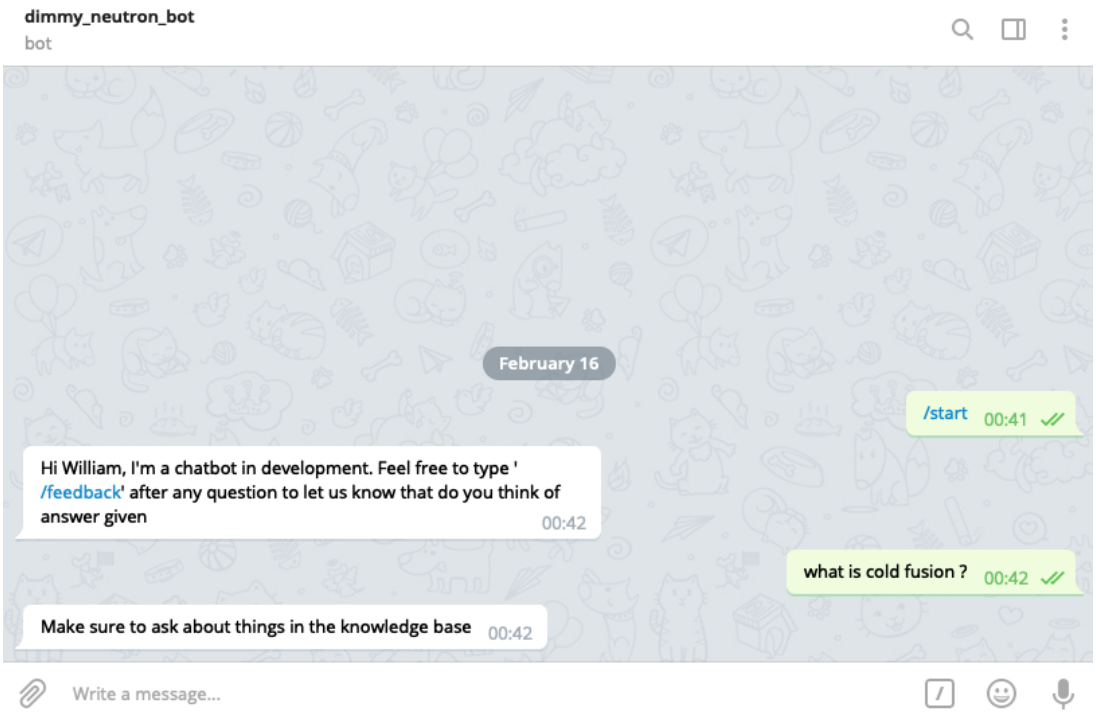


Figura 4.6: Exemplo de pergunta com nenhuma entidade reconhecida.

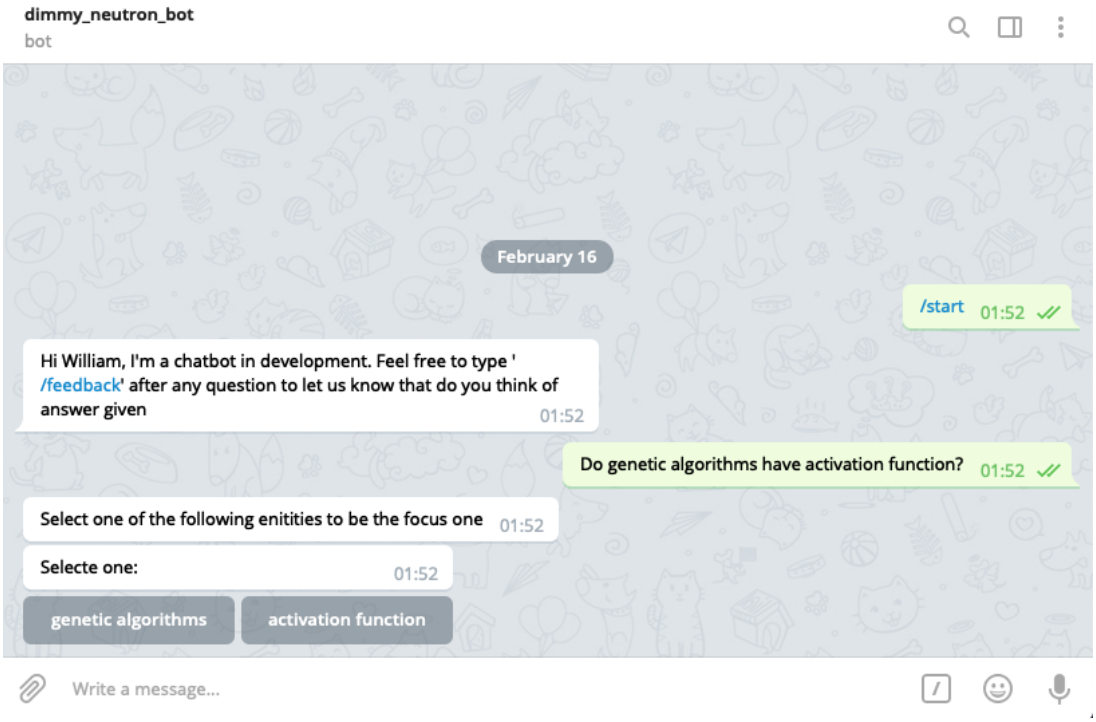


Figura 4.7: Seleção de opções para desambiguação.

