UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO GABRIEL CORTES

Quando, Onde, Quem, O que ou Por que? Um Modelo Híbrido de Classificação de Perguntas para Sistemas de Question Answering

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Dante A. C. Barone

Porto Alegre
2019

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Cortes, Eduardo Gabriel

Quando, Onde, Quem, O que ou Por que? Um Modelo Híbrido de Classificação de Perguntas para Sistemas de Question Answering / Eduardo Gabriel Cortes. – Porto Alegre: PPGC da UFRGS, 2019.

83 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Orientador: Dante A. C. Barone.

1. Classificação de texto. 2. Representação de texto. 3. Aprendizado de máquina. 4. Word2vec. I. Barone, Dante A. C.. II. Título.

Aos meus pais, os meus irmãos, a minha namorada e os meus amigos
que, com muito carinho e apoio, não mediram esforços para
que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

Gostaria de agradecer primeiramente aos meus pais, pelo incentivo aos estudos e a alcançar os meus objetivos. Agradeço à minha família, que sempre me deu apoio e incentivo na minha trajetória na carreira acadêmica.

Agradeço também à minha namorada, pelo carinho, companheirismo e por ter estado presente durante toda a minha jornada acadêmica. Agradeço aos meus amigos, que mesmo frente à ausência não deixaram de me apoiar.

Agradeço aos colegas de mestrado, que sempre tiveram algo a contribuir e me ensinar.

Agradeço aos professores que encontrei durante a minha jornada acadêmica, em especial às minhas orientadoras da graduação, professora Daniela Bagatini e Rejane Frozza, que me incentivaram e me deram suporte para que eu ingressasse no mestrado.

Agradeço ao CNPq pelos meses de bolsa de estudo durante o curso de mestrado, o que permitiu a dedicação exclusiva à pesquisa.

Gostaria de agradecer ao meu professor orientador, Dante Barone, por todo incentivo e suporte prestado, e ao colega Vinicius Woloszyn, pela disposição e apoio às pesquisas realizadas durante o curso.

RESUMO

Sistemas de *Question Answering* é um campo de pesquisa das áreas de Recuperação de Informações e Processamento de Linguagem Natural que propõe, de forma autônoma, responder perguntas feitas por humanos em linguagem natural. Uma das principais etapas desses sistemas é a classificação de perguntas, em que o sistema busca identificar o tipo de resposta que a pergunta se refere, facilitando a localização de informações específicas em sua base de dados. Comumente, modelos supervisionados de aprendizado de máquina são empregados nesta tarefa, em que o texto da pergunta é representado através de um vetor de características, como *Bag-of-words*, *Term Frequency-Inverse Document Frequency* (TF-IDF) ou *word embeddings*. Entretanto, a qualidade dos resultados produzidos por esses modelos são dependentes da existência de um grande conjunto de dados anotados para o treinamento, como também recursos computacionais e linguísticos externos. Esses recursos muitas vezes não estão acessíveis, devido a intensos esforços manuais na anotação de conjunto de dados ou pela falta de disponibilidade de recursos de qualidade para línguas não inglesa. Assim, este trabalho propõe uma abordagem híbrida para representação de texto que combina TF-IDF e *Word2vec* na tarefa de classificação de perguntas para sistemas de QA. Essa abordagem busca prover o tipo de resposta para perguntas em texto, utilizando diferentes tamanhos de conjuntos de treinamento com também sem a utilização de recursos computacionais e linguísticos complexos de serem adquiridos. Os experimentos realizados utilizando as coleções Chave e UIUC traduzida para o português, e variando o tamanho do conjunto de dados para treinamento, mostram estatisticamente que o modelo proposto atinge resultados satisfatório aplicado em diferentes modelos supervisionados.

Palavras-chave: Classificação de texto. representação de texto. aprendizado de máquina. word2vec.

When, Where, Who, What or Why? A Hybrid Question Classification Model for Question Answering Systems

ABSTRACT

Question Answering Systems is a field of Information Retrieval and Natural Language Processing that automatically answers questions posed by humans in a natural language. One of the main steps of these systems is the Question Classification, where the system tries to identify the type of question (i.e. if it is related to a person, time or a location) facilitate the generation of a precise answer. Machine learning techniques are commonly employed in tasks where the text is represented as a vector of features, such as Bag-of-words, Term Frequency-Inverse Document Frequency (TF-IDF) or word embeddings. However, the quality of results produced by supervised algorithms is dependent on the existence of a large, domain-dependent training dataset which sometimes is unavailable due to labor-intense of manual annotation of datasets or lack of availability of quality resources for non-English languages. In this work, we propose a hybrid model that combines TF-IDF and word embedding to provide the answer type to text questions using small and large training sets. Our experiments using the Chaves and UIUC translated for Portuguese datasets, using several different sizes of training sets, showed statistically that the proposed hybrid model reached promising results applied in different supervised models.

Keywords: Text classification, text representation, machine learning, word2vec.

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
CRF	*Conditional Random Field*
IR	*Information Retrieval*
LSTM	*Long Short-Term Memory*
MLP	Perceptron Multicamadas
NILC	*Interinstitutional Center for Computational Linguistics*
PLN	Processamento de Linguagem Natural
POS	*Part-of-Speech*
QA	*Question Answering*
REM	Reconhecimento de Entidades Mencionadas
RNA	Redes Neurais Artificiais
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency – Inverse Document Frequency*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um sistema de *Question Answering* (QA), também conhecido como sistema de Resposta Automática a Perguntas, é uma ferramenta computacional capaz de responder perguntas em linguagem natural. Diferente de sistemas de Recuperação de Informações tradicionais, em que o usuário deve fornecer palavras-chave e buscar em uma lista de documentos a informação desejada, os sistemas de QA têm a vantagem de fornecer como saída uma resposta precisa a uma pergunta de entrada. Deste modo, ter a entrada de dados na forma de uma pergunta em linguagem natural torna o sistema mais amigável, porém mais difícil de ser implementado.

Para que o sistema de QA consiga responder a perguntas de forma concisa e exata, é preciso tratar diversos problemas relacionados às áreas de Processamento de Linguagem Natural (PLN), Aprendizado de Máquina, Recuperação de Informações e Extração de Informação (SASIKUMAR; SINDHU, 2014). Normalmente, a arquitetura desses sistemas é composta dos seguintes módulos (JURAFSKY; MARTIN, 2014): I) Processamento da Pergunta: responsável por extrair informações da pergunta de entrada, como palavras-chave e tipo de resposta; II) Recuperação de Informações: responsável por buscar informações na sua base de dados para formular a resposta; e III) Processamento da Resposta: responsável por gerar a resposta de saída em linguagem natural através das informações das etapas anteriores.

Um dos primeiros sistemas de QA foi o sistema de domínio restrito *BASEBALL* (JR et al., 1961), que respondia perguntas sobre uma única temporada do esporte. Também, o sistema *LUNAR* de 1977, permitia a geologistas realizarem perguntas sobre o domínio de rochas. Mais recentemente, o sistema Watson da IBM, de domínio geral, conseguiu o feito de ganhar de humanos no *game-show Jeopardy!* em 2011. Além disso, esses sistemas podem ser divididos em duas classes, as de domínio específico (CAO et al., 2011), que devem ser capazes de responder perguntas exclusivamente de um domínio específico de conhecimento, e os sistemas de amplo domínio, que buscam responder a perguntas de um espectro amplo de conhecimento (MISHRA; JAIN, 2016).

A classificação de perguntas, ou reconhecimento do tipo de resposta é uma etapa importante do módulo de Processamento da Pergunta, que busca fornecer orientação significativa sobre a natureza da resposta requerida (LONI, 2011). A tarefa consiste em determinar a classe de pergunta, normalmente relacionado ao 5WH, acrônimos usado na língua inglesa para os principais tipos de perguntas (*Who?*, *What?*, *Where?*, *When?* e *Why?*). Por exemplo, a pergunta "Quem ganhou o último prêmio Nobel da Paz?"espera uma resposta da classe PESSOA, enquanto que a pergunta "Quando o homem pisou pela primeira vez na Lua?"espera uma resposta da classe TEMPO. Uma vez que a classe da pergunta é determinada, essa informação será usada nos próximos estágios do sistema (JURAFSKY; MARTIN, 2014). De acordo com (LONI, 2011), essa etapa é crucial para determinar a estratégia de busca de passagens de texto no módulo de Recuperação de Informações, já que todo o processo de extração de resposta depende da localização de entidades da mesma classe da pergunta.

Abordagens endereçadas à etapa de classificação de perguntas geralmente dependem de modelos baseados em regras manuais ou aprendizado supervisionado. Em modelos baseados em regras, são criadas manualmente regras através de observações empíricas de termos e classes gramaticais das perguntas, de modo a determinar padrões no texto associado à cada classe (HOVY; HERMJAKOB; RAVICHANDRAN, 2002; JURAFSKY; MARTIN, 2014). No entanto, a linguagem natural possui

um grande número de regras e exceções, o que demanda grandes esforços na criação dessas regras. Por outro lado, abordagens baseadas em aprendizado de máquina, como *Support Vector Machine* (SVM) e *Long Short-Term Memory* (LSTM), vêm apresentando excelentes resultados na tarefa de classificação de perguntas (LEE; DERNONCOURT, 2016; SARROUTI; ALAOUI, 2017; MA et al., 2017). No entanto, a qualidade dos resultados produzidos por esses algoritmos supervisionados são altamente dependente da existência de um grande conjunto de dados de treinamento, como também de recursos linguísticos externos.

1.1 Problema de pesquisa

Em tarefas de classificação de texto, a qualidade dos resultados produzidos por modelos supervisionados de aprendizado de máquina dependem da existência de um grande conjunto de dados para o treinamento. Além disso, o desempenho desses modelos também se fundam na utilização de características linguísticas, em que é necessária a obtenção de recursos computacionais e linguísticos externos. Comumente, encontra-se dificuldade para encontrar esses recursos com boa qualidade para línguas não inglesa.

De modo a superar o problema de baixo recursos disponíveis para línguas não inglesa, o presente trabalho de pesquisa propõe uma abordagem híbrida para a etapa de classificação de perguntas de sistemas de QA, de modo a não depender da qualidade de recursos externos, como um parser sintático ou uma ontologia. Desta forma, o trabalho objetiva-se a responder a seguinte questão de pesquisa através da hipótese proposta:

- **Questão:** A união de características lexicais e semânticas do texto de perguntas, aplicada em modelos supervisionados, são suficientes para atingir resultados satisfatórios na tarefa de classificação de perguntas em sistemas de QA, variando o tamanho do conjunto de treinamento?

- **Hipótese:** O presente trabalho propõe a utilização de um modelo híbrido de representação de texto para a tarefa de classificação de perguntas em sistemas de QA. O modelo híbrido proposto combina características lexicais, através de um vetor TF-IDF, e características semânticas, através de um vetor *Word2vec*, para representar o texto de perguntas. Ambas características não exigem recursos relativamente difíceis de conquistar, independente da língua. Portanto, espera-se que a união dessas características sejam suficientes para que modelos supervisionados de aprendizado de máquina atinjam resultados satisfatórios na tarefa, variando o tamanho do conjunto de treinamento.

A principal coleção utilizada para treinar e testar a abordagem proposta é a coleção Chave (MAGNINI et al., 2006). Essa é uma base de dados com perguntas e respostas na língua portuguesa composta de aproximadamente 4.000 perguntas, sendo que cerca de 1.000 dessas, contém ao menos uma resposta. Além disso, a coleção oferece cerca de 210.000 documentos de texto bruto para serem utilizados como base de conhecimento de um sistema de QA. Todas as perguntas da coleção podem ser respondidas com informações retiradas desses documentos.

Testa-se a abordagem proposta aplicando-a em diferentes classificadores, variando o tamanho do conjunto de treinamento, de modo a verificar seu desempenho em diferentes tamanhos de dados

para treinamento. Além disso, a abordagem é testada em um sistema de QA desenvolvido para a língua portuguesa. Assim, foi possível verificar a importância da etapa de classificação de perguntas em um sistema completo de QA. Os resultados são comparados com outras abordagens *baselines* de representação de texto.

1.2 Objetivos e Contribuições

O objetivo geral deste trabalho de pesquisa é a proposta de um modelo híbrido para a tarefa de classificação de perguntas em sistemas de QA. A abordagem une características lexicais, através de um vetor TF-IDF, e semânticas, através de um vetor *Word2vec*. As abordagens são aplicadas e testadas nos modelos supervisionados *Support Vector Machine* (SVM), *Multilayer perceptron* (MLP), e *Long short-term memory* (LSTM). Além disso, de modo a verificar o impacto da abordagem em um sistema completo de QA, é desenvolvido um sistema de QA para a língua portuguesa capaz de responder perguntas do tipo factuais e de amplo domínio de conhecimento. Deste modo, as contribuições resultantes dessa pesquisa estão listadas a baixo:

- Um levantamento bibliográfico sobre sistemas de QA, organizando e comparando as diferentes abordagens propostas na literatura.
- Uma comparação entre as abordagens de representação de texto *Bag-of-words* binário, *Bag-of-words* com TF-IDF e *Word2Vec* para a tarefa de classificação de perguntas usando coleções em Português e variando o tamanho do conjunto de treinamento.
- Uma comparação entre os modelos supervisionados SVM, MLP e LSTM para a tarefa de classificação de perguntas usando coleções em Português, aplicando diferentes abordagens de representação de texto e variando o tamanho do conjunto de treinamento.
- Uma abordagem híbrida para a tarefa de classificação de perguntas que estatisticamente apresenta resultados satisfatórios em diferentes modelos supervisionados e em diferente tamanhos de conjunto de treinamento.
- Um teste completo de um sistema de QA para língua portuguesa utilizando diferentes estratégias de representação de texto para a etapa de classificação de perguntas.

Além desse capítulo introdutório, este trabalho apresenta no Capítulo 2 a contextualização dos principais assuntos. No Capítulo 3, apresenta-se os trabalhos relacionados e uma descrição mais detalhada da abordagem híbrida proposta. No Capítulo 4, dispõe-se a metodologia de pesquisa. O Capítulo 5 apresenta os resultados obtidos e suas análises. Por fim, o Capítulo 6 realiza as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo destina-se a discutir a fundamentação teórica dos assuntos tratados nesta pesquisa. Serão abordadas as principais etapas de um sistema de QA conforme a arquitetura normalmente encontrada na literatura. Também apresentam-se as abordagens de representação de textos *Bag-of-words* binário, *Bag-of-words* com TF-IDF e *Word Embedding*. Por fim, discutem-se os principais modelos utilizados de aprendizado de máquina supervisionado para classificação de pergunta de texto.

2.1 Sistemas de QA

QA é um desafio da ciência da computação que tem o propósito de providenciar uma resposta precisa para uma pergunta em linguagem natural colocada por um humano. Para atingir esse objetivo, sistemas de QA precisam extrair informações necessárias para compreender a pergunta de entrada e buscar dados relevantes para responder a pergunta em uma base de dados. Por fim, o sistema deve gerar a resposta de saída a partir dessas informações.

Sistemas de QA podem ser classificados conforme o tipo de pergunta que o mesmo busca estar apto a responder e pela abrangência de domínio de conhecimento. O tipo da pergunta a ser respondida pode determinar a complexidade de um sistema de QA. Conforme, (KOLOMIYETS; MOENS, 2011), perguntas podem ser dividas em:

- Factoides: requerem como resposta um fato expressado no corpo do texto;
- Listas: requerem como resposta uma lista de entidades ou fatos;
- Definição: requerem uma definição, descrição ou opinião sobre algo solicitado na pergunta;
- Hipotéticas: requerem informações sobre um evento hipotético;
- Casuais: requerem explicações (Por quê?) de um evento ou fato;
- Relacionais: requerem informações sobre a relação entre duas entidades;
- Procedurais: requerem como resposta uma lista de instruções para realizar uma determinada tarefa;
- Confirmação: requerem como resposta um Sim ou um Não para um evento expressado na pergunta.

Se um sistema de QA objetiva-se em responder perguntas de um domínio específico de conhecimento, esse pode ser classificado como um sistema de domínio restrito. Normalmente esses sistemas empregam base de dados que buscam estruturar o conhecimento, o que facilita o processo de consulta, como um banco de dados relacional ou uma ontologia (CAO et al., 2011). Uma vez que o domínio de conhecimento é restringido, a quantidade de dados é menor, o que possibilita inserir conhecimento aprofundado sobre o assunto (BAO et al., 2014; SCHULZE et al., 2016). Já os sistemas de amplo domínio, objetivam-se em responder perguntas de qualquer área de conhecimento.

Normalmente esses sistemas trabalham com bases de conhecimentos não estruturadas de texto bruto. Uma vez que as perguntas abrangem um grande espectro de conhecimento, fica custoso modelar toda a informação disponível (MISHRA; JAIN, 2016).

Uma das características mais importantes de um sistema de QA é a quantidade de informação disposta em sua base de dados (LIN, 2002). A quantidade disposta pode ter grande influência em seu desempenho, uma vez que o sistema não é capaz de responder a uma pergunta se não tiver as informações necessárias sobre ela. Uma base de dados com um grande número de informações traz maior redundância de dados, característica positiva para o desempenho do sistema, que traz vantagens como:

- A informação redundante pode ser mais confiável, já que a informação de um único documento pode estar com um texto mal escrito ou com dados incorretos.
- Uma mesma informação disposta em diferentes formas pode facilitar a compreensão do texto a partir de técnicas de PLN mais complexas.

Coleção de documentos pré-selecionados podem ser utilizadas como base de dados de um sistemas de QA. Nessa abordagem, busca-se selecionar previamente o maior número possível de documentos de texto para servirem como fontes de informação. Esses documentos, normalmente, são criados utilizando fontes de informação com credibilidade, como textos de artigos acadêmicos ou notícias de jornais. Esses sistemas apresentam uma coleção de documentos de maior qualidade, porém, muitas vezes com quantidade menor de informações, comparado com sistemas que utilizam diretamente a web como base de dados, o que restringe a quantidade de perguntas que o sistema é capaz de responder (ABBAS et al., 2016).

A utilização da web como base de dados é uma abordagem que oferece uma quantidade massiva de informações. Sistemas de QA baseados em web normalmente utilizam um motor de busca para recuperar páginas web com relação à pergunta realizada, e então, extrai as informações necessárias de seu conteúdo (LIN, 2002). A grande quantidade de dados disposta na web permite que esses sistemas consigam responder uma quantidade maior de perguntas, como também, perguntas mais específicas sobre um determinado assunto (LEHMANN et al., 2012). Diferente de uma coleção de documentos pré-selecionados, a web permite que qualquer pessoa inclua ou edite informações, o que torna comum encontrar dados de baixa qualidade, como informações incorretas ou falsas (*Fake News*). Desta forma, muitos sistemas delimitam quais sites podem ser utilizados e quais não.

Diferentes trabalhos na literatura objetivam-se a identificar e tratar os problemas de dados de baixa qualidade encontrados na web. É possível verificar a corretude de uma informação utilizando checagem de fatos, em que uma declaração de fato é representada através de uma tupla sujeito-predicado-objeto e verificada em uma base de dados estruturadas (ontologias) com informações confiáveis (CIAMPAGLIA et al., 2015). Além disso, outras abordagens buscam classificar quando uma determinada informação é falsa ou verdadeira, utilizando apenas características linguísticas do texto (MONTEIRO et al., 2018).

Conforme (JURAFSKY; MARTIN, 2014; LONI, 2011), sistemas de QA normalmente utilizam de três módulos para compreender a pergunta de entrada, buscar informações para respondê-la e criar uma resposta final de saída. A Figura 2.1, demonstra o fluxo de processamento de uma arquitetura convencional. Esses módulos podem ser melhores definidos como:

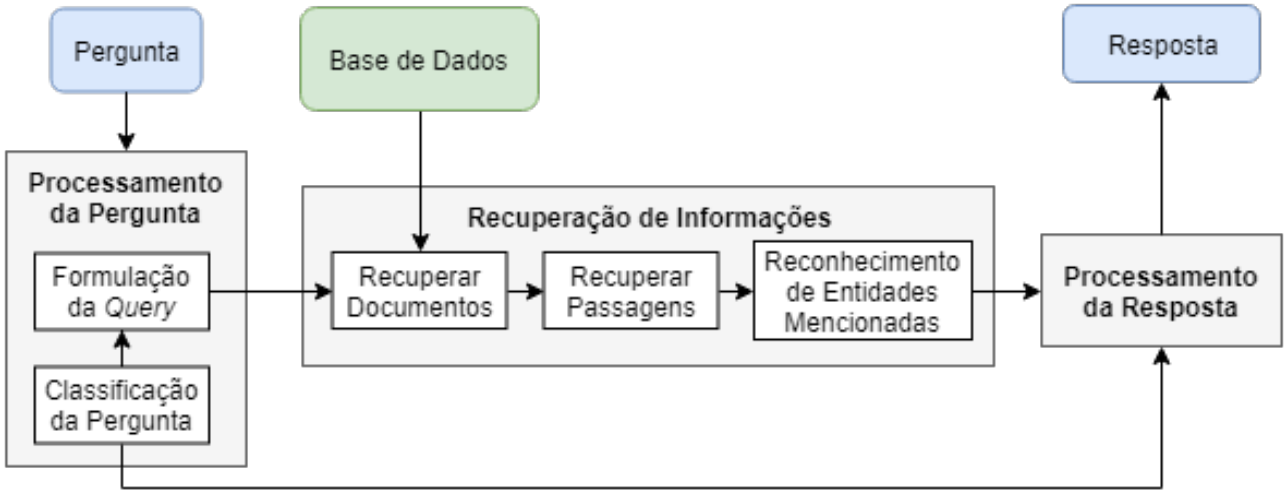


Figura 2.1: Arquitetura de um sistema de QA
Fonte: O Autor

- **Processamento da Pergunta:** O módulo tem como principal tarefa a extração de informações da pergunta de entrada, de modo a identificar o tipo de resposta para a pergunta, através da etapa de classificação de perguntas, e selecionar termos chave para consulta, através da etapa de formulação da *query*.

- **Recuperação de Informações:** Esse módulo tem como objetivo buscar as informações necessárias para responder a pergunta de entrada. Desta forma, utiliza-se uma etapa para recuperar documentos relevantes, e a partir desses, recuperam-se sentenças de texto com a etapa de recuperação de sentenças. Por fim, aplicam-se modelos de reconhecimento de entidades mencionadas (REM), também conhecidos como modelos de reconhecimento de entidades nomeadas, para identificar entidades nos textos das sentenças.

- **Processamento da Resposta:** O módulo final fica responsável por processar as sentenças de texto obtidas no módulo anterior e extrair segmentos de palavras que formarão a resposta de saída do sistema. Além das passagens, pode-se utilizar a classe da pergunta obtida na etapa de Processamento da Pergunta para determinar a estratégia de geração da resposta. Por fim, a etapa gera diferentes respostas e as ordena pela semelhança com a pergunta. Ao final, o sistema retorna a melhor resposta do ranque.

A Figura 2.2 demonstra um exemplo de como funciona o fluxo de processamento e saída dos dados de cada etapa da arquitetura do sistema de QA. A entrada de dados é a pergunta "Quem descobriu o Brasil?". Na primeira etapa, são extraídos os termos chave para formar a *query* e em seguida a pergunta é classificada como do tipo PESSOA, já que se espera uma pessoa como resposta. Na próxima etapa são recuperados documentos no contexto da pergunta e extraídos dessas, sentenças específicas e relevantes para a resposta. Por fim, é realizado o reconhecimento de entidades mencionadas, em que entidades do tipo PESSOA estão destacadas em azul, enquanto entidades do tipo LUGAR estão destacadas em verde no exemplo. A partir das entidades identificadas e classificadas,

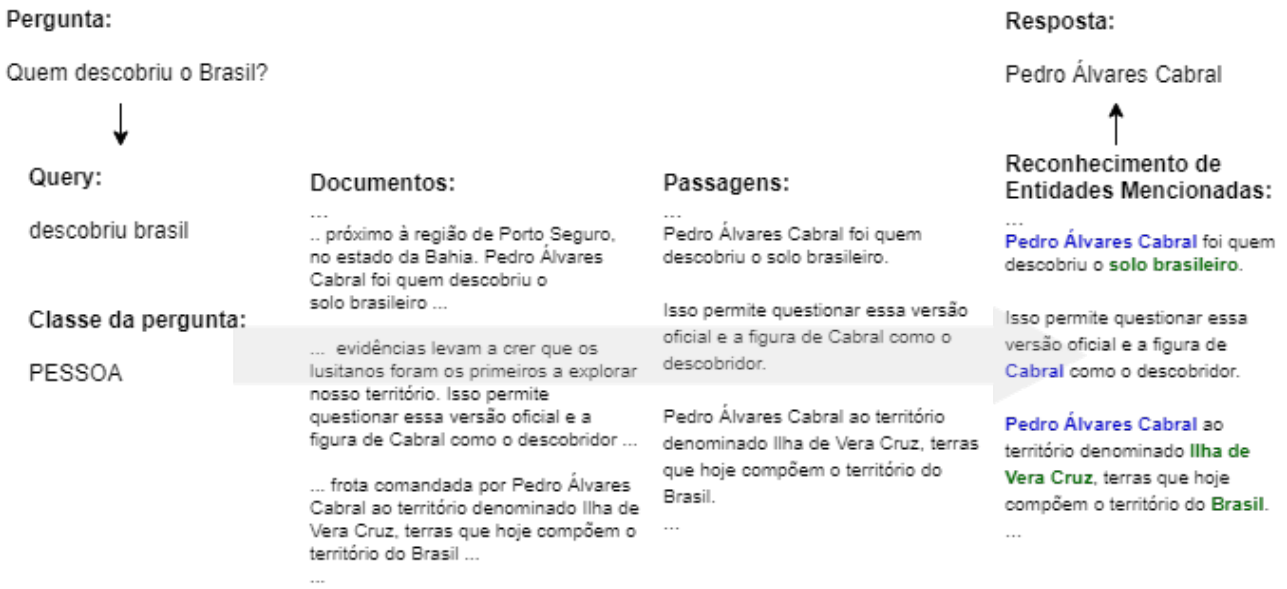


Figura 2.2: Exemplo do fluxo de processamento e saída das etapas de um sistema de QA
Fonte: O Autor

é criado um ranque de respostas candidatas e retorna-se a melhor classificada. No caso do exemplo, foi retornada a resposta final "Pedro Álvares Cabral". As próximas subseções apresentam com mais profundidade as etapas de cada módulo de um sistema de QA.

2.1.1 Processamento da Pergunta

Uma vez que o usuário submete uma pergunta ao sistema de QA, a primeira tarefa que o sistema deve cumprir é o entendimento da pergunta de entrada. Esse entendimento é realizado através de etapas que, comumente, dedicam-se a extrair aspectos importantes da pergunta, como palavras-chave e o tipo de informação que está sendo solicitando. Na literatura, encontram-se principalmente sistemas que utilizam duas etapas no módulo de Processamento da Pergunta, a classificação de perguntas e a formulação de *query* (AMARAL et al., 2008; LONI, 2011; SANGODIAH; MUNIANDY; HENG, 2015; ABBAS et al., 2016). Assim, assume-se que o módulo tem como principal tarefa a extração de informações da pergunta de entrada, de modo a identificar o tipo de resposta para a pergunta através da etapa de classificação de pergunta, e selecionar termos chave para consulta através da etapa de formulação da *query*.

Para identificar qual o tipo de resposta que a pergunta de entrada requer, o sistema de QA utiliza a etapa de classificação de pergunta. Essa etapa busca atribuir uma classe para a pergunta de acordo com o tipo de informação solicitada. Por exemplo, a pergunta “Qual a capital do Brasil?"requer uma localização, logo pode ser classificada como LOCAL. Já a pergunta “Quem descobriu o Brasil?"requer como resposta uma pessoa, logo pode ser classificada como PESSOA. A classe da pergunta é uma informação importante para as demais etapas do sistema de QA, em que essa pode

ser utilizada principalmente para especificar o tipo de informação a ser extraída de documentos de texto (JURAFSKY; MARTIN, 2014).

As primeiras abordagens a serem utilizadas foram as baseadas em regras, em que se criam regras manuais através de palavras ou notações sintáticas para determinar a classe de pergunta (HOVY; HERMJAKOB; RAVICHANDRAN, 2002). O problema dessa abordagem é a grande demanda de esforços para criar regras que abrangem todas as situações que a linguagem natural humana pode proporcionar. Deste modo, abordagens baseadas em aprendizado de máquina ganharam espaço na resolução do problema, em que se utilizam conjuntos de perguntas com classes anotadas para treinar modelos supervisionados (LAHBARI; OUATIK; ZIDANI, 2017).

A formulação de uma *query* é uma etapa importante que busca extrair palavras-chave do texto da pergunta de entrada, de modo a fornecer uma *query* para o módulo de Recuperação de Informações. Diferentes abordagens podem ser utilizadas para a criação de *queries*. Uma das principais abordagens é a remoção de *stopwords* do texto de modo a utilizar apenas as palavras restantes. Também, pode-se utilizar um parser sintático para selecionar palavras com classes gramaticais relevantes. Outra estratégia é a utilização de REM para extrair entidades mencionadas na pergunta. Por fim, é possível utilizar a própria classe da pergunta para determinar quais são os termos mais significativos para a *query* (LONI, 2011).

2.1.2 Recuperação de Informações

O módulo de Recuperação de Informações tem a tarefa de acessar a base de dados para consultar e extrair informações relevantes a uma *query* de entrada. Estas informações serão utilizadas para criar a resposta final do sistema de QA. Comumente, esse módulo é dividido em três etapas: a primeira é responsável por recuperar documentos relevantes da base de dados; a segunda extrai sentenças de texto a partir dos documentos recuperados; por fim, a terceira etapa identifica entidades mencionadas nos textos das sentenças através de um modelo de REM (KOLOMIYETS; MOENS, 2011).

A etapa de recuperação de documentos utiliza a *query* formulada na etapa de criação de *query* do módulo de Processamento da Pergunta. Através de um motor de busca, essa *query* é utilizada para recuperar documentos indexados em uma base de dados utilizada como fonte de informação do sistema. Essa base de dados pode ser uma coleção local de documentos de texto bruto, uma base de conhecimento de dados estruturados, como uma ontologia, ou até mesmo a web. A saída da etapa será um conjunto de documentos possivelmente relevantes a pergunta de entrada do sistema. Esses documentos são ranqueados, de modo que os mais relevantes fiquem nas primeiras posições.

Comumente, um documento contém uma grande quantidade de texto, enquanto que a pergunta de entrada normalmente requer uma informação mais específica. Desta forma, a grande maioria do texto de um documento não precisa ser processado para a extração de informação. Na etapa de recuperação de sentenças, os documentos são divididos em passagens de texto e mantidas aquelas que tenham maior relação com a pergunta. A relação pode ser definida pela proximidade dos termos da pergunta com a sentença ou verificar se a sentença contém uma entidade mencionada da mesma classe da pergunta. Por fim, a etapa tem como saída um ranque de passagens ordenado por um *score* de relevância.

O REM é uma das etapas do módulo de Recuperação de Informação que tem como tarefa a identificação e classificação das entidades encontradas no texto. Normalmente, essa etapa é utilizada para remover da lista de passagens aquelas que não contenham ao menos uma entidade da mesma classe da pergunta, o que reduz a quantidade de sentenças de texto a serem processadas pelo módulo de Processamento da Resposta. Além disso, pode-se aplicar REM em outras etapas do sistema, como na formulação da *query*. As abordagens de estado da arte encontradas na literatura utilizam modelos de aprendizado de máquina. Deste modo, utiliza-se corpus com entidades anotadas para treinar um modelo supervisionado capaz de identificar e classificar novas entidades (CHIU; NICHOLS, 2016).

Uma entidade mencionada pode ser composta por mais de uma palavra. Estas entidades são conhecidas como entidades mencionadas compostas. O Harem utiliza a notação *BIO* em suas classes. Deste modo, utiliza-se: a marcação *B* (*begin*) para palavras que estão no início da entidade; a marcação *I* (*inside*) para palavras que estão no meio e no fim da entidade; e a classe *O* (*outside*) para palavras que não são entidades. Por exemplo, a frase “João mora em Porto Alegre e estuda na UFRGS"pode ser representada através da lista de entidades “NOME-B, O, O, LOCAL-B, LOCAL-I, O, O, O, ORGANIZACAO-B".

2.1.3 Processamento da Resposta

Após o sistema recuperar sentenças de textos com as informações necessárias para responder a pergunta, o módulo de Processamento da Resposta é aplicado para extrair a resposta final de saída do sistema. As etapas desse módulo normalmente variam de acordo com os tipos de perguntas que o sistema suporta e a estratégia adotada para formulá-las. Sistemas que trabalham apenas com perguntas do tipo factóide normalmente contém um módulo de Processamento da Resposta mais simples. Já os que trabalham com perguntas do tipo definição, contém um módulo mais complexo, uma vez que esses precisam gerar textos ao invés de apenas extrair e retornar uma entidade mencionada do texto.

Uma das abordagens utilizada para extrair informações de uma passagem de texto é através de compreensão de leitura. O objetivo da abordagem é o entendimento de linguagem natural, de modo a permitir que o sistema seja capaz de extrair informações de uma sentença e sintetizar a resposta final do sistema através dessas informações. Para isso, normalmente a abordagem utiliza modelos supervisionados de redes neurais profundas. Esses modelos são treinados utilizando perguntas atreladas às passagens de textos e sua resposta (WANG; YAN; WU, 2018).

Para sistemas que trabalham exclusivamente com perguntas do tipo factóide, o módulo de Processamento da Resposta pode basear-se principalmente na criação de um ranque de respostas candidatas. Esse ranque é criado através de entidades mencionadas da mesma classe da pergunta que foram extraídas das sentenças de texto recuperadas. Para cada resposta candidata, calcula-se um *score* que determina sua posição no ranque. Esse *score* pode ser gerado através de diferentes critérios, como a frequência que a resposta candidata é encontrada ou pela proximidade da entidade no documento com demais termos contidos na pergunta (AMARAL et al., 2008). Sistemas que trabalham com perguntas do tipo definição também podem utilizar um ranque de respostas candidatas, porém, ao invés de extrair apenas entidades mencionadas de textos, esses podem utilizar templates de respostas para formular a resposta de saída do sistema. A própria classe da pergunta pode determinar a estratégia de geração da resposta. Por fim, o sistema retorna como resposta final a melhor colocada no ranque

de candidatas.

2.2 Representação de Texto para Aprendizado de Máquina

Classificação de texto é uma importante tarefa da área de aprendizado de máquina. O objetivo é atribuir uma ou mais classes para uma respectiva entrada de texto. Existem diversas aplicações da tarefa, como em filtragem de *spam*, análise de sentimentos, classificação de perguntas, etc. Modelos supervisionados de aprendizado de máquina utilizam vetores de conteúdo numéricos para representar instâncias de treinamento e predição. Na tarefa de classificação de texto, antes de classificar o texto, é necessário estruturá-lo em um vetor numérico. Deste modo, esta seção destina-se a apresentar as principais abordagens para representação de texto em vetores numéricos.

Uma das principais abordagens utilizadas para representar texto é através da técnica *Bag-of-words* (saco de palavras). A abordagem é um modelo convencional e simplificado de representação de texto usado no contexto de PLN e aprendizado de máquina. Um texto (documento, parágrafo, frase, etc.) é representado com *Bag-of-words* através de um conjunto de suas palavras, ou seja, o texto é representado através de um vetor numérico em que cada posição desse vetor representa uma palavra do vocabulário. Normalmente, cada elemento desse vetor recebe um valor binário ou sua frequência referente a ocorrência da palavra no texto em questão. Deste modo, a abordagem permite representar características lexicais do texto, porém, mesmo mantendo a multiplicidade, desconsidera a gramática e a ordem de palavras. Mesmo sendo uma abordagem simples, *Bag-of-words* é um modelo de representação de texto que normalmente atinge resultados consideráveis em tarefas de classificação de texto, mostrando-se uma abordagem *baseline* difícil de ser superada.

Embora comumente o vetor da abordagem *Bag-of-words* seja representado pela ocorrência das palavras, outra abordagem comum é a utilização do peso TF-IDF para representar a ocorrência de uma palavra. Assim, ao invés de empregar um vetor de multiplicidade, aplica-se a relação da frequência do termo inverso da frequência nos documentos. A abordagem permite atribuir maior peso a palavras mais raras do vocabulário do que palavras comuns. Por exemplo, a palavra "cardiologista"tem maior peso do que a palavra "melhor". O peso TF-IDF de uma palavra w em um documento d pode ser representado através da Equação 2.1, em que $freq_{f,d}$ é a frequência f da palavra w no documento d, N é a quantidade total de documentos da coleção e n_f é o número de documentos que contém a palavra w.

$$w_{f,d} = freq_{f,d} \times log_{10}\frac{N}{n_f} \tag{2.1}$$

Um típico problema da abordagem *Bag-of-words* é o grande número de dimensões que o vetor de representação pode chegar. Isso deve-se ao vasto vocabulário que as linguagens naturais humanas contém, em que um vocabulário pode ultrapassar facilmente as 50 mil palavras, dependendo do corpus utilizado. Outra desvantagem da abordagem é dificuldade da representação semântica do texto. Por exemplo, palavras sinônimas são diferenciadas da mesma forma do que duas palavras antônimas (SCOTT; MATWIN, 1998). Abordagens alternativas que busca suprir estas desvantagens são as baseadas em *word embedding* (incorporação de palavras). Estas abordagens representam cada palavra através de um vetor de tamanho fixo, que representa sua posição em um espaço semântico.

Um das principais abordagens baseadas em *word embedding* é o *Word2vec* (MIKOLOV et al., 2013). A abordagem consiste em um grupo de modelos de redes neurais rasas compostas por duas camadas de neurônios de modo a reproduzir uma retratação *word embedding*. Esses modelos recebem como entrada um grande corpus de texto em que reproduzem um espaço vetorial semântico. Assim, cada palavra do vocabulário recebe um vetor numérico que representa sua posição no espaço. Portanto, a abordagem tem a vantagem de representar palavras através de um vetor de tamanho fixo, normalmente com algumas centenas de dimensões. Além disso, a abordagem tem facilidade em representar semanticamente palavras. Por exemplo, normalmente palavras sinônimas estão mais próximas neste espaço semântico.

2.3 Modelos Supervisionados

Esta seção destina-se a apresentar os principais modelos supervisionados de aprendizado de máquina para classificação de perguntas. Os modelos apresentados foram selecionados segundo seu desempenho em tarefas de classificação de texto. Além disso, os modelos devem ser capazes de manipular as estruturas de dados utilizados na abordagem híbrida proposta neste trabalho, como também nas abordagens de *baseline*. Deste modo, foram selecionados os modelos SVM, MLP e LSTM. Os modelos *Naive Bayes Bernoulli* e *Naive Bayes Multinomial* não foram utilizados, pois o primeiro não manipula valores negativos, sendo esses encontrados em abordagens que utilizam *Word2vec*, e o segundo não manipula valores flutuantes, que são encontrados em abordagens que utilizam *Word2vec* e TF-IDF. Por fim, os modelos apresentados são os mesmos utilizados nos testes de classificação de pergunta discutidos no Capítulo 5.

2.3.1 *Support Vector Machine*

Support Vector Machine (SVM) é um modelo classificador linear binário que pode ser entendido como uma representação de pontos no espaço mapeados de forma que os pontos de cada categoria estejam divididos por um espaço que seja o mais amplo possível. Assim, novos pontos serão mapeados no mesmo espaço classificados pertencentes a uma categoria baseada em qual lado do espaço ele foi alocado. Portanto, um modelo SVM busca uma linha de separação (hiperplano) entre pontos de duas classes (CORTES; VAPNIK, 1995).

Um dos problemas do SVM convencional é a sua característica linear, pois em problemas do mundo real muitos padrões não serão linearmente separáveis. Assim, utilizam-se funções não reais para permitir que o modelo trabalho com dados que não são linearmente divisíveis. A partir disso, o SVM não-linear pode realiza mudanças de dimensionalidade, por meio das funções de kernel, como Sigmoide, RBF e Polinomial, para permitir que o modelo trate o problema como se fosse linear (MEYER; WIEN, 2001).

O SVM, sendo um modelo binário, classifica uma instância em apenas duas classes distintas. Porém, muito problemas apresentam mais de duas classes. Portanto, para acrescentar a característica de multiclasse para o modelo SVM geralmente segue-se a abordagem de decompor um problema multiclasse em um problema binário, chamadas de decomposição Um-Contra-Todos e Todos-Contra-

Todos. A primeira cria para cada classe um modelo SVM binário entre o espaço da classe e o restante do espaço pertencente as outras classes. Assim, ao levar em consideração todo o conjunto de modelos, é possível realizar classificações multiclasse. A segunda decomposição, ao invés de cria um novo modelo SVM para cada classe, cria para cada possível par de classe um novo modelo SVM permitindo assim todas as classes competirem entre si (HSU; LIN, 2002).

2.3.2 Perceptron Multicamadas

Redes Neurais Artificiais (RNA) é uma categoria de modelos de aprendizado de máquina baseados em uma coleção de neurônios artificiais conectados. A atividade desses modelos é fundamentada no funcionamento do cérebro, de modo que as conexões podem transmitir sinais de um neurônio artificial para outro, baseando-se nas sinapses de um cérebro biológico. Em uma RNA, o sinal entre neurônios artificiais são números reais, em que o valor de saída de cada neurônio é calculado através de uma função não linear chamada de função de ativação (SEBASTIANI, 2002).

As conexões entre neurônios artificias contém um valor de peso que é ajustado no processo de aprendizagem da RNA, de modo a ponderar os sinais sinápticos transmitidos através dessas conexões. Além disso, os neurônios artificias contém um valor limiar que determina se o sinal retornado através da função de ativação deve ser propagado. Em uma RNA, os neurônios normalmente estão distribuídos em camadas. Deste modo, o sinal recebido da camada de entrada é propagado nas camadas intermediárias (camadas ocultas), até a camada de saída.

Embora uma RNA tenha o objetivo de resolver problemas da mesma forma de um cérebro biológico, trabalhos da literatura buscam aplicar o modelo para resolver problemas específicos. O modelo vem sendo utilizado em problemas de reconhecimento de fala, tradução, classificação de texto, diagnóstico médico, visão computacional, etc. Deste modo, utiliza-se o modelo treinando especificamente com uma coleção de dados anotados associados ao problema em questão, sem a necessidade da criação de regras manuais (YANG; PEDERSEN, 1997). Por fim, uma das principais característica do modelo RNA é a capacidade de aprendizagem. Algoritmos como o *backpropagation* (retropropagação) permite o treinamento de redes neurais multicamadas de forma viável e eficiente. O *backpropagation* repropaga o sinal de erro da camada de saída, calculado através de uma função de custo, até a camada de entrada, ajustando os pesos sinápticos.

Uma rede Perceptron Multicamadas (MLP) é um modelo de RNA comumente aplicado em diversos problemas de aprendizado supervisionados. Esse modelo tem a característica de conter ao menos uma camada oculta, em que os sinais são propagados de forma direta pela rede, ou seja, não há propagação cíclica. Além disso, os neurônios desse modelo são amplamente conectados, de modo que todos os neurônios de uma camada estão conectados com todos os demais neurônios das camadas adjacentes. Quando uma RNA contém diversas camadas intermediárias, ela pode ser considerada uma Rede Neural Profunda, em que cada camada de neurônios tipicamente realizam diferentes transformações nos dados (GRAVES, 2012).

2.3.3 *Long Short-Term Memory*

Uma rede *Long Short-Term Memory* (LSTM) é um modelo de RNA recorrente que tem a característica de classificação levando em consideração dados de séries temporais. Uma RNA recorrente é uma topologia de aplicação capaz de criar memórias de análises de estados anteriores, de modo que o resultado de saída não depende somente da análise corrente, mas sim da combinação da análise atual com o resultado ou analises de entradas anteriores. Portanto, o modelo recorrente permite considerar o histórico de entrada através da capacidade de manter uma memória interna com o *feedback* das interações anteriores (GRAVES, 2012).

RNA recorrentes clássicas podem acompanhar as dependências arbitrárias de longo prazo referente ao histórico de entrada de dados. Porém, quando o modelo é treinado aplicando através de retropropagação, ao calcular o valor de gradientes, esse pode sofrer dos problemas conhecidos como *vanish* e *explode*. O primeiro acontece quando o valor de gradiente tende a zero, já o segundo acontece quando o valor tende ao infinito. Deste modo, o modelo LSTM é uma alternativa que resolve parcialmente os problemas de *vanish* e *explode*, uma vez que unidades LSTM permitem gradientes também fluam inalterados (ZAREMBA; SUTSKEVER; VINYALS, 2014).

Uma unidade LSTM é composta de uma célula que contém uma porta de entrada, uma porta de saída e uma porta de esquecimento. Essa célula tem a capacidade de armazenar valores de intervalos de tempo, de modo a regular o fluxo de informações que passam pelas três portas. O modelo LSTM foi desenvolvido para aprender tarefas que requerem memorias de eventos que aconteceram milhares de etapas discretas de tempos. Assim, o modelo funciona mesmo com longos atrasos entre eventos significativos e pode lidar com sinais que misturam componentes de baixa e alta frequência. Por fim, devido à capacidade flexível de tempo no armazenamento de eventos significativos, o modelo tem vantagem sobre RNA recorrentes clássicas, modelos ocultos de Markov e outros métodos de aprendizado de sequência em inúmeras aplicações (GRAVES, 2012; ZAREMBA; SUTSKEVER; VINYALS, 2014).

O presente trabalho de pesquisa aplica as abordagens de representação de texto discutidas neste capítulo na tarefa de classificação de perguntas em um sistema de QA desenvolvido. Além disso, os diferentes modelos supervisionados apresentados são testados com as diferentes abordagens de representação de texto. Por fim, o trabalho propõe uma abordagem híbrida de representação de texto baseada nas abordagens apresentadas.

3 TRABALHOS RELACIONADOS

Este capítulo destina-se a apresentar os principais trabalhos que compõem o estado da arte das áreas de QA. Para isso, discutem-se as competições (*tracks*) realizadas na área de QA, em que diferentes grupos de pesquisa disputam em uma mesma tarefa de modo a comparar seus desempenhos. Também abordam-se trabalhos relevantes na literatura, que são importantes para o tema desta pesquisa, e que apresentam resultados consideráveis, tanto para o problema de classificação de perguntas, como para o problema geral de QA. Por fim, a última seção apresenta a abordagem proposta para a tarefa de classificação de pergunta comparando com os demais trabalhos discutidos.

3.1 Competições para QA

As competições de QA dispõem de tarefas normalmente relacionadas à capacidade de um sistema responder de forma autônoma perguntas em linguagem natural de forma correta. De modo a comparar os resultados dos participantes de forma justa, os participantes recebem os mesmos recursos (corpus) para serem processados e gerar saídas. Estas competições são analisadas neste texto, de modo a selecionar trabalhos no estado da arte que são relevantes ao tema desta pesquisa.

QA@CLEF: (PEÑAS et al., 2015) é uma campanha de avaliação de sistemas de Resposta Automática a Perguntas do CLEF (MOTHE et al., 2015), que avalia a capacidade de sistemas computacionais responderem diferentes perguntas em linguagem natural. O conjunto de dados é construído pela organização do CLEF. O QA@CLEF é proposto desde 2003 (MAGNINI et al., 2003), sendo que em 2008 foi a última edição que utilizou a coleção em Português Chave. A organização direciona seus esforços em dois problemas: o primeiro é oferecer um exercício de avaliação caracterizado por cruzamento de linguagem, buscando cobrir o máximo de línguas possíveis, principalmente para as europeias; o segundo busca manter os desafios de competição de campanhas anteriores trazendo novas tarefas em cada edição, de modo a oferecer desafios para novos participantes e aos veteranos.

SQuAD: (*Stanford Question Answering Dataset*) (RAJPURKAR et al., 2016) é um conjunto de dados para sistemas de QA que avaliam a capacidade de compreensão de texto. As competições convencionais fornecem e permitem ao competidor utilizar apenas perguntas de entrada para que seja avaliada toda a capacidade do sistema desde o Processamento da Pergunta e Recuperação de Informações, até o Processamento da Resposta. Já a competição SQuAD, oferece a pergunta e uma passagem de texto, de modo que o sistema deve extrair as informações dessa sentença e sintetizar a resposta de saída. Deste modo, a competição tem foco na avaliação do módulo de Processamento da Resposta. A coleção é formada por 100.000 perguntas fornecidas por diversos voluntários (*crowdworkers*) baseadas em um conjunto de artigos da Wikipédia, onde a resposta a cada pergunta é um segmento de texto da passagem de leitura correspondente.