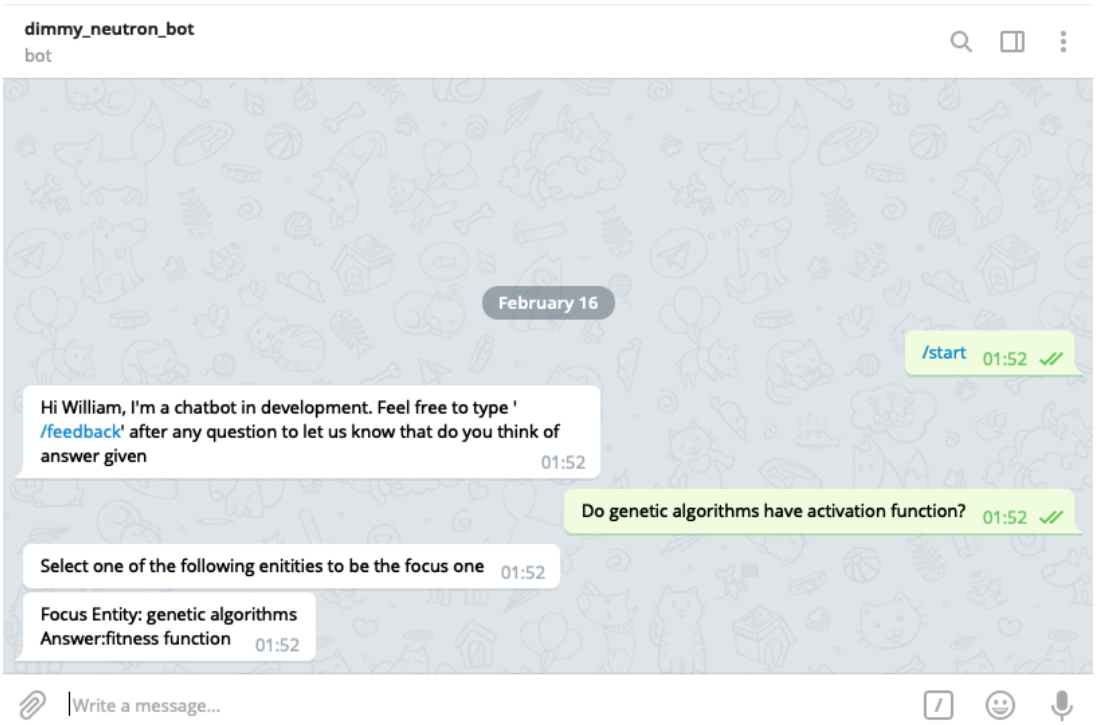


Figura 4.8: Evidenciação da entidade foco selecionada e sua respectiva resposta.

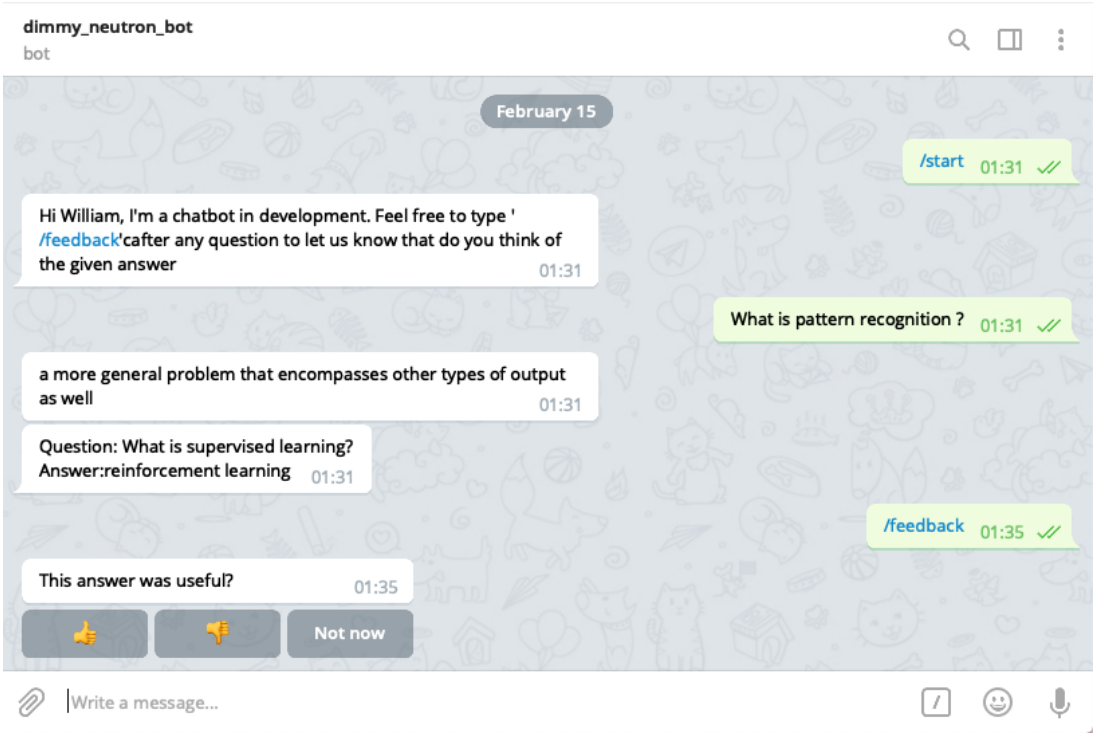


Figura 4.9: Envio do comando \feedback pelo usuário.