TREC-QA: (DANG; KELLY; LIN, 2007) é uma campanha de avaliação completa de sistemas de QA de amplo domínio. Sua primeira edição foi em 1999 (VOORHEES et al., 1999), focada em perguntas do tipo factoide. Na edição de 2003 (VOORHEES; DANG, 2003), a campanha disponibilizou perguntas do tipo definição, aumentando a dificuldade de seu conjunto. Além de documentos de noticiários e artigos, na edição de 2007, foram disponibilizados documentos de blogs, de modo que os sistemas participantes deveriam manusear textos de caráter informal e menos estruturados. Ao

decorrer das edições, a campanha de avaliação vem trazendo novas tarefas e desafios para a área de QA. Por fim, o TREC-QA disponibiliza para os participantes uma coleção de dados que contendo um conjunto de perguntas e documentos que abrange implicitamente as respostas para elas. Cabendo, assim, ao sistema extrair informações e formular a resposta a partir desses documentos para cada pergunta de entrada.

NTCIR-QA: (YAMAMOTO et al., 2016) é uma versão para línguas asiáticas equivalente das campanhas QA@CLEF e TREC-QA. O primeiro NTCIR-QA (KANDO et al., 1999) foi em 1999. Também, igualmente como as edições equivalentes ocidentais, o NTCIR-QA busca trazer em todas as edições novas tarefas e desafios, de modo a incentivar com que seus participantes se aprimorem e contribuam para o estado da arte de sistema de QA para línguas asiáticas. Ao decorrer dos anos a conferência vem apresentando constantemente desafios bilíngues entre a língua inglesa e línguas asiáticas, onde sistemas de QA participantes devem trabalhar com ambas as línguas.

3.2 Análise de trabalhos selecionados

Esta seção apresenta os principais trabalhos que compõem o estado da arte das áreas de classificação de pergunta e sistemas de QA. Os trabalhos de sistemas selecionados foram retirados das campanhas de avaliação de sistemas de QA, como também dos principais meios de publicação de trabalhos nas áreas de PLN, Recuperação de Informações e Aprendizado de Máquina. Selecionam-se os trabalhos de sistemas de QA para amplo domínio. Assim, os sistemas destes trabalhos, na sua maioria, apresentam etapas similares ao sistema desenvolvido neste trabalho de pesquisa.

Os trabalhos de (LONI, 2011; SANGODIAH; MUNIANDY; HENG, 2015) foram os mais recentes encontrados na literatura de que realizaram revisões do estado da arte sobre a área de classificação de perguntas. Ambos concluem que a classificação de perguntas é uma etapa crucial para um sistema de QA. Também perceberam um grande aumento nos trabalhos que utilizam abordagens baseadas em aprendizado de máquina ao invés de regras manuais, uma vez que essa última demanda uma grande quantidade de esforços na criação de regras. Em (LONI, 2011), aprofundam-se as diferentes características que podem ser extraídas de uma pergunta em texto, em que um quadro de trabalhos da literatura mostra que a grande maioria utiliza características de unigrama, como *Bag-of-words* e TF-IDF. Porém, percebe-se que grande parte desses trabalhos não utilizam características semânticas, sendo que o restante, na sua maioria, utilizam apenas sinônimos. Por fim, o trabalho conclui que a adição de características complexas podem comprometer a performance do modelo, já que essas podem adicionar ruídos e causar erros de classificação. Já no trabalho de (SANGODIAH; MUNIANDY; HENG, 2015), percebe-se uma maior mobilização para a utilização de abordagens para representação de texto de forma semântica, como *word embedding*, ao invés de utilizar apenas características lexicais, como *Bag-of-words* e TF-IDF. Também, percebe-se que na maioria dos trabalhos avaliados, os que utilizam o modelo supervisionado SVM apresentam os melhores desempenhos.

De modo a contribuir para o desempenho de sistema de QA para língua Árabe, o trabalho de (LAHBARI; OUATIK; ZIDANI, 2017) propõe a utilização de *Bag-of-words* e modelos de aprendizado supervisionado para a tarefa de classificação de perguntas. Emprega-se uma fusão das coleções de multilíngues UIUC e CLEF para a realização dos testes. Também, somente o tipo de pergunta da taxonomia Li & Roth é utilizado, em que se descarta os subtipos da taxonomia devido as particula-

ridades da língua Árabe e da coleção de dados. A taxonomia Li & Roth é um trabalho (LI; ROTH, 2002) que define uma hierarquia de classes e subclasses normalmente utilizadas no problema de classificação de perguntas. Foram realizados experimentos utilizando os modelos SVM, Naive Bayes e Árvores de decisão, em que os resultados mostram que, com 84 % das instâncias classificadas corretamente, o modelo SVM superou os demais.

O trabalho de (ZHOU et al., 2015) propõe o modelo supervisionado chamado de C-LSTM que combina CNN com LSTM para tarefas de classificação de texto. O modelo é capaz de aprender características ao nível de frases através de uma camada de convolução e o histórico dessa representação em uma sentença alimenta uma camada de LSTM. Os testes são realizados para a tarefa de classificação de sentimentos e classificação de perguntas. Utiliza-se a base de dados UIUC, composta por uma taxonomia de 6 classes. Os experimentos utilizam 1.000 instâncias para a busca de hiperparâmetros, 5.452 para treinamento do modelo e 500 para teste. Os resultados apresentados mostram que em ambas as tarefas, a abordagem obteve resultados promissores. No caso da tarefa de classificação de perguntas, a abordagem obteve acurácia de 94.6 %, enquanto que a melhor abordagem obteve acurácia de 95 %. Essa melhor abordagem, proposta por (SILVA et al., 2011), utiliza um modelo SVM e extrai do texto das perguntas o unigrama, bigrama, *wh-word*, POS tag, árvore de parser e *WordNet synsets*, por fim, testa-se a sentença em 60 regras codificadas manualmente.

O trabalho de (MOHD; HASHMY, 2018) propõe uma base de conhecimento semântica construída através do *WordNet* para ser aplicada como uma métrica de similaridade semântica. Para comparação de resultados, os experimentos são realizados com quatro diferentes modelos de aprendizado de máquina: *Nearest Neighbors*, *Naive Bayes*, Árvores de decisão e SVM. Para o modelo SVM realizam-se experimentos com *kernel* linear e o *kernel* proposto SVMSR, que aplica os conceitos de similaridade semântica proposta. O objetivo do trabalho é superar as desvantagens da abordagem *Bag-of-words* adicionando a métrica de relação semântica baseada em *WordNet* chamada de SR para um conjunto de palavras no *kernel* semântico proposto. O SR considera dois componentes em sua métrica, o tamanho e a profundidade do caminho no *WordNet*. Os experimentos utilizam a coleção UIUC e a taxionomia de Li & Roth, composta por 6 classes principais e 50 subclasses. Os resultados mostram que o modelo SVM utilizando o *kernel* SR obteve os melhores resultados, atingindo 91.9 % de acurácia na classifição de perguntas e 86.41 % na classificação de subclasses.

O trabalho de (AOUICHAT; AMEUR; GEUSSOUM, 2018) divide o problema de classificação de perguntas em duas etapas: a primeira utiliza um modelo SVM para identificar as principais classes da perguntas, enquanto que a segunda utiliza para cada classe principal um modelo CNN para classificar as subclasses. A primeira etapa utiliza a abordagem TF-IDF para representar o texto das perguntas, enquanto que a segunda, utiliza a abordagem *Word2vec*. Os experimentos utilizam a coleção UIUC e foram divididos em duas etapas, a primeira testou os modelos de aprendizado de máquina SVM, Máxima Entropia e Florestas Aleatórias para classificar as principais classes de perguntas. A segunda testou o modelo CNN com *Word2vec* de 100 unidades de dimensões para classificação de subclasses de perguntas. Os resultados mostram que o modelo SVM com abordagem TF-IDF obteve os melhores resultados na primeira etapa, com 93 % de *F1-Score*, e o modelo CNN, com a abordagem *Word2vec*, obteve 92,49 % de *F1-Score* na segunda etapa.

O trabalho de (WANG et al., 2017) é desenvolvido pelo grupo de pesquisa computacional de linguagem natural da Microsoft. Este trabalho, chamado de R-NET, é um dos mais bem colocados

no ranking de sistemas de QA da competição SQuAD. Seu objetivo é o entendimento de linguagem natural, uma etapa importante no módulo de Processamento da Resposta de um sistema de QA, que consiste na extração de informações e síntese da resposta final. A abordagem proposta é um modelo de redes neurais *end-to-end* para leitura de perguntas de estilo de compreensão. Divide-se essa abordagem em duas etapas: na primeira treina-se um modelo para extrair evidências a partir de passagens de texto relacionados à pergunta de entrada; a segunda etapa consiste na síntese da resposta a partir as informações extraídas. Utilizando a coleção SQuAD, os resultados dos experimentos mostram que o R-NET obteve *F1-Score* de 86,5 e *ExactMatch* de 79,9, enquanto que o desempenho humano é *F1-Score* de 91,2 e *ExactMatch* 82,3.

O sistema de QA (WANG; YAN; WU, 2018) propõe o uso de redes neurais profundas para a compreensão de leitura aplicada a sistemas de QA de amplo domínio. A abordagem proposta primeiramente codifica a pergunta e a passagem através de uma representação *word embedding*, de modo a capturar uma melhor representação em nível semântico. Em seguida, o modelo propõe uma hierarquia de atenção ao texto, de forma a capturar a relação entre a pergunta e a passagem em diferentes níveis de granularidade. Assim, a resposta é sintetizada progressivamente através um alinhamento multinível a hierarquia de atenção ao texto. Utilizando a coleção SQuAD, os resultados dos experimentos atingiram efeitos robustos, obtendo *F1-Score* de 87,0 e *ExactMatch* de 80,4, enquanto que o desempenho humano é *F1-Score* de 91,2 e *ExactMatch* de 82,3.

De modo a prover acesso à informação escrita através da inclusão digital, os trabalhos de (WILKENS et al., 2010; WILKENS; VILLAVICENCIO; VICARI, 2016) propõem o COMUNICA, um sistema de QA de voz para o português brasileiro. Esse sistema tem a capacidade de consultar base de dados estruturadas e não estruturadas. Para determinar a classe da pergunta, aplicam-se ferramentas rasas e profundas que acessam ontologias de amplo e específico domínio através de informações lexicais e semânticas. O módulo de Recuperação de Informações utiliza tando uma base de dados estruturada quanto uma não estruturada para realizar consultas. Por fim, com as informações recuperadas, é produzida a resposta em linguagem natural através de padrões de respostas pré-definidas. Além dos módulos convencionais, esse sistema contém dois módulos responsáveis pelo reconhecimento da fala de entrada e pela síntese da fala de saída. Os experimentos utilizaram a base de dados FAMURS, composta por perguntas em fala realizada por pessoas. O trabalho resultou em contribuições reduzindo a diferença entre agentes de conversação comerciais e acadêmicos investigando o impacto de sistemas de parser nas etapas do sistema.

Em (YAO, 2015), apresenta-se um sistema de QA composto somente de duas etapas principais. Diferente das abordagens convencionais de sistema de QA, o trabalho não realiza classificação de perguntas ou recuperação de informações, ao invés disso, propõe-se dois módulos, um responsável por identificar o tópico da pergunta de entrada e um outro para predizer a resposta correta. Para identificação do tópico da pergunta, a etapa utiliza dados eficientemente estruturados para reconhecer entidades nomeadas em seu texto. Já a etapa de predição, tem a tarefa de identificar a relação do tópico com a resposta de saída. Deste modo, utiliza-se uma estrutura de tupla composta por tópico de pergunta, relação e resposta para consultar a base de conhecimento *Freebase* (BERANT et al., 2013) e assim retornar a resposta de saída. Para experimentos, utiliza-se a coleção WEBQUESTIONS, um conjunto de dados com 6.626 pares de perguntas e resposta que devem ser respondidas através da *Freebase*. O sistema atingiu 53,5 de *F1-Score*, 7,8 pontos percentuais do que o segundo melhor

participante. Os autores atribuem a boa performance a dois possíveis motivos: I) a simplicidade dos métodos minimizaram a propagação de erros durante o fluxo de processamento; II) foi utilizada supervisão direta, enquanto que a maioria dos trabalhos anteriores usaram supervisão distante.

O trabalho de (ABBAS et al., 2016) propõe o WikiQA, um sistema de QA capaz de responder perguntas utilizando dados da Wikipédia através da base de conhecimento DBPedia (AUER et al., 2007). O sistema é composto pelos módulos de Processamento da Pergunta, Recuperação de Informações e Processamento da Resposta. No módulo de Processamento da Pergunta, os autores optaram por utilizar o *framework* de QA OpenEphyra (SCHLAEFER et al.,), que disponibiliza um módulo para classificação de perguntas que utiliza marcação de tipo e regras manuais para a classificação de perguntas. Na Recuperação de Informações, o sistema utiliza uma abordagem híbrida entre dados estruturados e texto brutos. Assim, primeiramente o sistema tenta encontrar a resposta diretamente na DBPedia, caso não encontre, a resposta é buscada em texto bruto da Wikipédia. Para o Processamento da Resposta, o sistema realiza a extração de respostas de sentenças aplicando REM e comparando os termos e seus sinônimos contidos na pergunta com os da sentença. Por fim, valida-se a resposta final verificando se essa é uma entidade do mesmo tipo da classe da pergunta. Para os experimentos, os autores utilizaram uma coleção com 200 perguntas e respostas criadas por 50 voluntários. Os resultados mostram que o sistema obteve *F1-Score* de 64 %, resultado superior aos demais sistemas de QA aplicando essa mesma coleção de perguntas. Porém, uma das principais limitações levantadas pelos autores foi o fato que o sistema só era capaz de responder perguntas que continham informações na Wikipédia.

Em (AMARAL et al., 2008) relata-se a participação do sistema de QA Priberam na competição QA@CLEF de 2008. Esse sistema vem obtendo os melhores resultados desde a edição de 2004 na tarefa de QA para a língua portuguesa utilizando a coleção Chave. O Priberam é um sistema de QA para amplo domínio, baseado em uma arquitetura de cinco etapas que utilizam recursos linguísticos e ferramentas de PLN. A primeira etapa é responsável pelo processo de indexação de documentos para consulta. A segunda realiza análise da pergunta, de modo a classificá-la através de regras manuais. A terceira faz a busca e recuperação de documentos utilizando palavras-chave, sinônimos, domínio ontológico e a classe da pergunta. A quarta etapa é responsável pela recuperação de sentenças. Por fim, a quinta etapa realiza a extração de respostas utilizando REM. Na edição de 2008 do QA@CLEF para a língua portuguesa (PETERS et al., 2009), o sistema Priberam obteve 63,5 % de acurácia respondendo perguntas da coleção Chave, apresentando o melhor desempenho entre os participantes.

A Tabela 3.1 apresenta a relação de trabalhos relacionados ao tema deste trabalho de pesquisa. Identifica-se que abordagens baseadas em regras criadas manualmente não estão sendo mais usadas, perdendo espaço para abordagens baseadas em aprendizado de máquina. Isso deve-se à grande demanda de quantidade de esforços na criação de regras. Já abordagens baseadas em aprendizado de máquina, normalmente utilizam modelos supervisionados em que necessitam-se apenas dados anotados para que o modelo aprenda a classificar. Percebe-se essa tendência principalmente na etapa de classificação de perguntas e para a compreensão de leitura no módulo de Processamento da Resposta. Também, pode-se observar que dentre os sistemas analisados que conseguem responder perguntas,

Tabela 3.1: Trabalhos Relacionados.
Fonte: O Autor

Trabalho	**Processamento de Pergunta**	**Recuperação de Informações**	**Processamento de Resposta**	**Coleção de Testes**	**Desempenho**
(ZHOU et al., 2015)	C-LSTM + *Word Embedding*	-	-	UIUC	94,6 % de acurácia na classificação de perguntas
(LAHBARI; OUATIK; ZIDANI, 2017)	*Bag-of-words* + (SVM, *Naive Bayes* ou Árvores de decisão)	-	-	UIUC + CLEF	84 % de acurácia com SVM na classificação de perguntas
(MOHD; HASHMY, 2018)	*WordNet* + *Bag-of-words* + (SVM, *Naive Bayes*, *Nearest Neighbors* ou Árvores de decisão)	-	-	UIUC	91,9 % de acurácia com SVM na classificação de perguntas
(AOUICHAT; AMEUR; GEUS-SOUM, 2018)	TF-IDF + SVM e *Word2Vec* + CNN	-	-	UIUC	93 % de *F1-Score* na classificação de perguntas
(WANG et al., 2017)	-	-	RNA Profunda + extração de evidências + síntese de respostas	SQuAD	86,5 % de *F1-Score* na resposta final
(WANG; YAN; WU, 2018)	-	-	Word Embedding + RNA Profunda	SQuAD	87 % de F1-Score na resposta final
(WILKENS; VILLAVI-CENCIO; VICARI, 2016)	Características Léxica + Sintática e Ontologias	Consulta em Base de dados Estruturadas e Não Estruturadas	Padrões pré-definidos de Respostas	FAMURS (Voz)	Reduziu diferença entre sistemas comerciais e acadêmicos
(YAO, 2015)	Classificação de Tópicos + REM	Consulta na Base de conhecimento Freebase	Busca através de Tupla na base de conhecimento Freebase	WebQuestions	53,5 % de *F1-Score* na resposta final
(ABBAS et al., 2016)	Regras Manuais com *OpenEphyra*	Consulta na Base de conhecimento DBPedia e textos da Wikipédia	Ranqueamento através de termos e sinônimos do texto da pergunta	200 perguntas criadas através de 50 voluntários humanos	64 % de F1-Score na resposta final
(AMARAL et al., 2008)	Regras Manuais + Léxico + Sintático	Ontologia Geral + Consulta em Texto	Extração de Resposta com REM	CLEF (Chave)	63,5 % de acurácia na resposta final

normalmente esses seguem a arquitetura descrita no trabalho de (JURAFSKY; MARTIN, 2014), apresentando os módulos de Processamento da Pergunta, Recuperação de Informações e Processamento da Resposta.

Os trabalhos discutidos e apresentados na Tabela 3.1 são baseados em sistema de QA de amplo domínio, mesmo assim, observa-se que alguns utilizam base de dados estruturadas, como ontologias. Espera-se essa característica em sistemas de domínio específico, porém, no caso dos trabalhos abordados, as bases de dados estruturados são de domínios gerais. Entende-se que essa característica insere conhecimento semântico de mundo externo ao sistema, ajudando-o a ganhar precisão na busca de informações. Por exemplo, com uma determinada ontologia, o sistema de QA pode identificar se a entidade "Porto Alegre"é um bairro, uma cidade, um estado ou um país.

3.3 Abordagem Híbrida Proposta

O objetivo da etapa de classificação de pergunta de um sistema de QA é determinar a classe semântica de um determinada pergunta, levando em consideração o tipo de resposta esperada. Por exemplo, a pergunta "Onde nasceu João?"espera como resposta uma entidade referente a localização enquanto que a pergunta "Quem mora na casa de João?"refere-se a uma entidade do tipo pessoa. Encontram-se principalmente duas abordagens para classificação de perguntas na literatura. A primeira, consiste na criação manual de regras utilizando características lexicais e sintáticas. A segunda aplica modelos supervisionados de aprendizado de máquina, de modo a treinar esse modelo com perguntas com classes já anotadas. Este trabalho de pesquisa propõe aplicar modelos supervisionados para a tarefa, unindo características lexicais e semânticas para a classificação de perguntas. Deste modo, esta seção destina-se a apresentar a abordagem híbrida proposta para tarefa.

A tarefa de classificação de pergunta pode ser considerada como um tipo de classificação de texto. Normalmente, uma instância em uma tarefa de classificação de texto é composta por uma quantidade maior de palavras do que uma pergunta. Um texto pode ser um comentário, uma resenha ou um artigo científico. Esses textos normalmente são formados por diversas palavras, ou até mesmo, diversos parágrafos, diferente de uma pergunta, que contém em média, aproximadamente, cinco palavras. Assim, uma diferença significativa entre às duas tarefas é a diferença na quantidade de informação disponível.

Uma maneira de representar informações de texto é através da abordagem *Bag-of-words* em que se representa o texto utilizando um multiconjunto de suas palavras, desconsiderando a ordem delas e a estrutura gramatical, mas mantendo sua multiplicidade. Ao invés de manter a multiplicidade das palavras, pode-se usar TF-IDF, em que quanto mais comum uma palavra em um conjunto de documentos, menor será seu peso no vetor de representação. Estas abordagens são conhecidas por representar lexicalmente o texto. Outra maneira de representação de informações de texto é através de *word embedding*, em que palavras ou sentenças são mapeadas em um vetor de números reais que representa sua posição no espaço semântico. Deste modo, está abordagem é conhecida por representar semanticamente o texto.

Comumente, na classificação de texto, abordagens de representação lexicais obtém boa performance aplicada em modelos supervisionados de aprendizado de máquina. Também, modelos de aprendizado profundo estão atingindo resultados promissores na tarefa utilizando abordagens de re-

presentação semântica com *word embedding*. Porém, como na tarefa de classificação de texto contém um número limitado de palavras, a utilização de uma só abordagem pode ser insuficiente pra classificar a pergunta corretamente. Além disso, para línguas não inglesa, comumente encontra-se dificuldade na aquisição de ferramentas com boa performance para extração de características sintáticas. Deste modo, este trabalho de pesquisa propõe uma abordagem híbrida, combinando características lexicais e semânticas da pergunta. Para isso, utiliza-se TF-IDF para representação lexical e *Word2vec* para representação semântica.

$$\langle tfidf_q, w2v_{q,v} \rangle \tag{3.1}$$

$$w2v_{q,v} = \frac{1}{n} \sum_{i=1}^{n} v_i \tag{3.2}$$

A representação da abordagem híbrida proposta pode ser representada formalmente através da Equação 3.1. Uma pergunta q é representada por um conjunto de palavras $w_1, \ldots, w_n$, em que cada palavra w_i contém um vetor v_i que representa sua posição no espaço semântico. Também, $tfidf_q$ é um vetor com s dimensões, em que s tem o tamanho do vocabulário de palavras, e cada posição desse vetor recebe o peso TF-IDF da palavra w_i da pergunta q. Por fim, $w2v_{q,v}$, representado na Equação 3.2, é um vetor com 300 dimensões, de modo que cada dimensão recebe a média aritmética das dimensões das palavras da pergunta q.

4 METODOLOGIA

Este capítulo destina-se a apresentar a metodologia de pesquisa aplicada neste trabalho. Esta pesquisa divide-se em duas etapas, a primeira objetiva desenvolver e testar a abordagem híbrida proposta para classificação de perguntas, enquanto que a segunda destina-se na aplicação dessa abordagem em um sistema de QA completo desenvolvido para a língua portuguesa. Deste modo, o capítulo aborda a metodologia utilizada para a validação da hipótese de pesquisa através do desenvolvimento e teste da abordagem híbrida e o desenvolvimento de um sistema de QA completo para sua aplicação e testes. Além da metodologia para avaliação da abordagem proposta, o capítulo apresenta a coleção Chave, principal *dataset* utilizado para avaliação e teste, as demais coleções para treinamento e testes, por fim, os métodos e métricas de avaliação de cada etapa do sistema de QA. A Figura 4.1 apresenta uma visão geral do processo de metodologia utilizado. As etapas apresentadas são aprofundadas no decorrer desse capítulo.

Na etapa de classificação de perguntas, determina-se qual é a classe da pergunta de entrada do sistema de QA, ou seja, qual é o tipo de resposta ao qual a pergunta está se referindo. Pode-se utilizar essa informação em diferentes etapas do sistema, mas principalmente, emprega-se para extrair de sentenças de texto a resposta final através de entidades mencionadas da mesma classe da pergunta. Assim, caso a classe da pergunta for classificada erroneamente, possivelmente o sistema não extrairá a entidade correta da sentença e retornará uma resposta final incorreta. Por esse motivo, esta pesquisa assume que a etapa de classificação da pergunta é uma etapa crucial para o desempenho de um sistema de QA. Deste modo, espera-se que a capacidade do sistema de responder perguntas de forma correta esteja diretamente relacionada com a capacidade de classificar as perguntas de entrada. Assim, realizam-se testes de desempenho da etapa de classificação da pergunta e também testes com o sistema de QA completo. Espera-se que os resultados dos teste de classificação de pergunta comparando as diferentes abordagens tenham desfechos similares aos resultados com o sistema de QA aplicando estas mesmas abordagens no sistema completo.

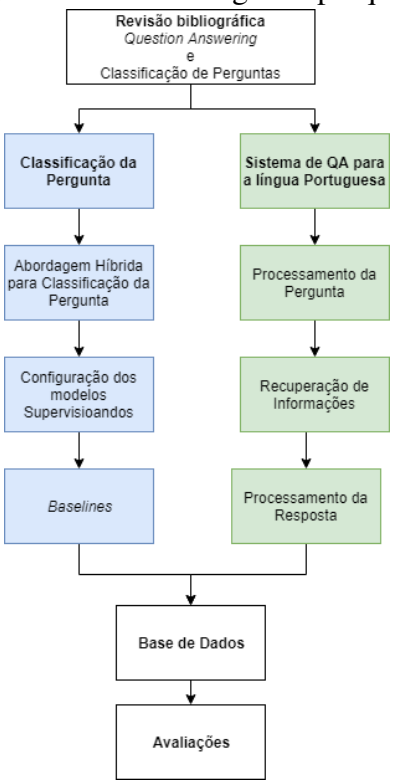
4.1 Testes de Classificação da Pergunta

Para a validação da proposta de classificação da pergunta através da abordagem híbrida, testa-se essa juntamente com outras abordagens de representação de texto utilizadas como *baseline*, em diferentes modelos supervisionados de aprendizado de máquina. Isso permite comparar o desempenho da abordagem proposta com outras abordagens já conhecidas. Tanto as abordagens de *baseline*, quanto os modelos supervisionados foram selecionados de acordo com seu desempenho em problemas relacionados à classificação de texto, observados em trabalhos na literatura.

A coleção Chave, disponibilizada através do site Linguateca[1], foi utilizada como principal *dataset* de teste, uma vez que a coleção está em português, a mesma também disponibiliza tanto os dados necessários para a classificação da pergunta, como também o restante dos dados necessários para realizar testes com um sistema completo de QA. Também, outros *datasets* são utilizados para

[1] https://www.linguateca.pt/CHAVE/

Figura 4.1: Metodologia de pesquisa.



Fonte: O Autor

testes de classificação da pergunta, porém, os mesmos não podem ser utilizados nos testes do sistema de QA, uma vez que não possuem dados para a tarefa.

Para realização de testes, utilizam-se os atributos 'categoria' e 'tipo' disponíveis em todas as perguntas na coleção Chave. A classe de cada instância do *dataset* é definida através de uma mistura desses dois atributos, conforme a subseção 4.3.1. De modo a obter resultados mais confiáveis, avalia-se cada modelo utilizando a técnica de validação cruzada (*cross-validation*), que permite avaliar a capacidade de generalização de um modelo. Além disso, como a hipótese de pesquisa envolve o teste com diferentes tamanhos de conjunto de treinamentos, os modelos e abordagens são testados variando o tamanho do conjunto de treinamento.

A implementação do modelo híbrido é realizada utilizando a linguagem de programação *Python* juntamente com as bibliotecas *Scikit-learn*[2], *Gensim*[3] e *Numpy*[4]. Estas bibliotecas permitem abstrair a complexidade de implementação de modelos de aprendizado de máquina, representação vetorial de texto e manipulação de estruturas de dados. O *Scikit-learn* é uma biblioteca que, além de oferecer modelos para aprendizado de máquina, inclui diversos algoritmos para representação de texto. O *Gensim* é uma biblioteca para modelagem de espaço vetorial que oferece recursos para trabalhar com modelos de *word embedding*. Por fim, o *Numpy* auxilia na manipulação de vetores.

[2]https://scikit-learn.org/
[3]https://radimrehurek.com/gensim/
[4]http://www.numpy.org/

Figura 4.2: Código da abordagem híbrida proposta para classificação da pergunta.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import KeyedVectors
import numpy as np


word2vec = KeyedVectors.load_word2vec_format('cbow_s300.txt')


def tfidf_vectorizer(MIN_GRAM=1, MAX_GRAM=1, LOWER=True):
    return TfidfVectorizer(analyzer=lambda x: x, strip_accents=None,
                           ngram_range=(MIN_GRAM, MAX_GRAM),
                           token_pattern=u'(?u)\\b\\w+\\b',
                           lowercase=LOWER)


class HybridVectorizer(object):

    def __init__(self, word2vec):
        self.word2vec = word2vec
        self.dim = len(word2vec)
        self.tfidf = tfidf_vectorizer()

    def fit(self, X, y):
        self.tfidf.fit(X)
        return self

    def transform(self, X):
        ret = []
        for sentence in X:
            w2v = np.mean([self.word2vec[w]
                           for w in sentence if w in self.word2vec]
                    or [np.zeros(self.dim)], axis=0)
            tfidf = self.tfidf.transform([sentence]).toarray()
            ret.append(np.concatenate([w2v, tfidf[0]]))
        ret = np.array(ret)
        return ret
```

Fonte: O Autor

4.1.1 Abordagem Híbrida para Classificação da Pergunta

Os testes da abordagem proposta utilizam modelos supervisionados de aprendizado de máquina. Assim, é necessário dividir a coleção em dados para o treinamento do modelo e dados de teste. Esses dados, primeiramente estarão no formato de texto e precisam ser transformados em um vetor numérico para então serem utilizados pelos modelos supervisionados. Deste modo, a transformação do texto em um vetor numérico consiste na aplicação do modelo híbrido proposto. A Figura 4.2 apresenta o código da abordagem proposta utilizada para representar o texto em um vetor de características para os modelos supervisionados.

Conforme o código da Figura 4.2, primeiramente, após importar as bibliotecas, carrega-se uma matriz *word2vec* externa utilizando a biblioteca *Gensim*. O modelo *word2vec* utilizado é o "CBOW

300 dimensões"disponibilizado pelo NILC[5] (HARTMANN et al., 2017), que consiste em um modelo de *continuous bag of words* já treinado utilizando um grande corpus de fontes e gêneros variados em português do Brasil e português europeu. O modelo *Word2vec* foi selecionado devido ao número de dimensões, uma vez que um número reduzido pode comprometer a performance na representação semântica e um número grande pode exigir maior processamento computacional. Desta forma, normalmente os modelos disponibilizados na literatura contêm aproximadamente 300 dimensões. Em seguida, a função *tfidf_vecotrizer()* é responsável por criar um modelo de representação de texto através de TF-IDF utilizando a biblioteca *Scikit-learn*.

A classe *HybridVectorizer* é responsável por transformar o texto em um vetor de características numéricas, representando a ocorrência de palavras e o espaço semântico médio dos termos inseridos nela. Para realizar essa transformação, é necessário treinar o modelo TF-IDF. Para isso utiliza-se a função *fit()* para passar um corpus de texto. Após treinado o modelo de TF-IDF, é possível utilizar a função *transform()*, que utiliza o modelo TF-IDF treinado anteriormente e um modelo *Word2vec* externo já treinado para transformar o texto em um vetor numérico. Além disso, é possível delimitar o tamanho máximo de vocabulário do modelo TF-IDF, desta forma, somente utilizam-se os termos mais frequentes do vocabulário.

4.1.2 Configuração dos Modelos Supervisionados

Os modelos supervisionados de aprendizado de máquina foram selecionados segundo seu desempenho em tarefas de classificação de texto. Além disso, os modelos devem ser capazes de manipular as estruturas de dados utilizados na abordagem híbrida proposta, como também nas abordagens de *baseline*. Deste modo, foram selecionados os modelos SVM, MLP e LSTM. Os modelos *Naive Bayes Bernoulli* e *Naive Bayes Multinomial* não foram utilizados, pois o primeiro não manipula valores negativos, sendo esses encontrados em abordagens que utilizam *Word2vec*, e o segundo não manipula valores flutuantes, que são encontrados em abordagens que utilizam *Word2vec* e TF-IDF.

O modelo LSTM utilizado foi baseado no trabalho de (KAWAKAMI, 2008) e é composto por 4 camadas, sendo que a primeira é a camada com LSTM, a segunda é uma camada densa, a terceira uma camada de *dropout* e a última uma camada densa de saída. Utiliza-se o otimizador *RMSprop* e a função de custo *categorical_crossentropy*. No caso do modelo LSTM, alguns aspectos da representação do texto em um vetor numérico diferem-se dos demais modelos, uma vez que o modelo utiliza uma dimensão extra para representar a sequência de palavras da sentença. Desta forma, não é necessário realizar a média dos vetores *Word2vec*, já que os vetores de cada palavra são utilizados individualmente. Para a aplicação do modelo foram utilizadas as bibliotecas *Keras*[6] e *TensorFlow*[7], que permitiram abstrair a complexidade de implementação de um modelo de redes neurais profundas.

Para os testes com o modelo SVM, primeiramente realizam-se testes individuais utilizando diferentes *kernels*. Também, testa-se cada modelo variando a parametrização, buscando uma melhor configuração de parâmetros. Assim, através de observações empíricas, seleciona-se o melhor *kernel* e

[5] http://nilc.icmc.usp.br/embeddings
[6] https://keras.io/
[7] https://www.tensorflow.org/

a melhor configuração de parâmetros observados para serem utilizados como modelo SVM nos testes com a classificação da pergunta. Para aplicação do modelo foi utilizada a biblioteca *Scikit-learn*.

O modelo MLP utilizado é composto por três camadas, sendo que a primeira é a camada de entrada, a segunda é a camada oculta, composta de 400 neurônios e com *dropout* de 0.2, e a última é a camada de saída, composta de 6 neurônios. Utiliza-se o otimizador *RMSprop* e a função de custo *categorical_crossentropy*. A taxa de aprendizado utilizada é de 0.001. Para a aplicação do modelo foram utilizadas as bibliotecas *Keras* e *TensorFlow*.

4.1.3 Baselines

Com o objetivo de comparar os resultados obtidos com a abordagem proposta para classificação da pergunta, foram selecionadas abordagens como *baselines* que são amplamente utilizadas em problemas de classificação de texto. Estas abordagens são de fácil aplicação, como também são *baselines* que apresentam desempenhos difíceis de serem superados, uma vez que apresentam performance relevante e diversos problemas de representação e classificação de texto. Deste modo, foram selecionadas três abordagens, as quais são:

- ***Bag-of-Words* (BoW):** Representa-se o texto através de um vetor de ocorrência de palavras, desconsiderando gramática e ordem das palavras, mas mantendo a multiplicidade das mesmas. Para aplicação dessa abordagem, utiliza-se o recurso *sklearn.feature_extraction.text.CountVectorizer* disponibilizado pela biblioteca *Scikit-learn* que auxilia no processo de implementação no problema.

- ***Bag-of-words* + TF-IDF (TFIDF):** Da mesma forma que o *Bag-of-Words*, representa-se o texto através de um vetor de ocorrência de palavras, porém, o valor de cada elemento do vetor refere-se à frequência do termo no texto inverso à sua frequência nos demais documentos. Para aplicação dessa abordagem, utiliza-se o recurso *sklearn.feature_extraction.text.TfidfVectorizer* disponibilizado pela biblioteca *Scikit-learn* que auxilia no processo de implementação no problema.

- ***Word2Vec* (W2V):** Abordagem utilizada para representar o texto através de *word embeddings*. Desta forma, representa-se o texto em um espaço semântico, onde cada palavra é um vetor de valores flutuantes. Para representar uma sentença, é realizada a média aritmética dos vetores semânticos das palavras da sentença. Utiliza-se o modelo externo "CBOW 300 dimensões"disponibilizado pelo NILC como modelo *Word2vec* já treinado (o mesmo modelo utilizado na abordagem proposta neste trabalho). Por fim, emprega-se o recurso *gensim.models.KeyedVectors* disponibilizado pela biblioteca *Gensim* para carregar e manipular o modelo *Word2vec*.

A coleção Chave é amplamente utilizada por trabalhos de pesquisa de sistemas de QA para a língua portuguesa. Porém, normalmente nestes, os problemas de pesquisas estão focados em apresentar resultados referentes ao desempenho final do sistema, e não da etapa de classificação da pergunta. Além disso, conforme as motivações expressas na seção 4.3.1, foram realizados pré-processamentos na coleção Chave que alteraram a maneira de como as classes das perguntas foram estruturadas originalmente. Por esses motivos, foi inviável comparar os resultados da abordagem desta pesquisa com resultados de trabalhos específicos da literatura que utilizaram a coleção Chave.

O código de implementação desenvolvido para realizar os testes de classificação da pergunta estão disponíveis no projeto *qc-hybrid*[8] no *Github*. O modelo "CBOW 300 dimensões"de *Word2vec* é disponibilizado pelo NILC [9]. Por fim, a coleção Chave é disponibilizada através do site do Linguateca[10].

4.2 Sistema de QA para a língua Portuguesa

De modo a mensurar a abordagem híbrida proposta para a etapa de classificação da pergunta aplicada em um sistema completo de QA, foi desenvolvido um sistema de QA para a língua portuguesa que permite testar distintas abordagens de classificação da pergunta. Como a coleção Chave contém perguntas de diferentes assuntos e dispõe documentos em texto bruto, o sistema de QA desenvolvido é de domínio amplo, que se objetiva a responder perguntas de qualquer domínio de conhecimento, e é capaz de trabalhar com dados não estruturados (texto bruto). Além disso, o sistema só é capaz de responder perguntas do tipo factoide, deste modo, as perguntas do tipo *Definition* são ignoradas para os testes com o sistema.

O sistema de QA desenvolvido baseia-se na arquitetura padrão encontrada na literatura (LONI, 2011; JURAFSKY; MARTIN, 2014). O sistema é composto de: um módulo de Processamento da Pergunta, responsável por classificá-la e gerar uma *query* para a recuperação de documentos; um módulo de Recuperação de Informações, responsável por recuperar documentos e passagens, e reconhecer e classificar entidades mencionadas no texto; por fim, um módulo para Processamento da Resposta, responsável por criar um ranque de perguntas candidatas e retornar a resposta final. Uma vez que esta pesquisa de mestrado objetiva-se na etapa de classificação da pergunta, as demais etapas do sistema de QA são desenvolvidas utilizando abordagens de simples implementação, que são tratadas nas próximas subseções. Para o desenvolvimento do sistema de QA utilizou-se da linguagem de programação *Python*.

4.2.1 Processamento da Pergunta

O módulo de Processamento da Pergunta é responsável por classificar a pergunta de entrada e gerar uma *query* para a etapa de Recuperação de Informações. Para o desenvolvimento do sistema de QA deste trabalho, na etapa de classificação da pergunta utiliza-se um modelo supervisionado já treinado com alguma abordagem que deseja-se testar utilizando o sistema de QA completo. Desta forma, para a classificação da pergunta, apenas é necessário aplicar o modelo treinado. Já para a geração da *query*, é necessário extrair os termos chaves da pergunta de entrada e criar *a query* para consultar a base de dados do sistema.

A aplicação do modelo supervisionado já treinado para classificação da pergunta é feita através de uma função responsável por determinar cada classe de pergunta para uma lista de perguntas de entrada. A Figura 4.3 apresenta o código da função utilizada para predizer a classe da pergunta dada

[8] https://github.com/eduardogc8/qc-hybrid
[9] http://nilc.icmc.usp.br/embeddings
[10] https://www.linguateca.pt/CHAVE/

Figura 4.3: Código da função utilizada para classificar uma lista de perguntas.

```
def predict_questions_class(model, questions):
    ret = []
    for question in questions:
        text = question['question']
        text = text.lower()
        text = nltk.word_tokenize(text)
        question['predict_class'] = model.predict([text])[0]
        ret.append(question)
    return ret
```

Fonte: O Autor

uma lista de perguntas a serem classificadas. A função recebe como entrada, através do parâmetro '*model*', um modelo supervisionado já treinado capaz de predizer a classe de uma pergunta e uma lista de perguntas, através do parâmetro '*questions*'. Em seguida, para cada pergunta da lista '*questions*', é atribuída a classe de pergunta através da função *predict()*. Para isso, primeiramente todos os caracteres do texto são passados para letras minúsculas e em seguida transforma-se o texto em uma lista de *tokens* utilizando a função '*word_tokenize()*' disponível através da biblioteca NLTK[11]. Por fim, a função retorna a lista de perguntas com suas classes preditas.

Para a criação das *queries* de cada pergunta, utiliza-se uma função responsável por criar uma *query* a partir de palavras que não são palavras vazias (*stop words*) do texto da pergunta. A Figura 4.4 apresenta o código da função utilizada para criar as *queries* de uma lista de perguntas de entrada. A função recebe como entrada uma lista de perguntas através do parâmetro '*questions*' e uma lista de palavras vazias através do parâmetro '*stop_words*'. Em seguida, para cada pergunta da lista '*questions*', é criada a sua *query* através das palavras da pergunta que não estão contidas na lista '*stop_words*'. Para isso, primeiramente todos os caracteres do texto são passados para letras minúsculas e em seguida transforma-se o texto em uma lista de *tokens* utilizando a função '*word_tokenize()*' disponível através da biblioteca NLTK. Por fim, a função retorna a lista de perguntas com suas *queries* criadas.

[11] https://www.nltk.org/

Figura 4.4: Código da função utilizada para criação de *queries*.

```
def make_query(questions, stop_words):
    ret = []
    for question in questions:
        text = question['question']
        text = text.lower()
        text = nltk.word_tokenize(text)
        key_words = []
        for token in text:
            if token not in stop_words:
                key_words.append(token)
        question['query'] = ' '.join(key_words)
        ret.append(question)
    return ret
```

Fonte: O Autor

4.2.2 Recuperação de Informações

O módulo de Recuperação de Informações é responsável pela recuperação de documentos relevantes à pergunta, pela seleção de sentenças desses documentos, e por fim, realiza a identificação e a classificação de entidades mencionadas nestas sentenças. Na etapa de recuperação de documentos, utiliza-se a ferramenta *Solr* desenvolvida para indexar e buscar os documentos da base de dados do sistema. A etapa de recuperação de sentenças é realizada através das palavras-chave da *query* e da etapa de REM. Essa, por sua vez, utiliza um modelo supervisionado treinado para identificar e classificar entidades no texto da sentença.

A base de conhecimento do sistema de QA é composta pelos documentos de texto bruto disponibilizados pela coleção *Chave*. Para a indexação e consulta desses documentos, optou-se por utilizar o sistema Solr[12], uma vez que esse abstrai a complexidade de implementação de um sistema de IR. Essa é uma ferramenta *open-souce* utilizada para indexar e consultar documentos, construída a partir da biblioteca *Apache Lucene*. Desse modo, através do sistema *Solr*, foram indexados os 209.282 documentos da coleção Chave.

Para realizar as consultas na base de dados, utilizam-se as *queries* construídas no módulo de Processamento da Pergunta. Porém, primeiramente, precisa-se passar para o formato exigido pelo sistema *Solr*. Assim, transforma-se a *query* de cada pergunta adicionando parâmetros de consultas. Esses parâmetros são utilizados para delimitar o número máximo de documentos que a consulta pode retornar e a estratégia de consulta. Desta forma, foi definido um máximo de 30 documentos a serem recuperados. A estratégia de busca foi definida para priorizar documentos que contenham maior quantidade de palavras da *query* e que estas estejam próximas entre si no documento. Também, esses documentos são recuperados ordenando-se do mais relevante ao menos relevante, permitindo que as demais etapas utilizem a informação da posição do documento neste ranque.

[12]http://lucene.apache.org/solr/

Figura 4.5: Código da função utilizada para recuperar sentenças de documentos.

```
import CoreNLP

def retrieval_passages(questions, ner_model):
    for question in questions:
        question['passages'] = []
        for document in question['documents']:
            passages = CoreNLP.ssplit(document_text)
            for passage in passages:
                entities = ner_model.predict(passage)
                if question['predict_class'] not in entities:
                    continue
                passage_score = 0
                for key_word in question['query'].split():
                    passage_score += passage.count(key_word)
                passage_score = (1+passage_score) * ((-document.rank + MAX_DOC)/MAX_DOC)
                question['passages'].append({
                    'passage': passage, 'entities': entities,
                    'passage_score': passage_score
                    })
    return questions
```

Fonte: O Autor

A seleção de sentenças de documentos recuperados na etapa anterior é realizada através da função '*retrieval_sentences()*', conforme a Figura 4.5. Essa função recebe como entrada a lista de perguntas, através do parâmetro '*questions*', e o modelo de REM já treinado, através do parâmetro '*ner_model*'. Para recuperar as sentenças dos documentos de cada pergunta, essa função primeiramente divide cada documento em sentenças de texto utilizando a biblioteca CoreNLP[13]. Em seguida, para cada sentença, predita-se as entidades mencionadas no texto e seleciona-se somente as que contenham ao menos uma entidade da classe da pergunta predita anteriormente.

Atribui-se uma pontuação para a sentença baseada em dois critérios: a frequência de termos iguais aos da *query*; e a posição do documento da sentença no ranque de documentos recuperados para a pergunta. A pontuação da sentença permite ordená-las pela sua relevância em relação à pergunta de entrada do sistema. Deste modo, de acordo com a função disposta na Figura 4.5 e através da Equação 4.1 criada, expressa-se a pontuação da sentença $SS_{s,dr}$ através de $WQ_{s,q}$, que determina a frequência de termos na *query* q inseridos da sentença s, e através de dr, que determina a posição do documento em seu ranque, em que MD é a quantidade máxima de documentos recuperados. Deste modo, a pontuação da sentença é atribuída de acordo com a frequência de termos comuns a *query* ponderada pela posição do documento em seu ranque.

$$SS_{s,dr,q} = (1 + WQ_{s,q}) \times (\frac{-dr + MD}{MD}) \tag{4.1}$$

[13]https://github.com/eduardogc8/egc-pyutils/blob/master/corenlp.py

4.2.3 Reconhecimento de Entidades Mencionadas

O REM é uma etapa importante do sistema de QA, uma vez que ela é responsável por identificar e classificar entidades no texto que poderão ser utilizadas como resposta final do sistema (ROSA; BARONE, 2018). Emprega-se a técnica principalmente nas etapas de recuperação de sentença e extração de informações, porém, a mesma pode ser utilizada em outras etapas, como na criação da *queries* no Módulo de Processamento da Pergunta. Para o sistema desenvolvido neste trabalho, utiliza-se um modelo supervisionado de aprendizagem de máquina para predizer quais são as entidades mencionadas de um texto de entrada.

Utiliza-se o modelo *Conditional Random Field* (CRF) para predizer as entidades mencionadas nos textos das sentenças. O CRF é um modelo de aprendizado de máquina supervisionado que consiste em uma técnica gráfica probabilística usada para classificar dados sequenciais considerando amostras vizinhas (LAFFERTY; MCCALLUM; PEREIRA, 2001). Assim, dada uma sequência de pares (características e classes), a técnica irá aprender um modelo para classificar novas sequências de dados. A biblioteca *sklearn-crfsuite*[14] foi utilizada para a aplicação do modelo no sistema de QA desenvolvido.

Para treinamento e teste do modelo CRF, utiliza-se a coleção disponibilizada pelo Harem[15], que é uma iniciativa para identificação e classificação automática dos nomes próprios na língua portuguesa. A coleção dispõe de duas bases de dados douradas, ou seja, coleções onde especialistas em linguística anotaram e revisaram as entidades mencionadas contidas nos textos (FREITAS et al., 2010). O Harem disponibiliza textos anotados com seguintes classes de entidades: TEMPO, VALOR, OBRA, LOCAL, ORGANIZACAO, PESSOA, ABSTRACCAO, COISA, ACONTECIMENTO, OBJECTO e OUTRO. Para o sistema de QA desenvolvido neste trabalho, somente as classes TEMPO, VALOR, LOCAL, ORGANIZACAO e PESSOA foram utilizadas, já que a coleção Chave não contém perguntas das demais classes. Mesmo assim, o modelo foi treinado utilizando todas as classes, visto que o mesmo está disponível para uso[16], desta forma uma maior abrangência de classes possibilita que o mesmo seja aplicado em uma gama maior de problemas.

O treinamento do modelo CRF utilizando a coleção Harem foi realizada aplicando características retiradas da sentença. Neste caso, utilizou-se como característica para a classificação de cada palavra da sentença a própria palavra e às duas anteriores a ela. Além disso, verifica-se se a palavra começa com caractere maiúsculo, se todas as letras da palavra são maiúsculas e se a mesma é um digito.

4.2.4 Processamento da Resposta

O módulo de Processamento da Resposta é responsável por extrair respostas candidatas tendo como base a lista de sentenças recuperadas no módulo de Recuperação de Informações, criando uma lista de respostas candidatas e selecionando a resposta final a partir dessa lista. Para isso, implementa-se duas funções responsáveis por extrair da lista de sentenças as respostas candidatas mais propícias

[14] https://sklearn-crfsuite.readthedocs.io
[15] https://www.linguateca.pt/HAREM/
[16] https://github.com/eduardogc8/qa-chave

e seleciona-se a resposta final a partir de uma pontuação atribuída a cada resposta candidata. Por fim, o sistema de QA retorna como saída a resposta candidata com maior pontuação selecionada.

A função '*answer_candidates()*' responsabiliza-se por extrair respostas candidatas com base na lista de passagens recuperadas no módulo de Recuperação de Informações. Conforme a Figura 4.6, a função recebe como entrada uma lista de perguntas, através do parâmetro '*questions*', em que para cada pergunta da lista, analisa-se todas as passagens recuperadas, buscando-se ao menos uma entidade da mesma classe da pergunta. Quando uma sentença contem ao menos uma entidade mencionada da classe da pergunta, a função extrai a entidade para ser uma resposta candidata e atribui diferentes critérios de pontuação que serão utilizados para calcular a pontuação final da resposta candidata. Os critérios de pontuação utilizados são: I) '*votes*', referente à quantidade de vezes que essa resposta foi extraída; II) '*doc_rank*', referente ao ranque do documento em que a passagem analisada foi recuperada; III) '*passage_score*', referente a pontuação de ranque da passagem calculada no módulo de Recuperação de Informações.

A resposta final do sistema é selecionada através da função '*final_answers()*', que se responsabiliza por atribuir uma pontuação final para cada resposta candidata e seleciona a mais propícia como resposta final do sistema. Conforme a Figura 4.7, a função recebe como entrada uma lista de perguntas, através do parâmetro '*questions*', em que para cada pergunta da lista, atribui-se uma pontuação final utilizando o parâmetro global '*WEIGHTS*' para ponderar a pontuação individual de cada critério de pontuação. Observa-se, através de testes e análise empírica, que a essa etapa atinge melhores resultados utilizando os seguintes pesos no parâmetro '*WEIGHTS*': '*doc_rank*' igual a -0.9; '*votes*' igual a 0.7; e '*passage_score*' igual a 0.7. Além disso, essa etapa normaliza cada parâmetro individualmente entre os valores 0.0 e 1.0 através da função '*normalize_parameters()*'.

4.3 Base de Dados

Esta seção destina-se a apresentar as bases de dados utilizadas para testar e avaliar as abordagens propostas neste trabalho de pesquisa. A principal base de dados utilizada foi a coleção Chave, uma vez que essa contém características importantes para a avaliação do modelo híbrido de classificação de pergunta, como também é composta pelos dados necessários para avaliação de um sistema de QA completo e suas diferentes etapas. Além dessa, foram utilizadas outras bases de dados para testar a abordagem híbrida em diferentes coleções e também uma coleção para treinamento de um modelo supervisionado para REM.

Figura 4.6: Código da função utilizada para extrair respostas candidatas.