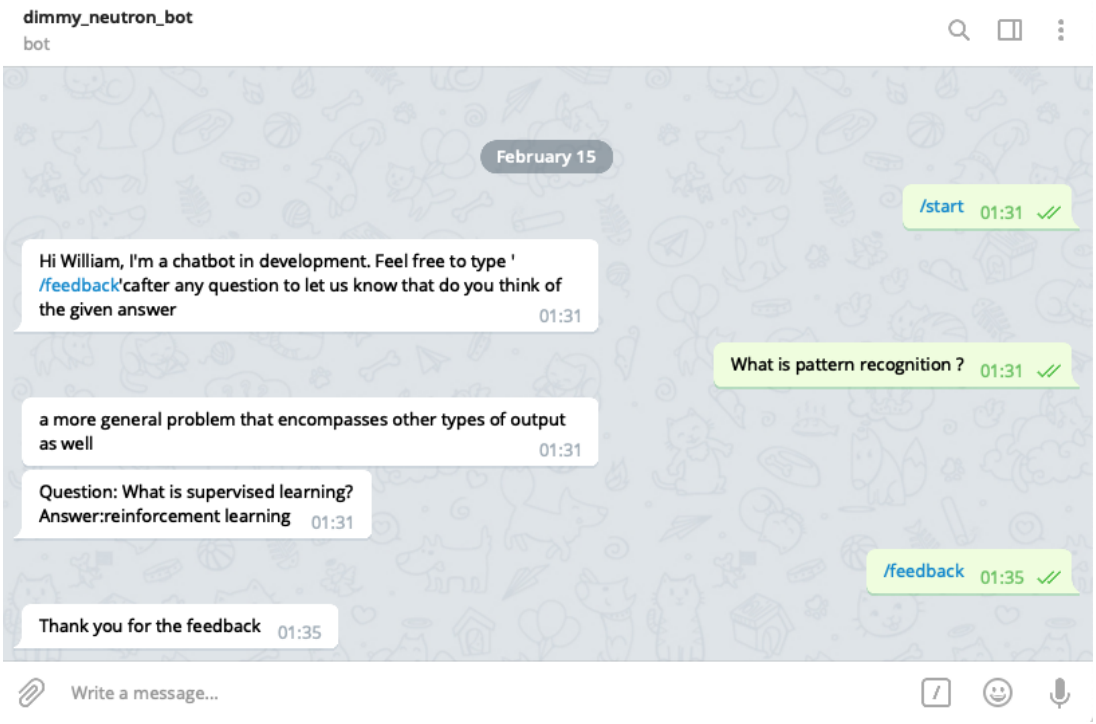


Figura 4.10: Mensagem de retorno após a avaliação.

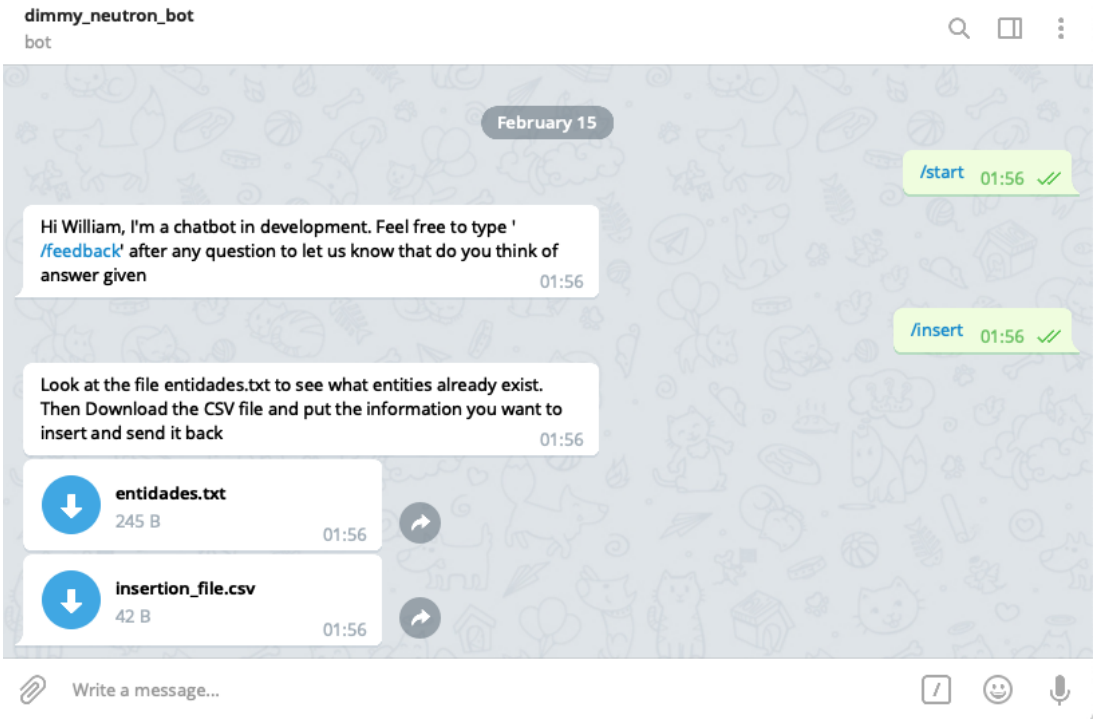


Figura 4.11: Envio do comando \insert pelo usuário e retorno com instruções e arquivos que auxiliarão na inserção.

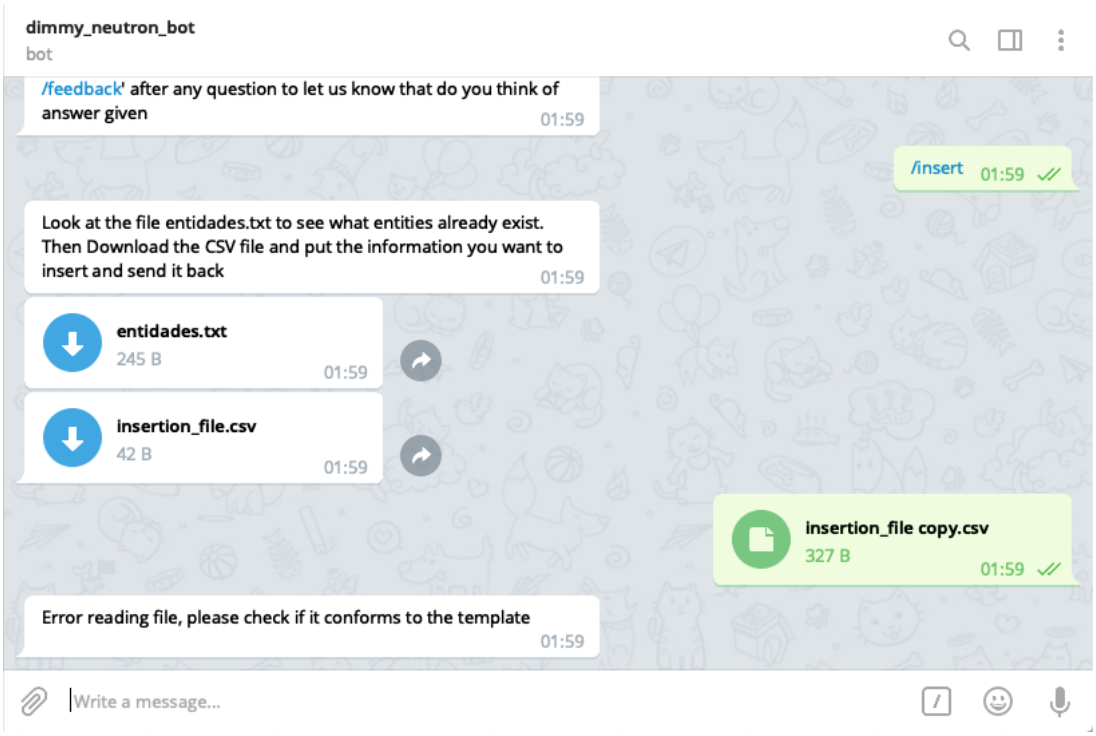


Figura 4.12: Exemplo de inserção de entidade com erro.