```
def answer_candidates(questions):

    candidates = []

    for question in questions:
        question['answer_candidates'] = []
        for passage in question['passages']:
            control = False
            for entity in passage['entities']:
                if entity != 'O':  # Se é uma entidade mencionada
                    class_entity = entity[:entity.index('-')].lower()
                    if class_entity == question['predict_class'].lower():
                        control = True
                        break
            if not control:
                continue

            candidate = {'answer': '', 'votes': 0, 'doc_rank': passage['doc_rank'],
                         'passage_score': passage['passage_score']}
            last = False
            words = passage['passage'].split()
            for index in range(len(words)):
                word = words[index]
                entity = passage['entitys'][index]

                if entity != 'O':  # Se é uma entidade mencionada
                    class_entity = entity[:entity.index('-')].lower()
                    suffix = entity[entity.index('-') + 1:].lower()
                    if suffix == 'b':  # Inicio da entidade mencionada
                        if candidate['answer'] != '':
                            insert_candidate(question, candidate)
                            candidate = {'answer': '', 'votes': 0,
                                         'doc_rank': passage['doc_rank'],
                                         'passage_score': passage['passage_score']}
                        if class_entity == question['predict_class'].lower():
                            candidate['answer'] += ' ' + word
                            last = True
                        else:
                            last = False
                    else:
                        if last:
                            if candidate['answer'] != '':
                                insert_candidate(question, candidate)
                                candidate = {'answer': '', 'votes': 0,
                                             'doc_rank': passage['doc_rank'],
                                             'passage_score': passage['passage_score']}
                        last = False
            if candidate['answer'] != '':
                insert_candidate(question, candidate)
                candidate = {'answer': '', 'votes': 0, 'doc_rank': passage['doc_rank'],
                             'passage_score': passage['passage_score']}
    return questions
```

Fonte: O Autor

Figura 4.7: Código da função utilizada para selecionar a resposta final.

```
WEIGHTS = {'doc_rank': -0.9, 'votes': 0.7, 'passage_score': 0.7}


def final_answers(questions):

    for question in questions:
        question['final_answer'] = ''
        normalize_parameters(question['answer_candidates'])

        # Determinar o score de cada answer candidate
        for candidate in question['answer_candidates']:
            candidate['score'] = 0
            for weight in WEIGHTS:
                candidate['score'] += candidate[weight] * WEIGHTS[weight]

        # Atribui a resposta final para cada questão
        if len(question['answer_candidates']) > 0:
            rank = sorted(question['answer_candidates'], key=lambda k: k['score'])
            question['final_answer'] = rank[-1]['full_answer']

    return questions
```

Fonte: O Autor

4.3.1 Coleção Chave

Para testar e avaliar as abordagens propostas, utiliza-se a coleção Chave (MAGNINI et al., 2006). Essa é uma base de dados com perguntas e respostas na língua portuguesa. A coleção é composta de aproximadamente 4.000 perguntas, sendo que cerca de 1.000 dessas, contêm ao menos uma resposta. Além disso, a coleção oferece aproximadamente 210.000 documentos de texto bruto para ser utilizado como base de conhecimento do sistema de QA. Esses documentos foram retirados de todas as edições dos jornais impressos PÚBLICO e FOLHA nos anos de 1994 e 1995. Todas as perguntas da coleção podem ser respondidas com informações retiradas dessas coleções de documentos.

Selecionou-se a coleção Chave neste trabalho uma vez que essa é comumente utilizadas em pesquisa de QA para língua portuguesa, como também foi construída para edições da competição do CLEF [17] dos anos de 2004 a 2008 com ajuda do Linguateca [18]. Estas duas são instituições reconhecidas nas áreas de PLN, Recuperação de Informações e Extração de Informações, o que faz com a Chave uma coleção padrão na pesquisa de QA para a língua portuguesa.

A Figura 4.8 apresenta a estrutura XML utilizada para representar uma pergunta na coleção Chave. Além de perguntas, respostas e documentos para consulta, a coleção Chave contém anotações em cada pergunta, o que permite avaliar diferentes etapas de um sistema de QA. Os atributos 'categoria' e 'tipo' permitem treinar e testar um modelo supervisionado de classificação de perguntas.

[17] http://www.clef-campaign.org
[18] https://www.linguateca.pt

Figura 4.8: Estrutura XML para representar uma pergunta na coleção Chave.

```
<pergunta ano="2004" id_org="0470" categoria="F" tipo="LOCATION" restrição="X"
          ling_orig="IT" tarefa_pt="0002" tarefa_it="0020" tarefa_nl="0007">

    <texto>
        Onde era o campo de concentração de Auschwitz?
    </texto>

    <resposta n="1" docid="LING-940804-089">
        Sul da Polónia
    </resposta>

    <resposta n="2" docid="LING-941120-083">
        Polónia
    </resposta>

    <resposta n="3" docid="LING-950126-162">
        Sudoeste da Polónia ocupada
    </resposta>

</pergunta>
```

Fonte: O Autor

Também, o atributo 'docid' das *tags* resposta permite avaliar o módulo de Recuperação de Informações, já que esse identifica o documento relevante para a resposta.

4.3.2 Pré-processamento

Para testar as abordagens de classificação de perguntas, utilizam-se os atributos 'categoria' e 'tipo' da coleção Chave. Porém, observou-se que essa anotação não estava consistente originalmente, uma vez que valores de um atributo poderiam aparecer em um atributo diferente, como também, os valores muitas vezes não estavam anotados de uma forma padronizada. Desta forma, foi necessário realizar um pré-processamento na coleção para criar um novo atributo chamado 'classe', que contém a classe correta da pergunta. Além disso, algumas classes de perguntas foram unidas, uma vez que estas eram similares e a abordagem ajudou a diminuir o problema de uma distribuição de classes desbalanceada.

A Figura 4.9 apresenta o código utilizado para normalizar e criar o atributo classe para cada pergunta. A Tabela 4.1 apresenta a distribuição de categoria e tipo da coleção Chave, onde as categorias/tipos com coloração de fundo foram unidas. A Tabela 4.2 apresenta a distribuição de classes depois da execução do pré-processamento, onde as classes com coloração de fundo azul foram utilizadas para treinamento e teste das abordagens de classificação de pergunta e sistema de QA completo, enquanto que as de coloração amarela foram utilizadas apenas no teste e treinamento das abordagens de classificação de pergunta.

Figura 4.9: Código utilizado para normalizar as classes da coleção Chave.