  - Se a pergunta não contiver nenhuma entidade cadastrada pela aplicação então o usuário recebe uma mensagem de aviso, como na Figura 4.6.
  - Caso a pergunta contenha múltiplas entidades do domínio de Inteligência Computacional, então o usuário deve selecionar qual deve ser usada como entidade foco para a melhor identificação da resposta, como ilustrado na Figura 4.7.

- Se a pegunta não requerer desambiguação, então o processo de sugestão segue a partir da Figura 4.1 com a seleção de uma entidade pelo usuário e substituição das opções de entidade por opções de perguntas pré-definidas sobre a entidade selecionada. Vide Figura 4.4.
- Após a seleção de uma das perguntas sugeridas, a resposta final é retornada. Vide Figura 4.5.
- Após a seleção da entidade foco, o contexto correspondente é selecionado e a resposta é enviada ao usuário juntamente com a explicitação da entidade foco usada. Vide Figura 4.8.
- O *feedback* é um processo opcional que deve ser iniciado pelo usuário através do comando `\feedback`. Em seguida será apresentado um menu com três opções, da esquerda para direita: avaliação positiva, avaliação negativa e cancelar. Como mostra a Figura 4.10.
- Após o recebimento do *feedback* o menu de opções é substituído por uma mensagem de agradecimento.
- A inserção também é iniciada por meio de um comando do usuário, como mostrado na Figura 4.11. Em seguida são retornados dois arquivos e um texto com instruções de como proceder para realizar a inserção. O primeiro contém uma lista das entidades já presentes na base de conhecimento. O segundo é um formulário no formato CSV que deve ser preenchido com as informações da nova entidade e, por fim, enviado de volta na conversa com o *bot*.

  - Se todos os campos do formulário forem preenchidos e não houver erro na leitura do arquivo, então a inserção ocorre e uma mensagem de sucesso é retornada. Vide Figura 4.12.
  - Caso contrário, uma mensagem de erro é retornada. Vide Figura 4.13.

4.2 Aplicação para o Domínio de documento PDF

Nesta etapa a atenção foi dedicada ao processo de preparação de métodos que permitissem a flexibilidade do domínio para documento selecionados pelo usuário em formato PDF.

Vale destacar a importância de especificar o capítulo usado para extrair a resposta neste domínio, pois há situações em que a resposta retornada pode ser válida para informação buscada. Entretanto, há casos que se o contexto usado não for adequado, essa resposta pode não ser correta.

Essa situação ocorre para o domínio do documento da JSPS com a pergunta "*How long can I be absent from Japan ?*", para a qual espera-se a reposta "*90 days*", extraída do contexto "*Monthly Financial Support from JSPS*". Porém a resposta fornecida é "*2 months*"e o contexto selecionado foi o capítulo "*4. Fellowship Extension*". A reposta errada é consequência da seleção equivocada do contexto, essa seleção ocorre devido a dois fatores: primeiro, o Babelfy identifica o termo "*long*"como uma entidade; segundo, o termo "*long*"está presente no texto do capítulo e na pergunta.

Em consequência existe o relacionamento entre a entidade *long* e o capítulo "*4. Fellowship Extension*"na base de conhecimento.

A aplicação do domínio do documento da JSPS tem um fluxo de uso similar, porém mais simples, pois não há inserção, sugestão, desambiguação ou *feedback*. A seguir apresentamos uma descrição desse fluxo acompanhada das figuras que representam o estado da aplicação.

- Após a iniciação do *bot*, a pergunta é enviada e processada. Vide Figura 4.14.
- Em seguida a resposta é retornada, sinalizando o contexto usado para extraí-la. Vide Figura 4.15.

Visto que este domínio é mais compatível com perguntas factuais, as quais podemos avaliar segundo um gabarito objetivo e pré-determinado, realizou-se um conjunto de teste, com um total de 34 perguntas, que pode ser vistas na Tabela A.1, sobre os 10 capítulos do documento. As respostas obtidas foram, primeiramente, divididas em duas classes: a classe das que o contexto usado para extrair a resposta foi o mesmo do gabarito; e a classe das que o contexto divergiu do gabarito. A Figura 4.16 mostra o resultado dessa categorização. Das 34 perguntas, 85,29% tiveram o contexto selecionado adequadamente e 14,71% não tiveram contexto não adequados.

Uma segunda categorização ocorreu, mas desta vez apenas sobre o conjunto da perguntas que tiveram o contexto adequado selecionado, um total de 29. Categorizamos estas de acordo com a resposta obtida: se foi igual a esperada, ela é categorizada como correta; se a resposta contém apenas parte da informação esperada ou contém mais informações

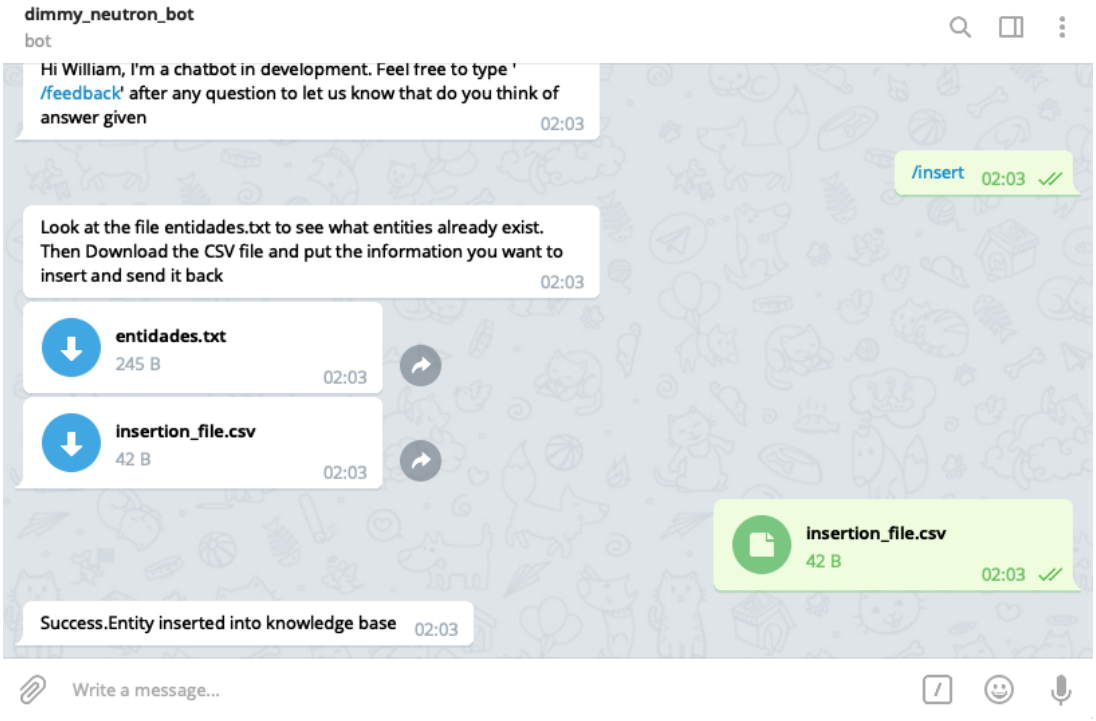


Figura 4.13: Exemplo de inserção de entidade com sucesso.

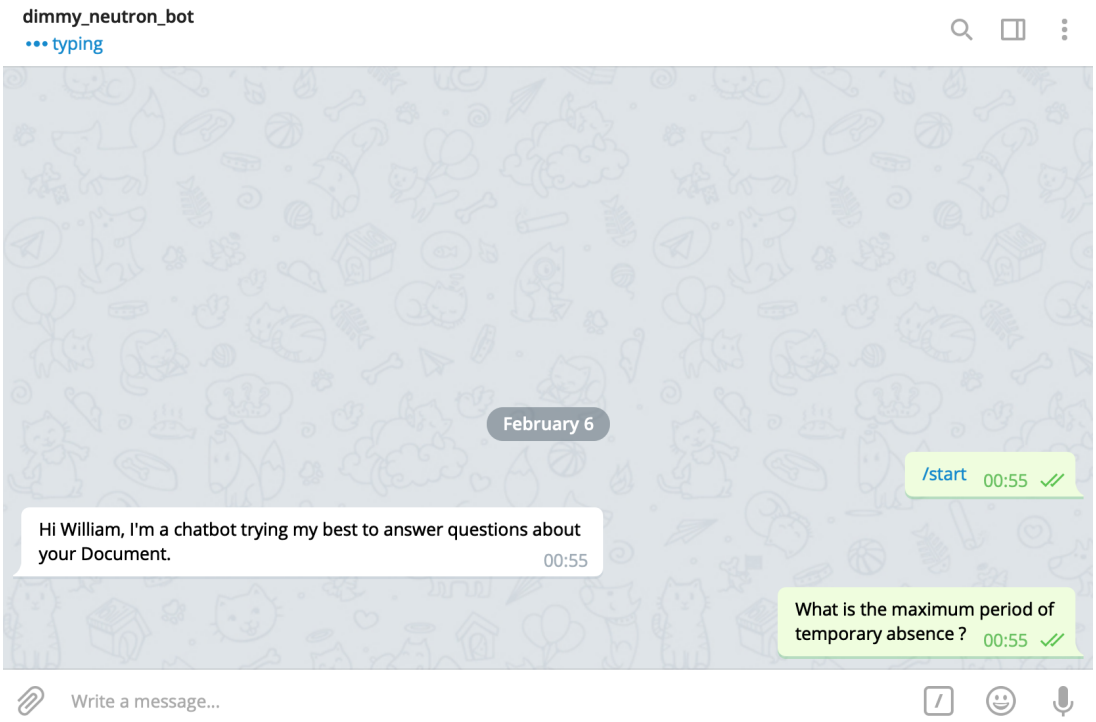


Figura 4.14: Pegunta sobre o domínio do documento PDF.