```
def pair_classification(category, type_):
    if category == 'COUNT':
        return 'MEASURE'
    if category == 'D' or category == 'DEFINITION':
        return 'DEFINITION'
    if category == 'F' or category == 'FACTOID':
        if type_ == 'COUNT':
            return 'MEASURE'
        elif type_ == 'MANNER':
            return 'DEFINITION'
        return type_
    if category == 'L' or category == 'LIST':
        if type_ == 'COUNT':
            return 'MEASURE'
        else:
            return type_
    if category == 'LOCATION':
        return 'LOCATION'
    if category == 'MEASURE':
        return 'MEASURE'
    if category == 'OBJECT':
        return 'DEFINITION'
    if category == 'ORGANIZATION':
        return 'ORGANIZATION'
    if category == 'OTHER' and (type_ == 'FACTOID' or type_ == 'LIST'):
        return 'OTHER'
    if category == 'OTHER' and not (type_ == 'FACTOID' or type_ == 'LIST'):
        if type_ == 'MANNER':
            return 'DEFINITION'
        return type_
    if category == 'PERSON' and type_ == 'DEFINITION':
        return 'DEFINITION'
    if category == 'PERSON' and not type_ == 'DEFINITION':
        return 'PERSON'
    return category
```

Fonte: O Autor

Tabela 4.1: Distribuição de classes da coleção Chave original.

Fonte: O Autor

Categorias/Tipos	Quantidade Categoria	Quantidade Tipo
COUNT	19	100
D	492	-
DEFINITION	110	28
F	2.036	-
FACTOID	609	162
L	52	-
LIST	77	10
LOCATION	39	506
MANNER	-	26
MEASURE	15	368
OBJECT	7	139
ORGANIZATION	16	548
OTHER	42	571
PERSON	38	778
TIME	24	340
X	800	800
Total	**4.376**	**4.376**

Tabela 4.2: Distribuição de classes da coleção Chave pré-processada.

Fonte: O Autor

Classe	Quantidade
DEFINITION	650
LOCATION	545
MEASURE	502
OBJECT	86
ORGANIZATION	356
OTHER	491
PERSON	582
TIME	364
X	800
Total	**4.376**

Uma vez que o sistema de QA desenvolvido neste trabalho trabalha somente com perguntas do tipo factoide, é necessário selecionar quais classes de perguntas serão utilizadas para realizar o teste com o sistema. Neste caso, conforme a Tabela 4.2, foram selecionados as perguntas das classes com fundo azul. A classe 'X' refere-se às perguntas com classes desconhecidas ou que não foram anotadas. Desta forma, foram ignoradas durante os testes. Ignora-se a classe 'OBJECT' devido à baixa quantidade de dados disponíveis e por apresentar perguntas que podem ser classificadas em outras classes. Os testes com o sistema de QA não utilizam a classe 'OTHER'.

4.3.3 Outras Coleções

Além da coleção Chave, outras bases de dados são utilizadas para testes de abordagens de classificação da pergunta, também utiliza-se uma coleção para treinar um modelo de REM. A avaliação com diferentes bases de dados oferece uma maior confiabilidade dos resultados, uma vez que se avalia a capacidade de generalização do modelo. Deste modo, avaliam-se as abordagens de classificação de pergunta tanto com a coleção Chave, como também como uma coleção exclusiva para o problema de classificação de pergunta que está disponível em Inglês e através de uma versão traduzida para o Português.

A coleção UIUC é uma base de dados proposta no trabalho de (LI; ROTH, 2002). A base é definida em duas camadas de classes, a primeira camada contém 6 classes, enquanto que a segunda

Tabela 4.3: Distribuição de classes da coleção UIUC.
Fonte: O Autor

Classe	Quantidade Treinamento	Quantidade Teste
ABBR	85	9
DESC	1164	138
ENTY	1250	94
HUM	1225	65
LOC	837	81
NUM	896	113
Total	**5457**	**500**

Tabela 4.4: Distribuição de classes da coleção Harem.
Fonte: O Autor

Classes	Quantidade
ABSTRACCAO	404
ACONTECIMENTO	128
COISA	133
LOCAL	1231
OBJECTO	2
OBRA	196
ORGANIZACAO	918
OUTRO	39
PESSOA	1033
TEMPO	435
VALOR	463
Total	**4.982**

contém 50 subclasses, ou seja, uma pergunta pode ser classificada em uma das 6 classes da primeira camada e em uma das 50 subclasses da segunda camada. Essa coleção está disponibilizada originalmente em inglês, porém, o trabalho de (COSTA et al., 2012) disponibiliza uma versão dessa mesma coleção traduzida para a língua portuguesa manualmente. Apresenta-se a distribuição de classes da primeira camada na Tabela 4.3.

Para treinar e testar o modelo de REM utilizado no sistema de QA completo, utiliza-se a coleção Harem. Para este trabalho de pesquisa, buscou-se uma base de dados que disponibiliza a marcação de entidades mencionadas em texto. A base de dados dispõe de duas coleções douradas, ou seja, coleções onde especialistas em linguística anotam e revisam as entidades mencionadas contidas nos textos. A coleção utilizada nesta pesquisa disponibiliza cerca de 5.000 entidades mencionadas em 130 documentos. A distribuição de classes que compõe a coleção está disposta na Tabela 4.4. As linhas coloridas são referentes às classes utilizadas pelo sistema de QA desenvolvido.

4.4 Métricas de Avaliação

Esta seção destina-se a apresentar as metodologias de avaliação. Para isso, apresentam-se as estratégias e métricas de avaliação utilizadas para avaliar tanto a etapa de classificação da pergunta, como também as demais etapas de um sistema de QA. Além disso, são apresentados os testes estatísticos utilizados para validar as hipóteses de pesquisa. Diferente da avaliação das abordagens de classificação da pergunta, onde se utiliza toda a coleção para teste e treinamento, a avaliação do sistema de QA e de suas etapas são realizadas através de uma divisão fixa de dados de treinamento e teste, uma vez que não é possível avaliá-lo sem a resposta correta previamente anotada e apenas cerca de 1.000 perguntas da coleção contém ao menos uma resposta. Deste modo, utilizam-se as perguntas que contenham ao menos uma resposta para teste e o restante são utilizadas para treinamento da abordagem de classificação da pergunta.

4.4.1 Avaliação de Classificação de Pergunta

A avaliação das abordagens de classificação de pergunta consiste em utilizar uma parcela da coleção de dados para testar modelos treinados com outra parcela da coleção. Realiza-se a divisão entre dados de treinamento e dados de teste através da técnica de validação cruzada, que consiste em dividir a coleção em diferentes pedaços, em que se utiliza cada um desses para teste enquanto que os demais são utilizados para treinamento. Essa técnica permite uma melhor avaliação da capacidade de generalização do modelo avaliado, já que variam-se os dados utilizado para treinamento e teste. Além disso, avaliam-se as abordagens de classificação de pergunta variando o tamanho da coleção de dados de treinamento, uma vez que para validar a hipótese dessa pesquisa deve-se avaliar as abordagens em conjuntos pequenos até conjuntos maiores para o treinamento de modelos.

Conforme a Tabela 4.2, a distribuição de classes da coleção Chave está desbalanceada. Por exemplo, a classe '*ORGANIZATION*' contém 356 instâncias enquanto que a classe '*DEFINITION*' contém 650 classes. Deste modo, quando a distribuição de classes é desbalanceada, a métrica de acurácia é considerada uma escolha pobre, uma vez que modelos que apenas predizem as classes mais frequentes podem ter um bom desempenho. Assim, são utilizadas três métricas para avaliar essa etapa: precisão, que é a fração de instâncias recuperadas que são relevantes; revocação, que é a fração de instâncias relevantes que são recuperadas; e *F1-score* que é a medida que combina precisão e revocação através da média harmônica de precisão e revocação, disposta na Equação 4.2.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.2}$$

Avaliam-se as abordagens de classificação de pergunta variando o tamanho do conjunto de treinamento, de modo em que cada interação, aumenta-se gradualmente esse tamanho. Desta forma, para cada interação, extrai-se o número correspondente de instância da base de dados em posições aleatórias para o treinamento dos modelos supervisionados com as diferentes abordagens. Para cada interação, repete-se esse processo 5 vezes e calcula-se a média aritmética das métricas de resultados de modo a reduzir ruídos e viés, buscando treinar e testar todo o conjunto de dados. Assim, em cada interação, os dados que não são utilizados para treinamento, são utilizados para teste.

4.4.2 Avaliação de Recuperação de Informações

Como o módulo de Recuperação de Informações consiste em diferentes etapas que podem ser avaliadas, testa-se a etapa de recuperação de documentos, recuperação de sentenças e o REM. Avalia-se a etapa de recuperação de documentos pela possibilidade de utilizar o atributo '*docid*' contidos nas tags 'resposta' de cada pergunta da coleção Chave. Uma vez que a coleção Chave disponibiliza os documentos de consulta em texto bruto, as sentenças não estão separadas, como também não contém anotações de quais sentenças são relevantes para uma determinada pergunta. Deste modo, para avaliar a etapa de recuperação de sentença, assume-se que a sentença é relevante caso a mesma contenha em seu texto ao menos uma das respostas da pergunta. Avalia-se a etapa de REM utilizando os dados anotados da coleção Harem.

Avaliam-se as etapas de recuperação de documentos e recuperação de sentenças utilizando as métricas de precisão, revocação e *F1-Score*. Nestas etapas, atribui-se maior importância para a métrica de revocação, uma vez que se espera que o sistema de QA consiga responder de forma correta somente se tiver as informações necessárias para responder a pergunta. Além disso, avaliam-se as etapas através da criação de uma curva de relação entre precisão por documentos ou sentenças recuperadas. Essa curva permite analisar uma melhor configuração de número máximo de documentos ou sentenças a serem recuperadas pelo sistema. Por fim, a avaliação da etapa de recuperação de documentos estará avaliando indiretamente a etapa do módulo de Processamento da Pergunta responsável por gerar uma *query* a ser utilizada nesta etapa, uma vez que a configuração de busca e palavras da *query* gerada tem impacto no desempenho dessa etapa.

Para a etapa de REM, utiliza-se a coleção Harem para avaliação através da técnica de validação cruzada com 5 *folds*. Uma vez, que é inevitável treinar e testar o modelo CRF utilizando palavras que não são entidades mencionadas, a coleção se torna muito desbalanceada, já que mais de 90% da coleção será composta por instâncias que não são entidades. Deste modo, utiliza-se a classe 'O' (instâncias que não são entidades mencionadas) para treinamento e teste, mas desconsidera-se ao calcular as métricas de avaliação. Mesmo assim, utilizam-se as métricas de precisão, revocação e *F1-score* para avaliar a etapa. Após os testes com a etapa, treina-se um modelo final com toda coleção para ser disponibilizado e utilizado no sistema de QA completo.

4.4.3 Avaliação de Processamento da Resposta

Para o módulo de Processamento da Resposta, avalia-se a capacidade do mesmo em extrair respostas candidatas e formular um ranque a partir de uma pontuação dada a cada resposta candidata. Para isso, utilizam-se as perguntas que contenham ao menos uma resposta da coleção Chave. Desta forma, para checar se uma determinada resposta está correta, verifica-se se a mesma é igual a pelo menos uma das respostas dispostas na coleção. Esse método de checagem permite a verificação automática das respostas, sem a necessidade de uma interversão humana. Porém, respostas que estão corretas, mas expressadas em texto de forma diferente da lista de respostas corretas da coleção, serão avaliadas como incorretas. Essa etapa está avaliando o sistema de QA como um todo, uma vez que essa é a última etapa do sistema, o desempenho dessa depende do desempenho de todas as etapas anteriores.

Avalia-se o módulo de Processamento da Resposta através da criação de uma curva de relação entre precisão de respostas corretas por respostas candidatas extraídas. Essa curva permite analisar o desempenho da pontuação de cada resposta candidata. Ou seja, caso a abordagem de atribuição de pontuação tenha um bom desempenho, as respostas corretas estarão nas primeiras colocações do ranque, caso contrário, as mesmas estarão distribuídas em outras posições. Por fim, é possível avaliar a capacidade final do sistema de QA, observando a precisão do mesmo em atribuir uma resposta correta na primeira colocação do ranque de respostas candidatas.

4.4.4 Teste Estatístico

O teste estatístico é um procedimento que permite verificar duas ou mais hipóteses, utilizando os dados observados do experimento. Nesta pesquisa, utiliza-se o teste-t para verificar se a abordagem híbrida proposta nesta pesquisa tem melhor desempenho do que as abordagens de *baselines* variando o tamanho do conjunto de treinamento. Desta forma, o teste-t permite aceitar ou rejeitar a hipótese nula, que no caso deste trabalho de pesquisa, o teste deve aceitar ou rejeitar que a abordagem híbrida não tem melhor desempenho do que as abordagens de *baseline* variando o tamanho do conjunto de treinamento para a tarefa de classificação de pergunta. Realiza-se a aplicação do teste estatístico através da biblioteca *scipy*[19]

Testam-se as abordagens de classificação de pergunta, variando o tamanho do conjunto de dados. Deste modo, compara-se o desempenho das diferentes abordagens para classificação de pergunta para diferentes tamanhos de conjunto de treinamento. Utiliza-se o teste-t para verificar se a abordagem híbrida, através da hipótese H, tem desempenho estatisticamente superior que as arbordagens de *baseline*. Para isso, aplica-se o teste-t com um nível de significância de 0.05. Deste modo, se o *valor-p* do teste estatístico for menor do que 0.05, signifca que abordagem híbrida obteve desempenho estatisticamente superior, já que se rejeita a hipótese nula $H0$. O *valor-p* representa uma medida de evidência contra a hipótese nula, neste caso, *valor-p* < 0.05 significa que existe uma evidência forte contra a hipótese nula $H0$, permitindo rejeitá-la.

[19]https://www.scipy.org/

5 EXPERIMENTOS E ANÁLISE DOS RESULTADOS

Este capítulo destina-se a apresentar os experimentos realizados com seus resultados gerados e suas análises. O capítulo é dividido em três seções. A primeira é responsável por apresentar os experimentos, resultados e sua análise na tarefa de classificação de pergunta utilizando as abordagens híbridas e *baselines* com diferentes modelos supervisionados de aprendizado de máquina. Já a segunda seção, aborda os experimentos, resultados e sua análise para o sistema de QA desenvolvido para a língua portuguesa. Por fim, a terceira seção apresenta as considerações gerais sobre os resultados gerados nos dois experimentos.

5.1 Classificação da Pergunta

Os experimentos e resultados efetuados a partir dos testes realizados com as abordagens de classificação de pergunta são apresentados na próxima subseção. Esses testes utilizam a coleção Chave e são gerados com diferentes modelos supervisionados de aprendizado de máquina. Varia-se o tamanho do conjunto de treinamento utilizando validação cruzada com 5 *folds*. Para cada um destes modelos, apresenta-se ao final dos testes o gráfico de *F1-Score* por tamanho de conjunto de treinamento, as matrizes de confusão e a tabela de desempenho mostrando precisão, revocação e *F1-Score* de cada abordagem. Para estas tabelas, utilizam-se os resultados das abordagens aplicando o tamanho máximo de dados para treinamento. Ao final, disponha-se o gráfico de tempo de execução para o treinamento dos modelos de cada abordagem e realiza-se testes com a coleção UIUC. Por fim, conclui-se essa seção com uma subseção de análise dos resultados.

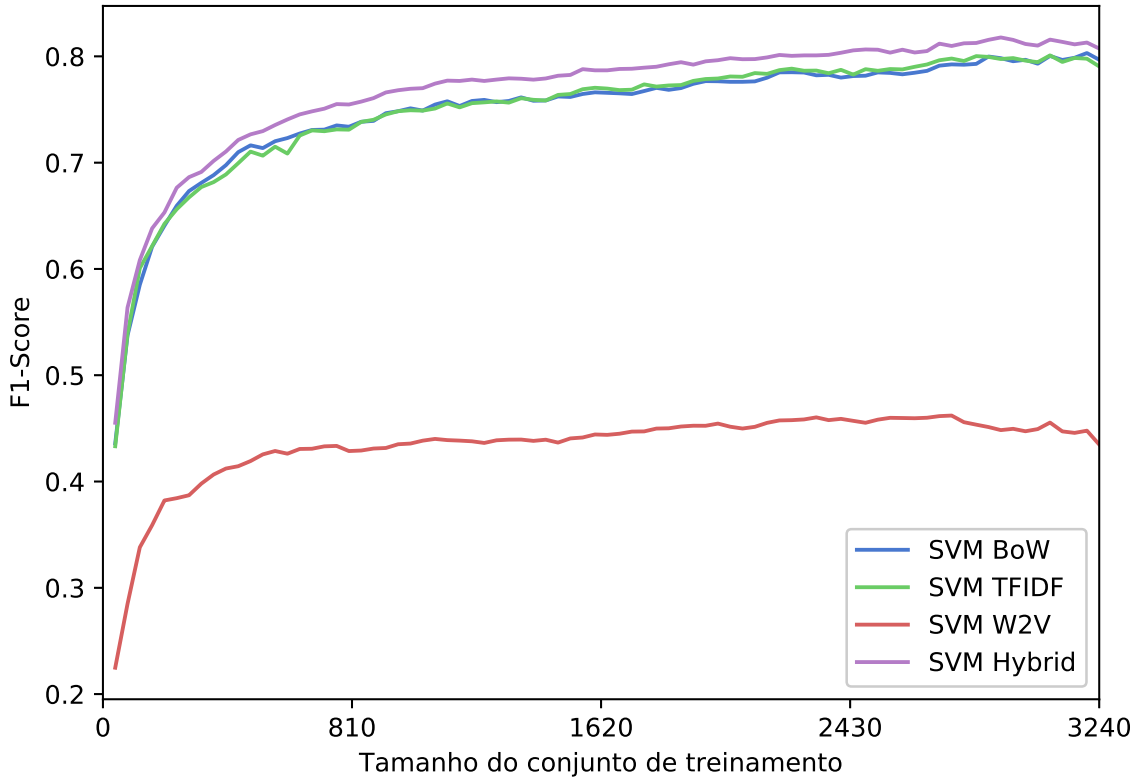
5.1.1 Experimentos e Resultados

Os experimentos realizados aplicaram três diferentes modelos de aprendizado de máquina, SVM, MLP e LSTM. Para todos os modelos, as abordagens utilizaram um vocabulário de 1.000 termos. Ou seja, os 1.000 termos mais frequentes para as abordagens que utilizam *Bag-of-words* ou TF-IDF. Isso evita aplicar um vetor com todo o vocabulário de termos para os modelos supervisionados, uma vez que o vocabulário total de um corpus de texto pode ultrapassar facilmente o tamanho de 50.000 termos, deste modo, utilizam-se apenas os mais frequentes.

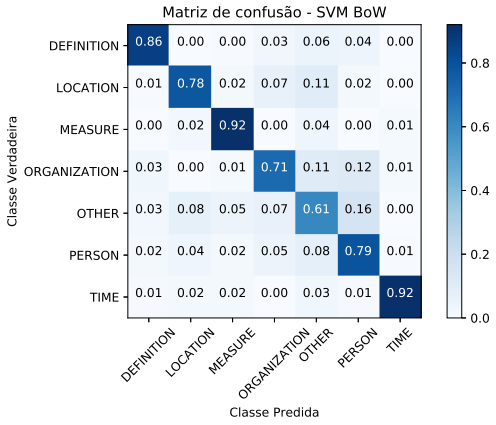
As matrizes de confusão apresentadas permitem analisar a efetividade das abordagens para cada classe de pergunta, uma vez que as linhas da mesma representa as instâncias em uma classe predita e cada coluna representa as instâncias em uma classe real. As tabelas de desempenho permitem avaliar a performance de cada abordagem através das métricas de precisão, revocação e *F1-Score*. A precisão permite mensurar a fração de instâncias recuperadas que são relevantes enquanto que a revocação permite mensurar a fração de instâncias relevantes que são recuperadas. Por fim, o *F1-Score* mensura a performance das abordagens baseando-se tanto na precisão quando na revocação. Os próximos resultados apresentados foram gerados utilizando a coleção Chave. Ao final da subseção, apresentam-se os resultados gerados utilizando a coleção UIUC.

Para o experimento das abordagens com o modelo SVM, utiliza-se um modelo com *kernel* linear, uma vez que se observou que essa configuração de *kernel* apresenta melhor resultado do que as demais para a tarefa de classificação de perguntas com a coleção Chave. Assim, supõe-se que o problema é linearmente divisível. Também, os hiper parâmetros utilizados são os padrões já configurados pela biblioteca *scikit-learn*. Desta forma, a função de custo utilizada foi '*squared_hinge*' e o valor para o parâmetro de penalidade C foi de 1.0. Os resultados do experimento estão dispostos na Figura 5.1, que apresenta o *F1-Score* pelo tamanho do conjunto de treinamento, na Figura 5.2 que apresenta as matrizes de confusão das abordagens e na Tabela 5.1, que apresenta a precisão, revocação e *F1-Score* para o maior conjunto de treinamento.

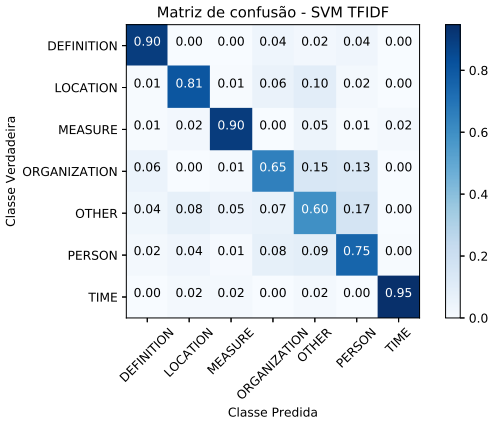
Figura 5.1: *F1-Score* variando tamanho dos dados de treinamento das abordagens de classificação de pergunta utilizando SVM.



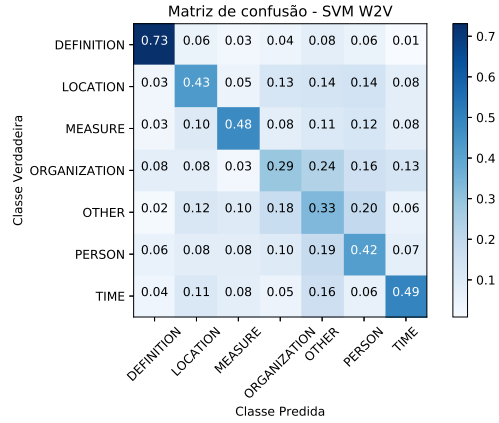
Fonte: O Autor



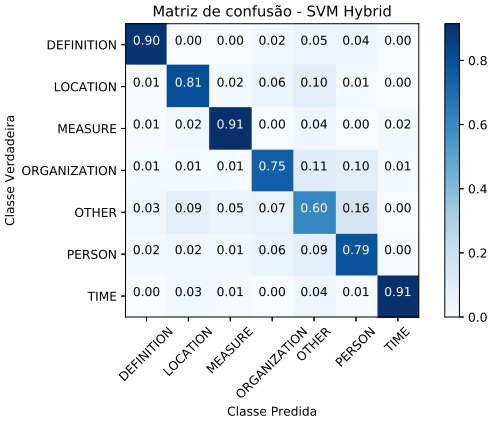
(a) *Bag-of-words*



(b) TFIDF



(c) *Word2vec*



(d) Híbrido

Figura 5.2: Matrizes de Confusão das abordagens utilizando o modelo SVM.
Fonte: O Autor

Conforme os resultados dispostos na Figura 5.1, a abordagem híbrida, denominada na legenda como *SVM Hybrid*, obteve a melhor performance em todos os tamanhos de conjunto de treinamento para o modelo SVM, em que obteve resultados em média 3 pontos percentuais melhores do que os das abordagens *Bag-of-words* e TF-IDF. A abordagem *word2vec* obteve os piores resultados, em que atingiu resultados em média 35 pontos percentuais piores do que os das abordagens *Bag-of-words* e TF-IDF. As matrizes de confusão dispostas na Figura 5.2, mostram que *DEFINITION, MEASURE* e *TIME* foram as classes melhores classificadas, enquanto que a classe *OTHER* foi classificada erroneamente mais vezes. Também, a abordagem *word2vec* obteve 73 pontos percentuais na classe *DEFINITION*, enquanto que as demais não passaram dos 49 pontos percentuais. A Tabela 5.1 mostra que a abordagem híbrida obteve performance superior nas métricas de precisão, revocação e *F1-Score* utilizando 3.240 instâncias para o treinamento.