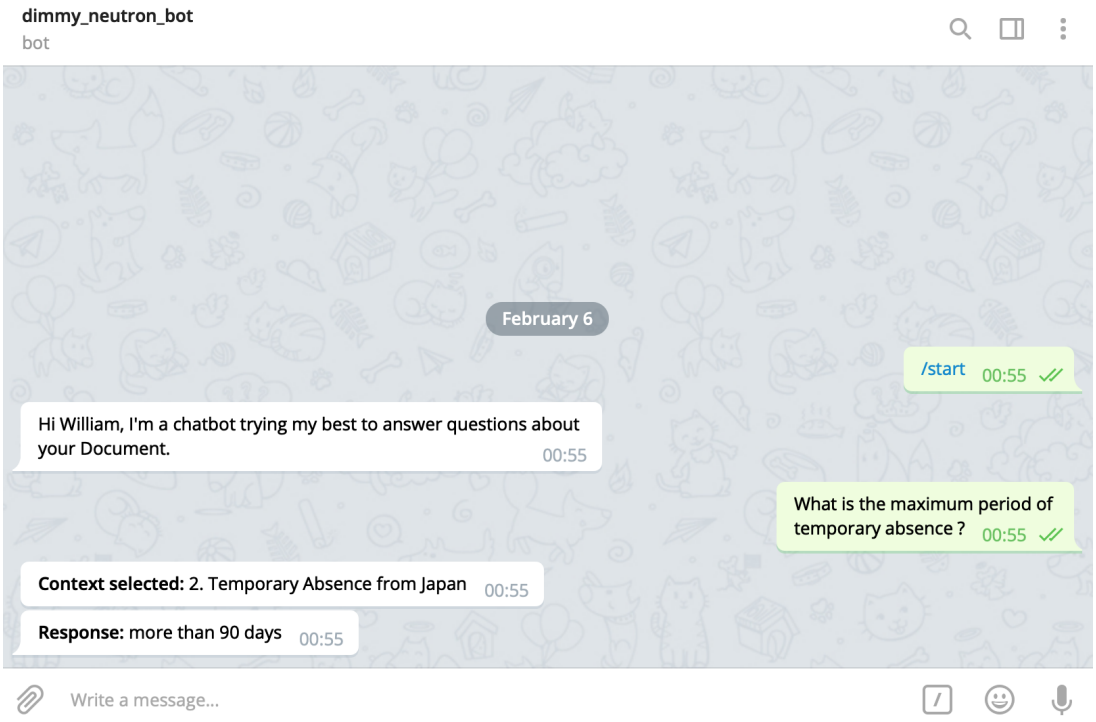


Figura 4.15: Resposta à pegunta do domínio do documento PDF.

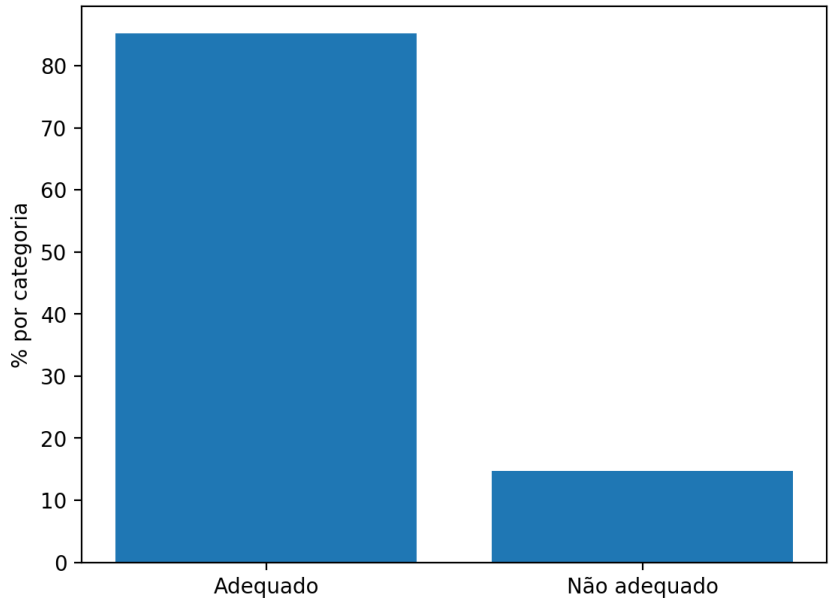


Figura 4.16: Categorização das perguntas de acordo com a seleção do contexto realizada.

do que o necessário, ela é categorizada como aproximada. Se a resposta fornecida não contém nenhuma parte da informação requerida na pergunta, então ela é classificada como errada. A Figura 4.17 ilustra os resultados.

Do total, 44,83% foram corretas, 41,38% foram aproximadas e 13,79% foram erradas.

A porcentagem de perguntas com contexto selecionados adequadamente indica que a política de seleção de contexto foi satisfatória para as necessidades do projeto.

4.3 Diferenças entre os Domínios

Além das diferenças de presenças das funções de sugestão de temas, desambiguação pelo usuário, inserção de novos conhecimento e o *feedback*, é interessante notar contrastes decorrentes dos respectivos domínios.

Após o desenvolvimento de ambas as aplicações, ficou claro que a diferença entre os requisitos das duas versões e a diferença entre os domínios, fizeram com que as funcionalidades particulares fossem encorporadas. Por exemplo, o requisito de flexibilidade do domínio para a aplicação da segunda etapa teve como consequência a necessidade remodelar o banco de dados que armazena o conhecimento, de forma a permitir o relacionamento N:N entre entidades e contextos, além de ignorar possíveis relacionamentos entre as entidades do documento. Da mesma forma que o fato do documento da JSPS possuir capítulos com muitas referências às entidades típicas de outro capítulo fez surgir a necessidade de um método rebuscado de seleção do contexto para a pergunta.

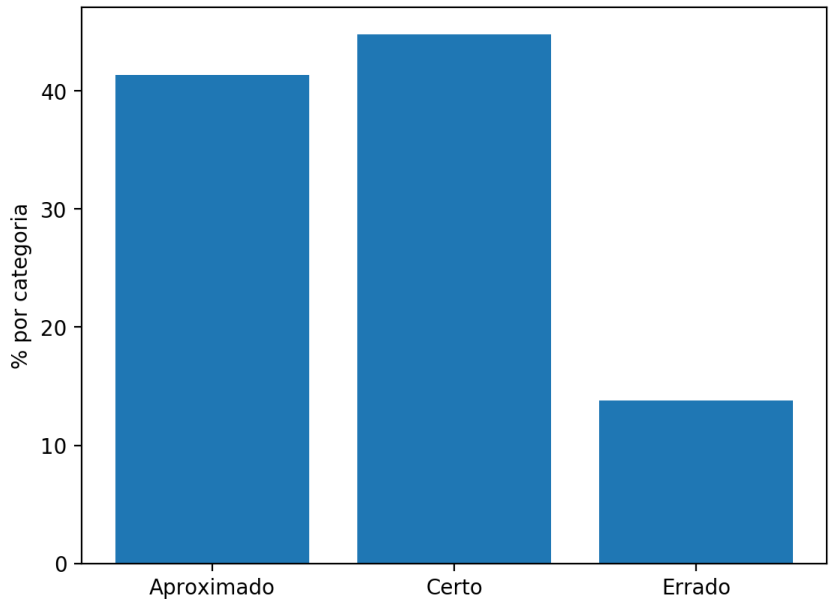


Figura 4.17: Categorização das perguntas de acordo com a proximidade da resposta ao gabarito.

Capítulo 5

Conlusão e Trabalhos Futuros

Neste capítulo apresentamos as conclusões a respeito das aplicações desenvolvidas e oportunidades de continuação deste trabalho.

5.1 Conclusão

O objetivo da primeira etapa da pesquisa, elaborar uma aplicação para exploração de um domínio fechado por meio de perguntas do usuário e resposta geradas com base na modelagem desse domínio, foi alcançado. É possível obter resposta sucintas, tirar dúvidas pontuais e explorar a relação entre conceitos da área. Porém, a complexidade de alguns conceitos torna difícil aludi-los apenas em forma textual. Além disso, a ocorrência de respostas incorretas cresce com aumento da complexidade das perguntas feitas. Perguntas factuais, cujas respostas estão explicitadas no contexto, são normalmente respondidas com exatidão. Já perguntas que relacionam diferentes informações ou simplesmente que precisam ser respondidas com palavras que não estão presentes no contexto, recebem respostas menos satisfatórias.

O fato de a resposta retornada pelo modelo BiDAF constituir um trecho do contexto fornecido, representou uma perda de alcance nas perguntas respondidas corretamente. Especialmente para o cenário em que o domínio é a Inteligência Computacional e o público alvo são iniciantes no assunto, que frequentemente requer algum tipo de elaboração da informação no contexto para constituir uma resposta direta a pergunta.

Contudo, a ideia de usar a associação entre entidades e contexto pré-cadastrados, para livrar do usuário da responsabilidade de fornecer a fonte da resposta para sua dúvida, foi validada e serviu aos objetivos propostos. O domínio da primeira etapa permitiu que a aplicação respondesse perguntas sobre uma gama de assuntos tendo como base apenas alguns contextos correspondentes as entidades *Neural networks*, *Supervised Learning*, etc.

Ambas as etapas da pesquisa foram beneficiadas pela divisão da informação do domínio em contextos independentes, que permitiu superar o problema do limite de tamanho do contexto de entrada para o modelo BiDAF. Além disso, o uso de múltiplos contextos facilitou a elaboração da base de conhecimento das duas etapas do projeto, na primeira etapa permitiu a associação direta entre os trechos de artigos do Wikipédia e os contextos. E Na segunda etapa, utilizou-se a estrutura já existente de capítulos do documento, logo foi possível associar diretamente contextos à capítulos.

5.2 Trabalhos Futuros

Os dados coletados dos *feedbacks* de resposta do domínio de Inteligência Computacional podem ser usados para melhorar a aplicação. Por exemplo, é possível usar os números de *feedbacks* negativos fornecidos nas respostas sobre uma entidade para avaliar se o contexto associado a esta entidade é adequado.

Existem também várias mudanças e comparações que podem ser exploradas em trabalhos futuros. A variação dos algoritmos de reconhecimento de entidade e algoritmo de predição da resposta e suas bases de treinamento podem gerar trabalho de análise comparativa da ferramenta para diferentes configurações. Por exemplo, a substituição do modelo BiDAF, proposto por Seo *et al* [8] pelo modelo *Dinamic memory Networks*, introduzido em Kumar *et al* [7]. Essa variação é relevante devido sua capacidade de gerar resposta constituída por palavras não pertencentes ao contexto.

Explorar outros métodos de estruturação e seleção de contexto é uma forma de ajudar a melhorar os resultados obtidos. O uso de diferentes contextos para uma mesma entidade pode trazer benefícios ao domínio de Inteligência Computacional, explorando diferentes aspectos da entidade em cada um dos múltiplos contextos.

Por fim, o domínio da segunda etapa também poderia ter sua lógica de seleção de contexto reformulada para levar em conta a frequência com que as entidades aparecem em cada contexto, ou seja, tornar mais relevantes as entidades exclusivas de um contexto.

Referências

[1] Allam, Ali e Mohamed Haggag: *The question answering systems: A survey.* 2:211–221, setembro 2012. 4, 5, 7

[2] Derici, Caner, Kerem Çelik, Ekrem Kutbay, Yiğit Aydın, Tunga Güngör, Arzucan Özgür e Günizi Kartal: *Question Analysis for a Closed Domain Question Answering System*, páginas 468–482. Springer International Publishing, Cham, 2015, ISBN 978-3-319-18117-2. https://doi.org/10.1007/978-3-319-18117-2_35. 4, 17, 19

[3] Cuteri, Bernardo: *Closed domain question answering for cultural heritage.* Em *Proceedings of the Doctoral Consortium of AI*IA 2016 co-located with the 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016), Genova, Italy, November 29, 2016.*, páginas 17–22, 2016. http://ceur-ws.org/Vol-1769/paper03.pdf. 4, 5, 17, 19

[4] P. Lende, Sweta e M M. Raghuwanshi: *Closed domain question answering system using nlp techniques.* janeiro 2016. 4, 6, 7, 17

[5] Chung, Hoojung, Young In Song, Kyoung Soo Han, Do Sang Yoon, Joo Young Lee, Hae Chang Rim e Soo Hong Kim: *A practical qa system in restricted domains.* Em Aliod, Diego Mollá e José Luis Vicedo (editores): *ACL 2004: Question Answering in Restricted Domains*, páginas 39–45, Barcelona, Spain, July 2004. Association for Computational Linguistics. 5, 6

[6] Lu, Yichao, Phillip Keung, Shaonan Zhang, Jason Sun e Vikas Bhardwaj: *A practical approach to dialogue response generation in closed domains.* CoRR, abs/1703.09439, 2017. http://arxiv.org/abs/1703.09439. 5

[7] Kumar, Ankit, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani e Richard Socher: *Ask me anything: Dynamic memory networks for natural language processing.* CoRR, abs/1506.07285, 2015. http://arxiv.org/abs/1506.07285. 5, 7, 11, 18, 56

[8] Seo, Min Joon, Aniruddha Kembhavi, Ali Farhadi e Hannaneh Hajishirzi: *Bidirectional attention flow for machine comprehension.* CoRR, abs/1611.01603, 2016. http://arxiv.org/abs/1611.01603. 5, 7, 12, 56

[9] *O que é um chatbot?* https://medium.com/botsbrasil/o-que-é-um-chatbot-7fa2897eac5d, acesso em 2019-01-18. 6