Tabela 5.1: Desempenho final das abordagens utilizando SVM e 3.240 instâncias para treinamento.
Fonte: O Autor

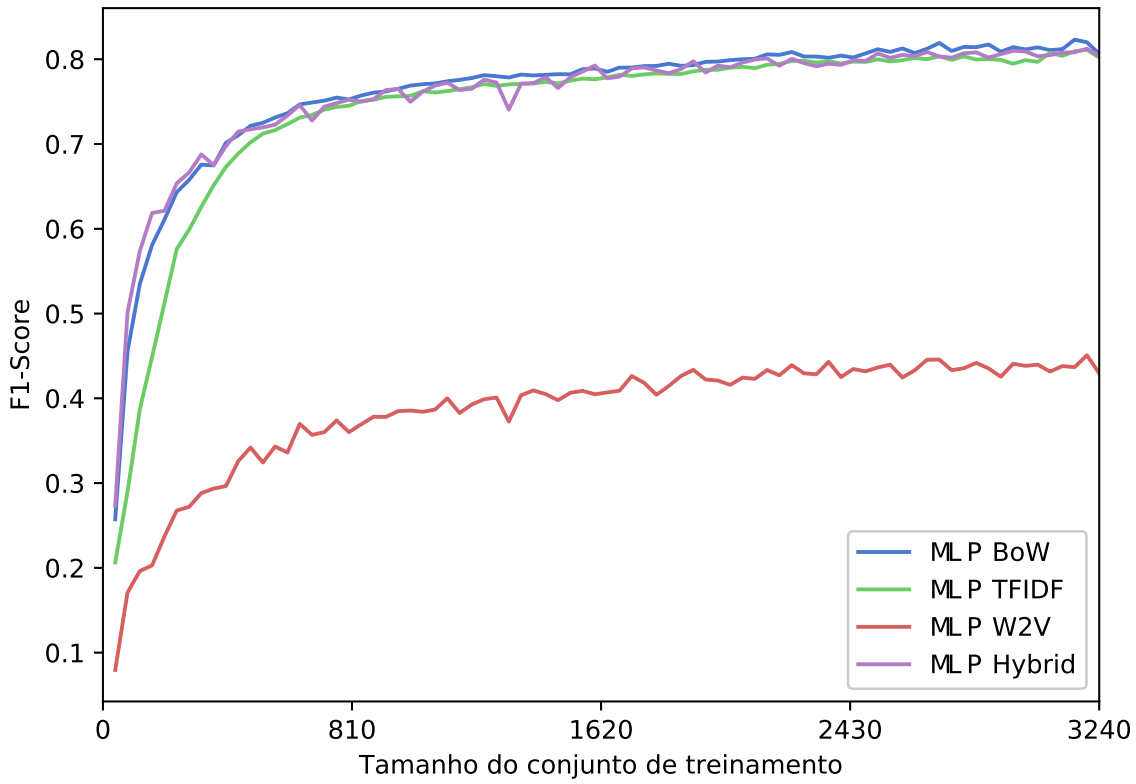
Modelo	**Precisão**	**Revocação**	***F1-Score***
SVM BoW	0.7981	0.8023	0.7967
SVM TFIDF	0.7904	0.7950	0.7905
SVM W2V	0.4587	0.4577	0.4347
SVM Hybrid	**0.8072**	**0.8132**	**0.8073**

Para o experimento das abordagens com o modelo MLP, utilizam-se redes compostas por uma camada de entrada, uma camada intermediária densa com *dropout* de 20 % e uma camada de saída com 7 neurônios. O tamanho da camada de entrada varia conforme a abordagem, em que para as abordagens *Bag-of-words* e *TF-IDF* utilizam-se 1.000 entradas (tamanho do vocabulário), para a abordagem *Word2vec* utilizam-sem 300 entradas (número de dimensões *word embedding*) e para a abordagem híbrida utilizam-se 1.300 entradas (vocabulário + dimensões *word embedding*). Também, para acompanhar a variação de tamanho da camada de entrada, a quantidade de neurônios da camada intermediária varia para cada abordagem, em que para a *Bag-of-words* e *TF-IDF* utilizam-se 500 neurônios, para a *Word2vec* utilizam-sem 150 neurônios e para a híbrida utilizam-se 650 neurônios. Os modelos são treinados em 10 épocas utilizando uma taxa de aprendizado de 0.001. Os resultados do experimento estão dispostos na Figura 5.3, que apresenta o *F1-Score* pelo tamanho do conjunto de treinamento, na Figura 5.4 que apresenta as matrizes de confusão das abordagens e na Tabela 5.2, que apresenta a precisão, revocação e *F1-Score* para o maior conjunto de treinamento.

Tabela 5.2: Desempenho final das abordagens utilizando MLP e 3.240 instâncias para treinamento.
Fonte: O Autor

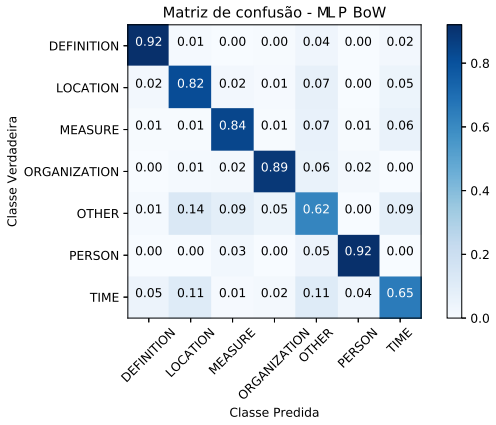
Modelo	**Precisão**	**Revocação**	***F1-Score***
MLP BoW	0.8064	**0.8112**	**0.8064**
MLP TFIDF	0.7997	0.8091	0.8020
MLP W2V	0.4455	0.4597	0.4295
MLP Hybrid	**0.8084**	0.8077	0.8042

Figura 5.3: *F1-Score* variando tamanho dos dados de treinamento das abordagens de classificação de pergunta utilizando MLP.

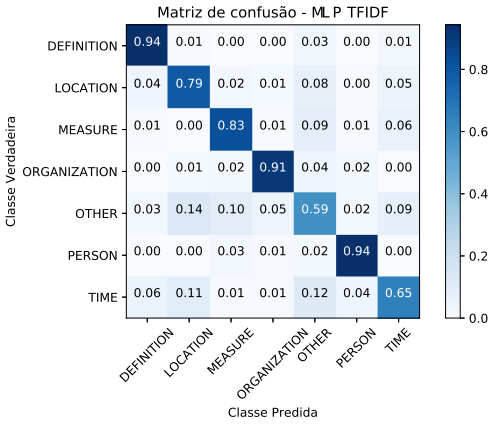


Fonte: O Autor

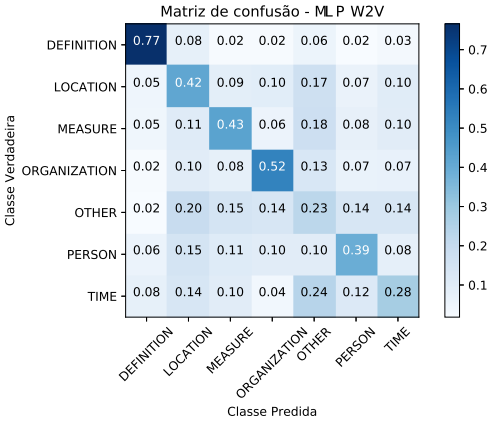
Conforme os resultados dispostos na Figura 5.3, as abordagens híbridas, *Bag-of-words* e TF-IDF atingiram as melhores performances no modelo MLP, em que obtiveram resultados em média 35 pontos percentuais melhores do que os da abordagem *Word2vec*. As matrizes de confusão dispostas na Figura 5.4, mostram que na maioria das vezes *DEFINITION, LOCATION, MEASURE, ORGANIZATION* e *PERSON* foram as classes melhores classificadas, enquanto que as classes *OTHER* e *TIME* foram classificadas erroneamente mais vezes. Também, a abordagem *word2vec* obteve 77 pontos percentuais na classe *DEFINITION*, enquanto que as demais não passaram dos 52 pontos percentuais. A Tabela 5.2 mostra que a abordagem híbrida obteve performance superior apenas na métrica de precisão, enquanto que para as métricas revocação e *F1-Score* a abordagem *Bag-of-words* obteve melhor performance utilizando 3.240 instâncias para o treinamento.



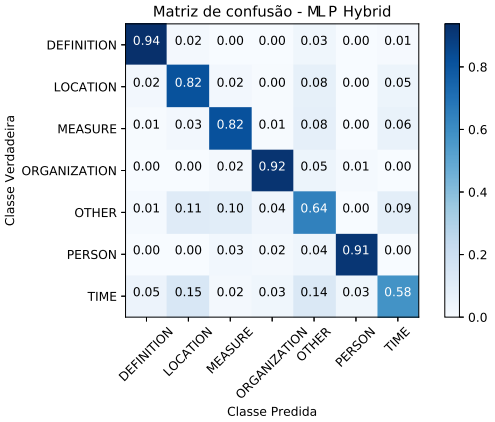
(a) *Bag-of-words*



(b) TFIDF



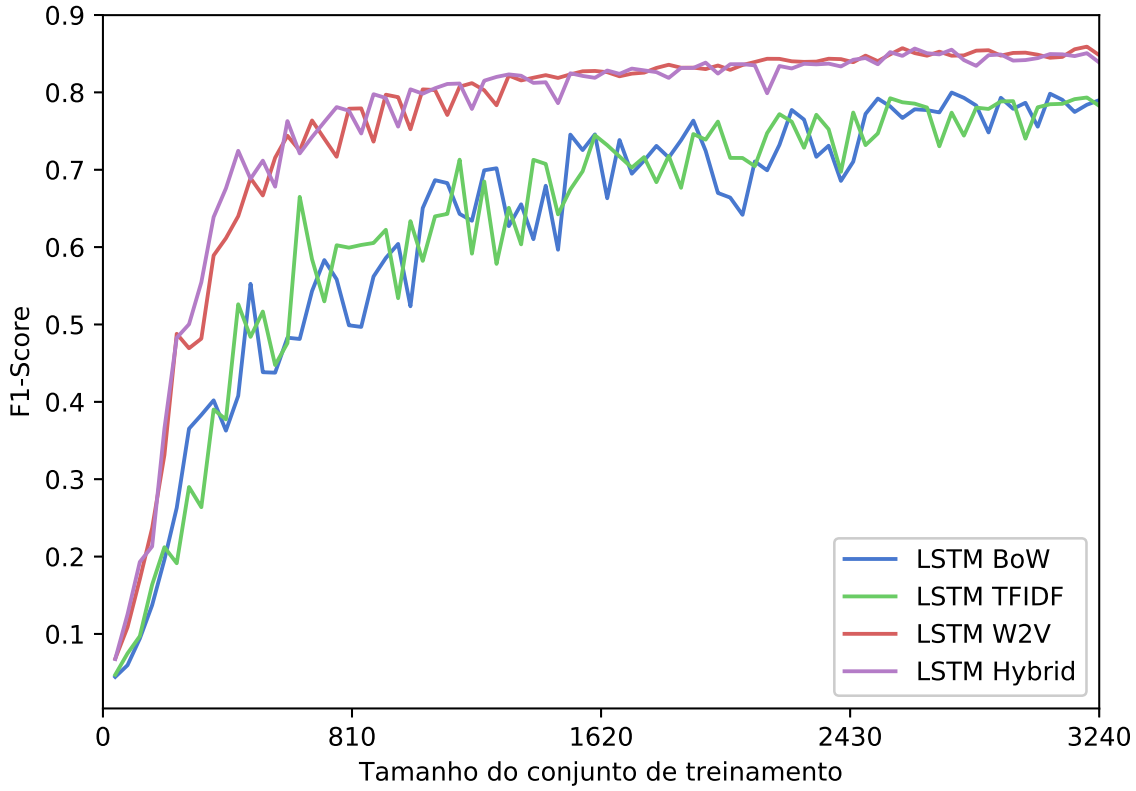
(c) *Word2vec*



(d) Híbrido

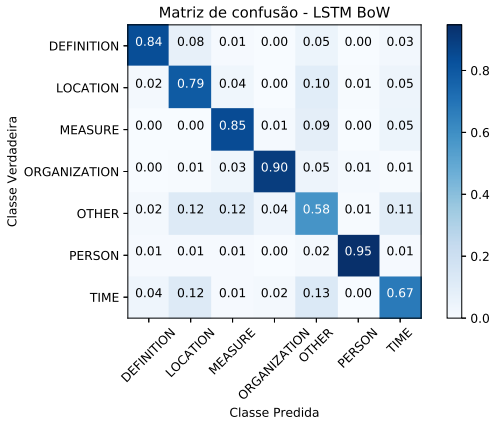
Figura 5.4: Matrizes de Confusão das abordagens utilizando o modelo MLP.
Fonte: O Autor

Figura 5.5: *F1-Score* variando tamanho dos dados de treinamento das abordagens de classificação de pergunta utilizando LSTM.

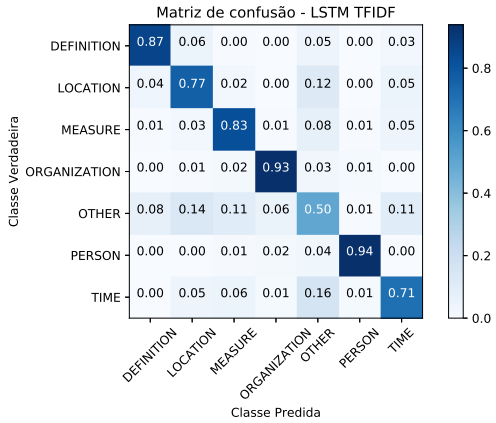


Fonte: O Autor

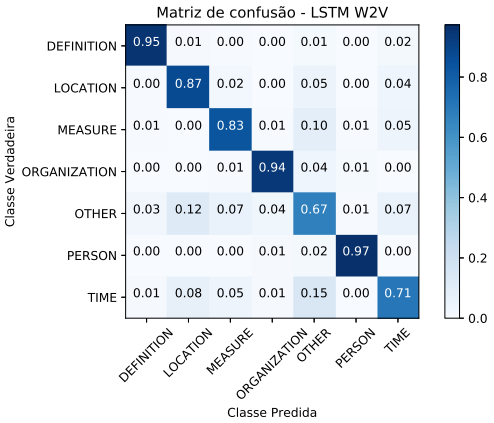
Para o experimento das abordagens com o modelo LSTM, utilizam-se redes compostas por uma camada de entrada, uma camada intermediária LSTM, uma camada intermediária densa com *dropout* de 50 % e uma camada de saída com 7 neurônios. O tamanho da camada de entrada e os neurônios das camadas intermediárias variam conforme a abordagem, da mesma forma que as camadas do modelo MLP. Os modelos são treinados em 10 épocas utilizando uma taxa de aprendizado de 0.001. Os resultados do experimento estão dispostos na Figura 5.5, que apresenta o *F1-Score* pelo tamanho do conjunto de treinamento, na Figura 5.6 que apresenta as matrizes de confusão das abordagens e na Tabela 5.3, que apresenta a precisão, revocação e *F1-Score* para o maior conjunto de treinamento.



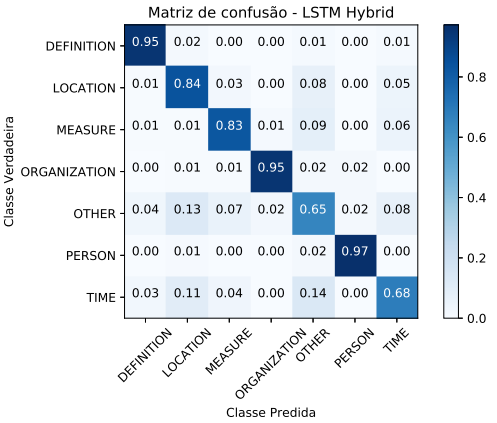
(a) *Bag-of-words*



(b) TFIDF



(c) *Word2vec*



(d) Híbrido

Figura 5.6: Matrizes de Confusão das abordagens utilizando o modelo LSTM.
Fonte: O Autor

Tabela 5.3: Desempenho final das abordagens utilizando LSTM e 3.240 instâncias para treinamento.
Fonte: O Autor

Modelo	**Precisão**	**Revocação**	***F1-Score***
LSTM BoW	0.7911	0.7988	0.7899
LSTM TFIDF	0.7807	0.8036	0.7826
LSTM W2V	**0.8481**	**0.8545**	**0.8481**
LSTM Hybrid	0.8389	0.8431	0.8386

Conforme os resultados dispostos na Figura 5.5, as abordagens híbrida e *Word2vec* atingiram as melhores performances para o modelo LSTM, em que obtiveram resultados em média 10 pontos percentuais melhores do que os das abordagens *Bag-of-words* e TF-IDF. Também, observa-se um maior ruído nos resultados das abordagens baseadas em *Bag-of-words*. As matrizes de confusão dispostas na Figura 5.6, mostram que as classes *DEFINITION*, *ORGANIZATION* e *PERSON* foram melhores classificada, enquanto que as classes *OTHER* e *TIME* obtiveram o pior desempenho. A Tabela 5.3 mostra que a abordagem *Word2vec* obteve performance superior nas métricas de precisão, revocação e *F1-Score* utilizando 3.240 instâncias para o treinamento.

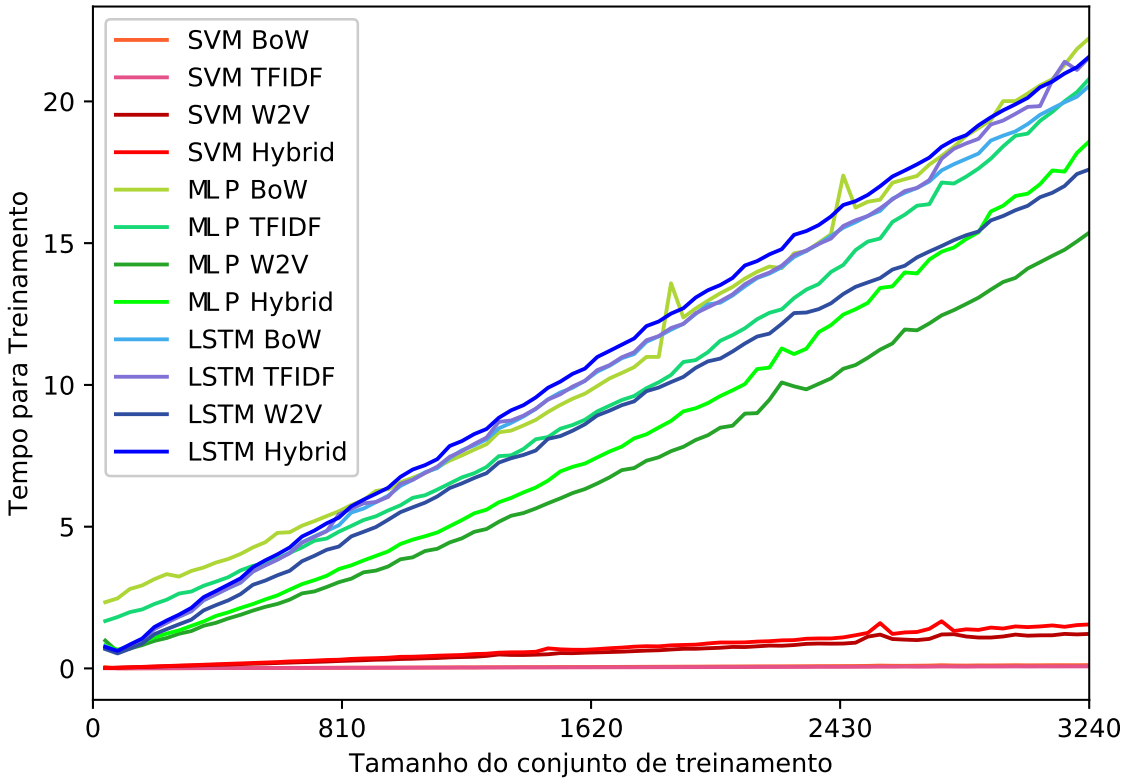
Tabela 5.4: *p-values* resultante do teste estatístico.
Fonte: O Autor

Modelos e Abordagens	***p-value***
SVM Hybrid e SVM BoW	**0.0433**
SVM Hybrid e SVM TFIDF	**0.0468**
SVM Hybrid e SVM W2V	**2.2733e-89**
MLP Hybrid e MLP BoW	0.7637
MLP Hybrid e MLP TFIDF	0.3014
MLP Hybrid e MLP W2V	**3.3664e-71**
LSTM Hybrid e LSTM BoW	**0.0423**
LSTM Hybrid e LSTM TFIDF	**0.0478**
LSTM Hybrid e LSTM W2V	0.8669

A Tabela 5.4 apresenta os resultados da aplicação do teste-t estatístico comparando a abordagem híbrida com as demais abordagens de *baseline* aplicada em todos os modelos supervisionados. Aplicam-se os resultados de *F1-Score* obtidos nos testes variando o tamanho do conjunto de treinamento. Deste modo, uma vez que a janela de variação de conjunto é de 40 instâncias e o tamanho máximo do conjunto de treinamento é de 3.240, cada um dos vetores de resultados para o teste estatístico contém 80 amostras. Os resultados destacados na tabela foram os que atingiram um *p-value* < 0.05 e, desta forma, permitem rejeitar a hipótese nula. Assim, para o modelo SVM, a abordagem híbrida apresenta performance estatisticamente superior às abordagens *Bag-of-words*, TF-IDF e *Word2vec*. Para os modelos MLP, a abordagem híbrida apresenta performance estatisticamente superior à abordagem *Word2vec*. Por fim, para o modelo LSTM, a abordagem híbrida apresenta performance estatisticamente superior às abordagens *Bag-of-words* e *TF-IDF*.

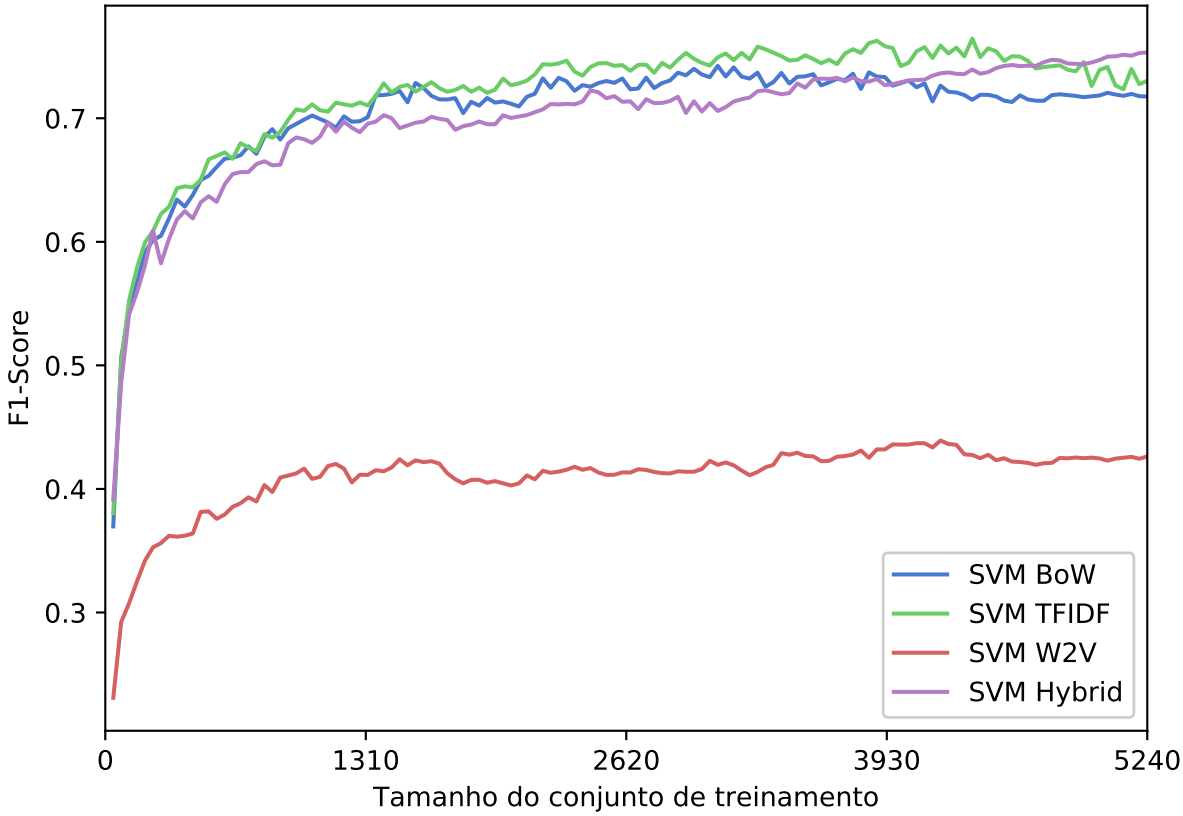
A Figura 5.7 apresenta o tempo de treinamento dos modelos supervisionados para cada uma das abordagens de classificação de pergunta. Obtém-se a média do tempo de treinamento ao utilizar a técnica de validação cruzada para cada modelo treinado. Assim, é possível obter o tempo médio em segundos levados para treinar cada modelo e abordagem variando a quantidade de instâncias de treinamento. Os resultados mostram que, na maioria das vezes, a abordagem híbrida consome mais tempo do que as demais abordagens. Também, o modelo SVM leva menos tempo dos demais modelos, sendo que quando aplicadas as abordagens *Bag-of-words* e TFIDF, o tempo de execução não passa 1 segundo mesmo com 3.240 instâncias. Os modelos LSTM e MLP apresentam mais tempo gasto, sendo que o modelo LSTM na maioria das vezes leva mais tempo para o treinamento.

Figura 5.7: Tempo de execução médio de treinamento variando tamanho dos dados de treinamento para as abordagens de todos os modelos.



Fonte: O Autor

Figura 5.8: Resultados das abordagens utilizando a coleção UIUC e o modelo SVM.



Fonte: O Autor

De modo a testar as abordagens em uma coleção diferente da coleção Chave, testam-se as abordagens com a coleção UIUC. A Figura 5.8 apresenta os resultados obtidos utilizando o modelo SVM. As abordagens TF-IDF, *Bag-of-words* e híbrida obtiveram os melhores resultados, em que apresentam em média performance 30 pontos percentuais melhores do que a abordagem *Word2vec*. Também, observa-se que após 4.200 instâncias de treinamento, a abordagem híbrida apresenta performance superior às abordagens *Bag-of-words* e TF-IDF. A Tabela 5.5 mostra que a abordagem híbrida obteve performance superior nas métricas de precisão, revocação e *F1-Score* utilizando 5.240 instâncias para o treinamento.

Ao comparar os resultados obtidos com a abordagem híbrida, apresentados na Tabela 5.5, com os resultados dos trabalhos de (AOUICHAT; AMEUR; GEUSSOUM, 2018; MOHD; HASHMY, 2018), apresentados na Tabela 3.1, percebe-se que estes obtiveram resultados melhores, já que atingiram *F1-Score* de 93 % e acurácia de 91,9 % respectivamente, enquanto que a abordagem híbrida obteve 75,33 % de *F-Score*. Primeiramente, o trabalho de (AOUICHAT; AMEUR; GEUSSOUM, 2018) emprega a coleção UIUC para a língua Arábica, diferente da coleção utilizada nos resultados gerados nesta dissertação. Além disso, o trabalho aplica dois modelos supervisionados, SVM e

Tabela 5.5: Desempenho das abordagens utilizando a coleção UIUC e 5.240 instâncias para treinamento.

Fonte: O Autor

Modelo	**Precisão**	**Revocação**	***F1-Score***
SVM BoW	0.7069	0.7666	0.7176
SVM TFIDF	0.7237	0.7931	0.7304
SVM W2V	0.4370	0.4712	0.4263
SVM Hybrid	**0.7332**	**0.7932**	**0.7533**

CNN, enquanto que a abordagem híbrida utiliza apenas o modelo SVM. Já o trabalho de (MOHD; HASHMY, 2018), emprega a coleção original UIUC para a língua inglesa. Além disso, os autores aplicam abordagens mais complexas que requerem recursos externos, como o *WordNet*. Por fim, embora os resultados obtidos com a abordagem híbrida sejam inferiores a esses trabalhos, a abordagem proposta tem a vantagem de utilizar conceitos simples, sem a necessidade de múltiplos modelos supervisionados ou recursos externos construídos especificamente para uma determinada língua. Isso faz da abordagem híbrida uma opção de fácil aplicação, independente da coleção de dados ou idioma.

5.1.2 Análise dos resultados

De modo a gerar resultados e análises confiáveis para a tarefa de classificação da pergunta, foram utilizados diferentes procedimentos para gerar esses resultados, como: as abordagens são testadas em diferentes tamanhos de conjunto de treinamento; utilizam-se diferentes modelos de aprendizado de máquina, de forma a verificar o comportamento das abordagens em diferentes modelos; Todos os testes são realizados com validação cruzada com 5 *folds*, que permite variar os dados utilizado para treinamento e teste, além de reduzir ruídos e viés da coleção de dados; Geram-se matrizes de confusão com a média dos valores da validação cruzada, o que possibilita uma análise específica do desempenho de cada classe de pergunta; Por fim, realizam-se testes estatísticos de forma a determinar estatisticamente se a abordagem híbrida proposta tem melhor desempenho do que outra abordagem.

Os resultados da Figura 5.1 e os testes estatísticos da Tabela 5.4 mostram que a abordagem híbrida proposta obteve os melhores resultados para o modelo SVM. Basicamente em todos os tamanhos de conjunto de treinamento a abordagem apresenta desempenho superior, exceto quando o tamanho do conjunto é inferior a 200 instâncias, o desempenho da abordagem é próximo ao desempenho de *Bag-of-words* e TF-IDF. Observa-se que a abordagem híbrida consegue aproveitar os pontos positivos das abordagens TF-IDF e e *Word2vec*. Por exemplo, quando uma dessas abordagens tem uma queda de desempenho, a abordagem híbrida mantém-se com desempenho estável.

Para o modelo MLP, a abordagem híbrida apresentou resultados similares às abordagens *Bag-of-words* e TF-IDF, onde estas três obtiveram os melhores desempenhos utilizando MLP. Diferente do model SVM, a abordagem híbrida só obteve resultados estatisticamente superiores a abordagem *Word2vec*, conforme a Tabela 5.4. Acredita-se que a abordagem híbrida não obteve melhores resultados devido a sensibilidade do modelo MLP, pois, conforme os resultados na Figura 5.3, com o modelo MLP, a abordagem apresentou-se sensível ao baixo desempenho da abordagem *Word2vec*. Essa conclusão deve-se a observação de que a linha de desempenho da abordagem híbrida (*MLP Hybrid*)

tem o mesmo comportamento ruidoso da abordagem *Word2vec* (*MLP W2V*). Além disso, percebe-se que esse ruído é reduzido na abordagem híbrida conforme o tamanho do conjunto de treinamento aumenta.

No modelo LSTM, a abordagem híbrida obteve resultados similares a abordagem *Word2Vec*. Ambas obtiveram os melhores resultados e, conforme a Tabela 5.4, a abordagem híbrida obteve resultados estatisticamente superiores às abordagens *Bag-of-words* e TF-IDF. Porém, conforme a Figura 5.5 e diferente dos modelos SVM e MLP, a abordagem *Word2Vec* obteve os melhores resultados entre os *baselines*. Observa-se em trabalhos na literatura que regularmente aplica-se *word embedding* para classificação de texto quando utilizado o modelo LSTM. Assim, a abordagem *Word2vec* (*LSTM W2V*) sozinha é suficiente para atingir resultados satisfatórios e, portanto, a abordagem TF-IDF não contribui para o modelo híbrido. Deste modo, as abordagens híbrida e *Word2vec* têm desempenhos equivalentes para o modelo LSTM.

Entre as abordagens de *baseline*, percebe-se que *Bag-of-words* e TF-IDF obtiveram desempenhos consideráveis para todos os modelos supervisionados. Já a abordagem *Word2Vec* obteve desempenho inferior, em média 35 pontos percentuais de *F1-Score* a menos, nos modelos SVM e MLP. Por outro lado, a abordagem, juntamente com a híbrida, obtive os melhores desempenhos no modelo LSTM. Deste modo, para a tarefa de classificação de perguntas, a representação de texto através de *Word2vec* apresenta-se a melhor alternativa para o modelo LSTM comparada com *Bag-of-words* e TF-IDF. Já para as abordagens *Bag-of-words* e TF-IDF, apresentam-se as melhores alternativas para os modelos SVM e MLP. Acredita-se que a forma com que os dados são passados aos modelos influenciaram a performance da abordagem *Word2vec*, pois no LSTM, não se realiza a média dos vetores das palavras e, portanto, a representação de uma palavra não é dissolvida em um cálculo de média.

Através das matrizes de confusão, observa-se o desempenho individual de cada classe de pergunta e também a similaridade entre elas. As classes *DEFINITION* e *MEASURE* foram melhores classificadas em todos os modelos, o que pode indicar que estas apresentam termos e representações semânticas mais distintas do que as demais classes, o que facilita sua classificação. Já a classe *OTHER* apresentou o pior desempenho. Como a mesma representa as classes de perguntas mais abstratas, acredita-se que estas contém características semelhantes com as demais classes, o que dificultou sua classificação. Também, constata-se que as classes *PERSON* e *ORGANIZATION* são confundidas entre si. Acredita-se que o motivo deve-se a que perguntas dessas classes têm estruturas muito semelhantes, em que muitas vezes precisa-se do conhecimento externo de mundo para classificá-las. Por exemplo, as perguntas "Quem ganhou a copa do mundo de 1994?"e "Quem descobriu o Brasil?"têm estruturas muito parecidas, porém, são de classes distintas, neste caso as classes *ORGANIZATION* e *PERSON* respectivamente.

Entre os modelos supervisionados de aprendizado de máquina, o SVM e o MLP obtiveram performances similares entre as melhores abordagens aplicadas. Ambos os modelos, conforme as Tabelas 5.1 e 5.2, utilizando 3.240 instâncias atingiram o *F1-Score* de 0.8073 e 0.8064 respectivamente. Além disso, conforme as Figuras 5.1 e 5.3 percebe-se que os modelos começam a estabilizar seu desempenho aproximadamente a partir das 800 instâncias de treinamento. Já o modelo LSTM apresentou os melhores resultados, atingindo *F1-Score* de 0.8481 com 3.240 instâncias de treinamento. Também, a performance do modelo estabiliza aproximadamente a partir das 800 instâncias de treinamento. Por fim, percebe-se que no modelo LSTM, a diferença entre a performance das aborda-

gens baseadas em *word embedding* (*Word2vec* e híbrida) e as puramente de vetores de frequência de vocabulários (*Bag-of-words* e TF-IDF) diminui ao passo que o tamanho do conjunto de treinamento aumenta.

Os testes com a coleção UIUC mostraram o desempenho das abordagens em uma coleção de dados diferente. Os resultados mostram que as abordagens *Bag-of-words* e TF-IDF obtiveram o melhor desempenho com até 4.000 instâncias de treinamento, enquanto que a abordagem híbrida presentou o melhor desempenho quando o conjunto de treinamento era maior. Também, ao utilizar 5.240 instâncias de treinamento com o modelo SVM, a melhor abordagem atingiu 0.7533 de *F1-Score*, conforme a Tabela 5.5. Comparando com a coleção Chave, a UIUC demonstrou-se uma coleção mais desafiadora na tarefa de classificação de perguntas, uma vez que mesmo utilizando 6 classes de pergunta (uma a menos do que a coleção Chave) e 5.240 instâncias de treinamento, a abordagem híbrida obteve *F1-Score* 0.054 a menos na coleção UIUC.

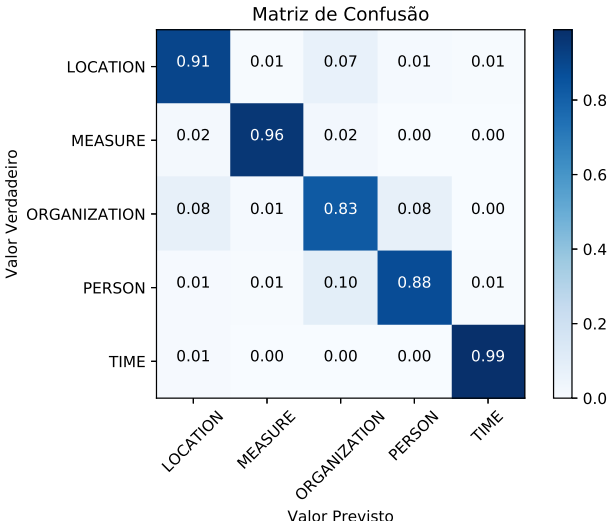
Através dos testes de classificação da pergunta, a abordagem híbrida mostrou-se promissora para a tarefa, uma vez que em todos os testes a abordagem apresentou os melhores resultados ou resultados próximos e equivalentes aos das melhores abordagens. Desta forma, a abordagem apresenta a característica de generalização, uma vez que essa pode ser empregada nos diferentes modelos testados e apresentar resultados consideráveis. Para o modelo SVM, a abordagem híbrida mostrou-se a melhor opção, já que atingiu o melhor desempenho em todos os tamanhos de instâncias de treinamento para coleção Chave, e o melhor desempenho para treinamentos com mais de 4.000 instâncias para a coleção UIUC. Por fim, conforme a Figura 5.7, o tempo de execução para o treinamento da abordagem híbrida mostrou-se um pouco acima das demais abordagens, porém, essa diferença pode não ser significativa levando em consideração que a abordagem poderá agregar ganho de performance na tarefa de classificação de perguntas.

5.2 Sistema de QA

Esta seção destina-se a apresentar os resultados obtidos com o sistema de QA desenvolvido para língua portuguesa. O objetivo no desenvolvido desse sistema é testar as diferentes abordagens de representação de texto para a etapa de classificação de pergunta. Assim, verificar o impacto da etapa na performance geral do sistema de QA. Deste modo, esta seção é dividida nas subseções que apresentam os resultados de cada módulo do sistema e suas análises. Por fim, a subseção 5.2.3 dispõe o desempenho final do sistema com as diferentes abordagens de classificação de perguntas.