[10] Burges, Christopher J. C.: *Towards the machine comprehension of text: An essay*, 2013. 7

[11] Russell, Stuart e Peter Norvig: *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edição, 2009, ISBN 0136042597, 9780136042594. 7

[12] Rizkallah, Juliette: *The big (unstructured) data problem*. `https://www.forbes.com/sites/forbestechcouncil/2017/06/05/the-big-unstructured-data-problem/#bb5a15c493a3`, acesso em 2019-01-08. 7

[13] Weston, Jason, Antoine Bordes, Sumit Chopra e Tomas Mikolov: *Towards ai-complete question answering: A set of prerequisite toy tasks*. CoRR, abs/1502.05698, 2015. `http://arxiv.org/abs/1502.05698`. 8, 11

[14] Bird, Steven, Ewan Klein e Edward Loper: *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edição, 2009, ISBN 0596516495, 9780596516499. 8

[15] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard e David McClosky: *The Stanford CoreNLP natural language processing toolkit*. Em *Association for Computational Linguistics (ACL) System Demonstrations*, páginas 55–60, 2014. `http://www.aclweb.org/anthology/P/P14/P14-5010`. 8, 9

[16] *Conll2003*. `https://www.clips.uantwerpen.be/conll2003/ner/`, acesso em 2019-01-10. 9

[17] *Babelfy*. `http://babelfy.org/`, acesso em 2019-01-10. 9, 14, 15

[18] *Named entity recognition allennlp*. `https://demo.allennlp.org/named-entity-recognition`, acesso em 2019-01-10. 9, 14

[19] Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee e Luke Zettlemoyer: *Deep contextualized word representations*. CoRR, abs/1802.05365, 2018. `http://arxiv.org/abs/1802.05365`. 9

[20] Moro, Andrea, Alessandro Raganato e Roberto Navigli: *Entity linking meets word sense disambiguation: a unified approach*. Transactions of the Association for Computational Linguistics, 2:231–244, maio 2014. 9

[21] Mikolov, Tomas, Kai Chen, Gregory S. Corrado e Jeffrey Dean: *Efficient estimation of word representations in vector space*. CoRR, abs/1301.3781, 2013. 10

[22] *Vector representations of words - tutorial*. `https://www.tensorflow.org/tutorials/representation/word2vec`, acesso em 2019-01-10. 10

[23] Pennington, Jeffrey, Richard Socher e Christopher D. Manning: *Glove: Global vectors for word representation*. Em *Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543, 2014. `http://www.aclweb.org/anthology/D14-1162`. 11

[24] *Exploração do squad*. https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Nikola_Tesla.html, acesso em 2019-01-11. 12

[25] *Open calais*. http://www.opencalais.com/opencalais-demo/, acesso em 2019-01-12. 14

[26] *Lx-center*. http://lxcenter.di.fc.ul.pt/tools/en/LXSyllabifierEN.html, acesso em 2019-01-12. 14

[27] Cardoso, Nuno: *Rembrandt - a named-entity recognition framework*. Em *LREC*, 2012. 14

[28] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev e Percy Liang: *Squad: 100, 000+ questions for machine comprehension of text*. CoRR, abs/1606.05250, 2016. http://arxiv.org/abs/1606.05250. 17

[29] Hewitt, Steven: *Question answering with tensorflow*. https://www.oreilly.com/ideas/question-answering-with-tensorflow, acesso em 2019-02-01. 18

[30] *A free, open-source ontology editor and framework for building intelligent systems*. https://protege.stanford.edu/, acesso em 2019-02-01. 26

Apêndice A

Testes realizados

Pergunta	Resultado
Will I receve a **housing subsidy** from JSPS ?	Aproximada
There is **family allowance** for when my **family** comes to **Japan** ?	Aproximada
What is the **maximum period** of **temporary absence** ?	Certa
When my **maintenance allowance** being reduce ?	Errada
How many **days** I'm allowed to take of **leave of absence** ?	Certa
Do I need **JSPS** permission to take leave of absence for **research** purposes ?	Aproximada
Do I need any **immigration papers** before leaving **Japan**?	Aproximada
National health insurance is mandatory for **foreign residents** ?	Errada
Everyone needs a **national health insurance** plan ?	Aproximada
How many **days** I have to sign up for the **national health insurance** ?	Certa
How much will **national health insurance** cost me ?	Errada
Are **automobile accidents** coved by **JSPS insurance** ?	Errada
May I extend my **fellowship** to **finish up research** ?	Errada
What is the **limit** to submitting the **extension procedure**?	Aproximada
How long it for the **application aproval** ?	Certa
Will **JSPS** approve all **extension applications** ?	Certa
What is the **maximum number** of **certificates JSPS** can issue ?	Certa
Does **JSPS** issue a **certification letter** for seeking a new **position** ?	Aproximada
Can I get a **certificate** if I am continuing my **employment** at the same **institute** ?	Certa
Can **JSPS** issue a **certificate** with the title of my **research** on it ?	Errada
What is the **advantage** of getting a "**Professor**" **visa** ?	Certa
What are the basic **documents** I may need to submit to extend the **period** of my stay ?	Aproximada
How can I invite my family to Japan ?	Aproximada
Are **JSPS allowances tax** exempt ?	Errada
Which **option** should I **choose**, "self-employed"or "**employee**"or "other"?	Certa
For what **Research Travel Allowance** shouldn't be used for ?	Certa
Who makes the grants-in-aid **application** ?	Certa
What is the **maximum grant** I can receive in one year ?	Aproximada
When the **grant** is received ?	Certa
If **fellow** and **host** desagree who is to decide how to use the **grant**?	Errada
Who can help me with **problems** associated with living in **Japan** ?	Certa
Does **JSPS** cover "**key money**" (Rei-kin) ?	Errada
Can **JSPS** be my **guarantor** ?	Aproximada
Who is eligible for Japanese language training ?	Aproximada

Tabela A.1: Conjunto de perguntas e resultados usados para testar a performance da ferramenta para o domínio do documento da JSPS. Em negrito estão as entidade presentes em cada pergunta.