Como a coleção Chave compõem-se de perguntas com e sem respostas, as que não continham ao menos uma resposta foram utilizadas como dado de treinamento para o modelo supervisionado, enquanto que as que continham foram utilizadas para testes, uma vez que as respostas dessas perguntas permitem testar a capacidade do sistema de responder essas de forma correta. Além disso, como o sistema desenvolvido responde apenas perguntas do tipo factoide, utilizam-se apenas as perguntas das classes *LOCATION*, *MEASURE*, *ORGANIZATION*, *PERSON* e *TIME*, totalizando 2.435 perguntas. Uma vez que as perguntas contêm dados anotados, para as demais etapas do sistema, foram utilizadas todas as perguntas para testes ou, dependendo da etapa, apenas as que continham resposta.

Figura 5.9: Matriz de confusão do modelo híbrido utilizando os dados dispostos no experimento com o sistema de QA.



Fonte: O Autor

5.2.1 Processamento da Pergunta

No módulo de Processamento da Pergunta, obtiveram-se os resultados da etapa de classificação da pergunta. Já para a etapa de formulação de *query*, pode-se obter indícios de seu desempenho através dos resultados do módulo de Recuperação de Informações. Deste modo, para gerar os resultados da etapa de classificação de pergunta, utilizam-se as perguntas da coleção Chave que não contém resposta para o treinamento e as demais para testes. Assim, totaliza-se 1.779 instâncias para treinamento e 656 instâncias para teste. Utiliza-se o modelo SVM nesta etapa, uma vez que esse demonstrou o melhor desempenho da abordagem híbrida em relação às demais abordagens.

Os resultados da etapa de classificação da pergunta mostra que ao utilizar a abordagem híbrida, obtém-se precisão de 0.899, revocação de 0.913 e *F1-Score* de 0.904. A Figura 5.9 apresenta a matriz de confusão resultando do experimento. Percebe-se que as classes *TIME* e *MEASURE* obtiveram desempenho acima de 0.95 em verdadeiros positivos enquanto que as classes *ORGANIZATION* e *PERSON* obtiveram desempenho abaixo de 0.90. Observa-se que nos testes da seção 5.1 de classificação da pergunta, a abordagem híbrida com o modelo SVM obteve *F1-Score* de 0.8073, 0.0967 inferior aos testes no sistema de QA, porém, nestes, ao invés de utilizar todas as 7 classes, utilizam-se apenas 5 classes, o que justifica o ganho de performance.

Os resultados mostram que a abordagem híbrida aplicada a um modelo SVM atinge *F1-score* acima dos 90 pontos percentuais na tarefa de classificação de perguntas na Coleção Chave utilizando apenas as perguntas do tipo factóide. Enquanto que o modelo teve facilidade na classificação de perguntas da classe *TIME*, percebe-se novamente que perguntas do tipo *PERSON* e *ORGANIZATION* foram confundidas pelo classificador, já que *PERSON* obteve falso positivo de 0.10 em relação a *ORGANIZATION* e *ORGANIZATION* obteve falso positivo de 0.08 em relação a *PERSON*. Também, as classes *LOCATION* e *ORGANIZATION* apresentar níveis consideráveis de confusão. Acredita-se

que ambas possam confundir quando uma resposta é da classe *LOCATION* ou *ORGANIZATION*. Por exemplo, "Alemanha"pode ser uma organização ou uma localidade.

5.2.2 Recuperação de Informações

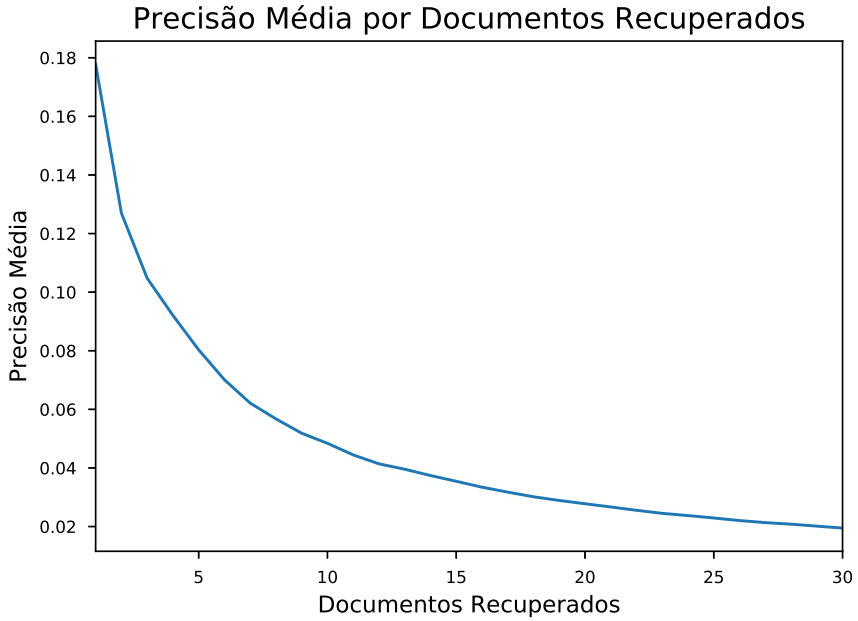
No módulo de Recuperação de Informações, obtiveram-se os resultados da etapa de recuperação de documentos utilizando os dados da coleção Chave, uma vez que essa disponibiliza para cada resposta o ID de identificação de documentos relevantes. Como a coleção não dispõe da localização de sentenças relevantes nos documentos, obtiveram-se os resultados da etapa de recuperação de sentenças verificando se uma determinada sentença contém em seu corpo uma das possíveis respostas de uma determinada pergunta em questão. Por fim, para a etapa de REM, obtiveram-se os resultados utilizando a coleção Harem, em que essa disponibiliza textos com entidades anotadas, o que permite treinar um modelo supervisionado para identificar e classificar as entidades de uma determinada sentença.

A Figura 5.10 apresenta os resultados da técnica utilizada na etapa de recuperação de documentos. O gráfico mostra a relação entre a precisão de relevância pela posição no ranque de documentos recuperados. Obtém-se essa relação através da média de precisão das consultas realizas com todas as perguntas da coleção Chave que contenham ao menos uma resposta. Desta forma, a curva apresentada mostra que em média 18 % dos documentos na primeira posição do ranque são relevantes. Também, observa-se que a técnica obtém melhor performance até a décima posição do ranque, uma vez que a partir dessa posição, a precisão média se estabiliza abaixo dos 5 pontos percentuais. Por fim, utilizando um máximo de 30 documentos recuperados, a técnica de recuperação de documentos aplicada obteve precisão média igual a 0.0347, revocação média igual a 0.7023 e *F-Score* médio igual 0.0622.

Apesar da coleção Chave disponibilizar a identificação de documentos relevantes para as respostas de suas perguntas, em média, encontram-se no máximo três documentos anotados por pergunta. Também, através de observações manuais e empíricas, observa-se que muitos dos documentos recuperados e que não estão anotados como relevantes contém as informações necessárias para a resposta. Portanto, esses documentos podem ser classificados como relevantes. A falta da anotação desses deve-se a grande quantidade de trabalho manual e exaustivo para os criadores da coleção, uma vez que é necessário verificar para cada pergunta todos os documentos da coleção (mais de 200 mil), quais são relevantes para ela. Assim, os valores resultantes do teste de recuperação de documentos não refletem a real performance da técnica aplicada, e sim, colabora com uma estimativa se aos poucos documentos relevantes anotados estão sendo recuperados. Portanto, neste caso a métrica mais importante a ser considerada é a revocação. Por fim, os resultados desse teste permitiu concluir que a técnica obteve desempenho mais significativo nos dez primeiros documentos recuperados, conforme a curva da Figura 5.10.

Na Figura 5.11, apresentam-se os resultados da técnica utilizada na etapa de recuperação de sentenças. O gráfico mostra a relação entre a precisão de relevância pela posição no ranque de sentenças recuperadas. Obtém-se essa relação através da média de precisão de relevâncias das sentenças recuperadas. Desta forma, a curva apresentada mostra que em média 20 % das sentenças na primeira

Figura 5.10: Precisão média por documentos recuperados no experimento com o sistema de QA.



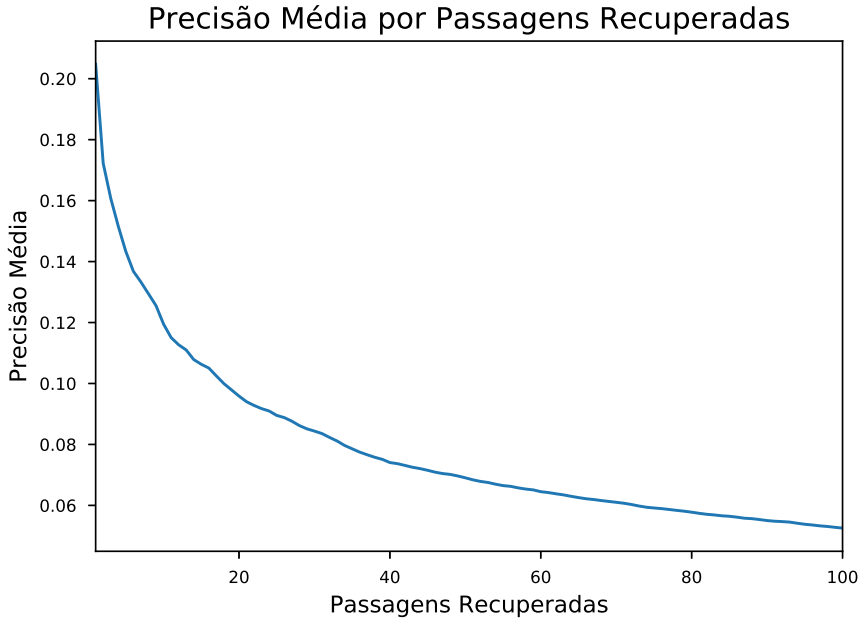
Fonte: O Autor

posição do ranque são relevantes. Também observa-se que a técnica obtém melhor performance de precisão até a quadragésima posição do ranque, uma vez que a partir dessa posição, a precisão média se estabiliza a baixo dos 8 pontos percentuais. Por fim, verificou-se que para 65,24 % das perguntas da coleção, foi possível recuperar ao menos uma sentença relevante, sendo que a precisão média de relevância por sentença recuperada é de 3,65 %.

Uma vez que a coleção Chave não realiza a marcação de sentenças de documentos, como também não dispõe a informação de quais sentenças são relevantes, não é possível calcular a revocação e o *F1-Score* na etapa de recuperação de sentenças. Deste modo, os resultados da etapa oferecem a precisão média de relevância por sentenças recuperadas e a porcentagem de perguntas que obtiveram ao menos uma sentença relevante recuperada. Assim, conforme a curva da Figura 5.11, a técnica para recuperar e ranquear sentenças do sistema obteve desempenho mais significativo nas 40 primeiras sentenças recuperadas.

A Tabela 5.6 apresenta os resultados obtidos nos testes da etapa de REM. Esses resultados foram obtidos utilizando a coleção Harem juntamente com o modelo supervisionado CRF e aplicando a técnica de validação cruzada com 8 *folds*. A grande maioria das instâncias da coleção são da classe 'O' (*outside*), que representam palavras que não são entidades mencionadas. Assim, os cálculos das métricas ignoraram as instâncias dessas classes, uma vez que se estas são consideradas, a performance do modelo atinge resultados superiores, que podem não representar seu real desempenho. As linhas da tabela em destaque representam as classes de entidades que foram utilizadas pelo de QA. Também, divide-se cada classe em duas, de modo com que as que contém '-B' (*begin*) representam início de entidade e as que contém '-I' (*inside*) representam os restantes dos termos de uma entidade mencionada composta.

Figura 5.11: Precisão média por passagens recuperadas no experimento com o sistema de QA.



Fonte: O Autor

Os resultados da etapa de REM mostram que dentre as classes utilizadas pelo sistema de QA, as classes *VALOR-I*, *TEMPO-I* e *LOCAL-B* obtiveram os melhores desempenhos, uma vez que estas atingiram *F1-Score* superior a 0.75. Já as classes *ORGANIZACAO-I*, *LOCAL-I* e *PESSOA-I* obtiveram os piores desempenhos, sendo que a classe *'ORGANIZACAO-I'* obteve *F1-Score* de 0.582. Por fim, no geral, o modelo obteve precisão de 0.673, revocação de 0.622 e *F1-Score* de 0.63. A função da etapa de classificação da pergunta tem ligação direta com funções da etapa de REM, uma vez que a primeira fica responsável por identificar a classe de pergunta e a segunda fica responsável por encontrar entidades mencionadas no texto dessa mesma classe. Logo, percebe-se que a etapa de REM obteve desempenho inferior ao da etapa de classificação de perguntas, que atingiu *F1-Score* de 0.904. Deste modo, a etapa de REM impede que o desempenho geral do sistema de QA atinja resultados semelhantes aos da etapa de classificação de perguntas.

5.2.3 Processamento da Resposta

Como o módulo de Processamento da Resposta é o último estágio da arquitetura de uma sistema de QA, este subcapítulo apresenta o desempenho final do sistema desenvolvido. Deste modo, obtiveram-se os resultados referente a etapa de ranqueamento de respostas candidatas e o desempenho geral do sistema de QA variando o tamanho de dados de treinamento para o modelo de classificação da pergunta. Da mesma forma que as etapas anteriores, utiliza-se da coleção Chave as perguntas sem respostas como instâncias de treinamento enquanto que as que contém ao menos uma resposta são utilizadas como instâncias de teste.

Tabela 5.6: Performance do modelo de REM para o o sistema de QA
Fonte: O Autor

Classe	**Precisão**	**Revocação**	**F1-Score**
VALOR-B	0.766	0.694	0.728
VALOR-I	0.951	0.661	0.780
OBJECTO-B	0.000	0.000	0.000
OBRA-B	0.308	0.276	0.291
OBRA-I	0.115	0.222	0.151
ABSTRACCAO-B	0.507	0.514	0.510
ABSTRACCAO-I	0.409	0.466	0.435
ACONTECIMENTO-B	0.333	0.138	0.195
ACONTECIMENTO-I	0.263	0.200	0.227
TEMPO-B	0.800	0.677	0.733
TEMPO-I	0.864	0.826	0.844
PESSOA-B	0.724	0.698	0.710
PESSOA-I	0.732	0.670	0.700
LOCAL-B	0.810	0.757	0.783
LOCAL-I	0.748	0.671	0.707
COISA-B	0.800	0.148	0.250
COISA-I	0.800	0.133	0.229
ORGANIZACAO-B	0.722	0.743	0.732
ORGANIZACAO-I	0.464	0.782	0.582
OUTRO-B	1.000	0.200	0.333
OUTRO-I	0.000	0.000	0.000
Média	**0.673**	**0.622**	**0.630**

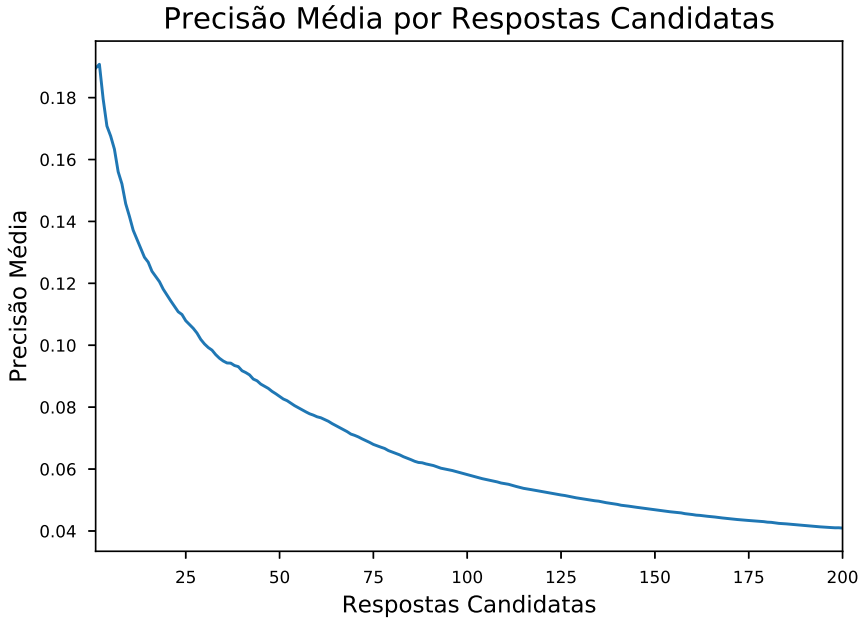
A Figura 5.12 apresenta os resultados obtidos com a técnica utilizada na etapa de extração e ranqueamento de respostas candidatas. O gráfico mostra a relação entre precisão média de relevância pela posição no ranque de respostas candidatas extraídas. Deste modo, a curva apresentada mostra que aproximadamente em média 20 % das respostas candidatas na primeira posição do ranque estão corretas. Também, observa-se que a técnica obtém melhor performance de precisão até a posição 100 do ranque, uma vez que a partir dessa posição, a precisão média se estabiliza abaixo dos 6 pontos percentuais. Por fim, verifica-se que para 71.646 % das perguntas da coleção, foi possível extrair ao menos uma resposta candidata correta.

A verificação se uma resposta retornada pelo sistema de QA está correta é feita de forma automática. Deste modo, para cada pergunta, utilizam-se as respostas corretas na coleção Chave para cruzá-las com as respostas geradas pelo sistema. Assim, as respostas não são verificadas manualmente por humanos. Também, como a verificação é feita de forma automática e direta, respostas corretas retornadas pelos sistemas, expressadas de maneiras diferentes da forma expressa na coleção, serão pontuadas como incorretas. Assim, os resultados podem não refletir a real capacidade do sistema de QA em responder de forma correta as perguntas, e sim, disponibilizar uma estimativa de desempenho do mesmo. Essa estimativa, utilizada nos resultados da Figura 5.13, é suficiente para verificar se o desempenho geral do sistema de QA tem o mesmo comportamento dos resultados de classificação de perguntas utilizando diferentes abordagens.

Através dos resultados da Figura 5.12 estima-se que o sistema de QA apresenta aproximadamente 20 % de precisão em sua capacidade de responder perguntas de forma correta. Embora que para 71.646 % das perguntas da coleção Chave a resposta correta é extraída, a técnica utilizada para ranquear as respostas candidatas conseguem em média 20 % das vezes colocar uma resposta correta na primeira posição do ranque. Observa-se esse mesmo desempenho nas etapas anteriores no módulo de Recuperação de Informações, em que, conforme as Figuras 5.10 e 5.11, em média 20 % das vezes, um documento relevante, uma sentença relevante ou uma resposta correta estão na primeira posição de seus respectivos ranques. Deste modo, embora estas etapas apresentem desempenhos consideráveis na sua revocação, apresentam baixa precisão. Acredita-se que o baixo desempenho na precisão deve-se às abordagens simples utilizadas no ranqueamento de sentenças e respostas candidatas, em que se aplicam pontuações baseadas apenas em classes de entidades mencionadas, termos das perguntas e frequência de respostas iguais.

Para obter os resultados apresentados na Figura 5.13, avalia-se a precisão geral do sistema de QA variando o tamanho do conjunto de treinamento, aplicando validação cruzada com 5 *folds* e variando a abordagem de classificação de perguntas. O intuito do teste é de verificar se os resultados de classificação de perguntas apresenta o mesmo comportamento que o desempenho geral de um sistema de QA utilizando as diferentes abordagens, uma vez que se assume que a etapa de classificação da pergunta tem influência considerável no desempenho final do sistema. Utiliza-se o modelo SVM para gerar os resultados, uma vez que esse apresentou diferença mais significativa entre a abordagem híbrida e as demais de *baseline*. Observa-se que a performance do sistema de QA aumenta conforme o tamanho do conjunto de treinamento. Também, utilizando a abordagem híbrida com 1680 instâncias de treinamento, a precisão do sistema é de aproximadamente 17 %, valor próximo a precisão média da primeira posição do ranque de respostas candidatas apresentada na Figura 5.12.

Figura 5.12: Precisão média por respostas candidatas no experimento com o sistema de QA.



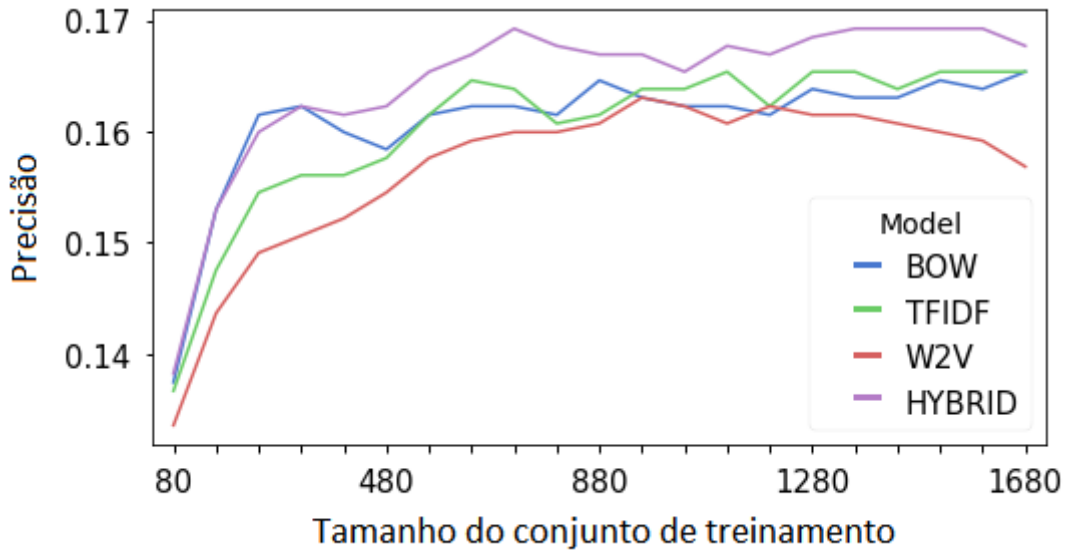
Fonte: O Autor

Os resultados da Figura 5.13 apresentam comportamento semelhante aos resultados de classificação da pergunta, apresentados na Figura 5.1, onde se utilizou as mesmas abordagens e o mesmo modelo supervisionado. Observa-se que em ambos os gráficos, a abordagem híbrida apresenta melhor desempenho, seguida pelas abordagens *Bag-of-words* e TF-IDF, e por fim, *Word2vec* apresentando o pior desempenho. Assim, os resultados apresentam evidências de que a abordagem de classificação de perguntas tem papel significativo no desempenho geral de um sistema de QA. Também, comparando os dois gráficos de desempenho, observa-se que o gráfico de resultados do sistema de QA apresenta comportamento mais ruidoso do que o gráfico de resultados de classificação de perguntas. Isso deve-se aos erros do sistema de QA que são causados por etapas que não são as de classificação de pergunta, logo erros causados pelas demais etapas podem causar maior variação no resultado final do sistema.

5.3 Consideração sobre os resultados

Esta seção destina-se a apresentar as considerações gerais sobre os resultados obtidos nos experimentos de classificação de perguntas e do sistema de QA desenvolvido. O objetivo dos testes de classificação de perguntas foi testar as diferentes abordagens para representação de perguntas de texto em diferentes modelos supervisionados de aprendizado de máquina. Já os testes com o sistema de QA desenvolvido, visaram verificar o impacto da etapa de classificação de perguntas em um sistema completo de QA, de modo a verificar se os resultados obtidos com as diferentes abordagens têm o mesmo comportamento de performance dos testes de classificação de perguntas. De modo a contribuir com a confiabilidade dos resultados, os experimentos foram realizados variando o tamanho

Figura 5.13: Precisão do sistema de QA variando o tamanho de dados de treinamento para classificação de perguntas utilizando SVM.



Fonte: O Autor

do conjunto de treinamento, utilizando validação cruzada, aplicando testes estatísticos e aplicando diferentes abordagens de avaliação.

No experimento de classificação de perguntas, utilizaram-se os modelos supervisionados SVM, MLP e LSTM. Para ambos, realizaram-se testes aplicando a abordagem híbrida proposta e as demais abordagens de *baseline*. Os resultados mostram que a abordagem híbrida obteve resultados superiores, uma vez que em todos os testes, essa obteve os resultados estatisticamente melhores ou próximos a melhor abordagem, o que também indica que a abordagem tem a característica de generalização. Ou seja, isso evidencia que a combinação de características lexicais, juntamento com característica semânticas têm performance consideravelmente robusta em diferentes modelos supervisionados de aprendizado de máquina para a tarefa de classificação de perguntas em português. Também, destacou-se o modelo LSTM, que obteve os melhore resultados, atingindo 83,86 pontos de *F1-Score* com a abordagem híbrida, 4,08 pontos percentuais a mais do que o modelo SVM, segundo colocado.

Conforme a análise dos resultados, a abordagem TF-IDF consegue representar padrões de palavras importantes para a classificação de uma pergunta, como os termos "Quando", "Onde", "Por que"e "Quem". Esses termos, na maioria das vezes, podem determinar sua classe, porém, essa abordagem não apresenta uma boa representação semântica da sentença, uma vez que duas classes de perguntas podem conter um desses mesmo termos, variando apenas a semântica dos outros termos. Diferente dessa, a abordagem *Word2vec* consegue representar uma pergunta através de vetores de *word embedding*, que determina a posição de uma palavra em um espaço semântico. Porém, ao realizar a média desses vetores, de modo a representar toda a pergunta em um vetor de tamanho fixo, se perde a representação de palavras importantes para a classificação. Assim, conforme os resultados obtidos, a combinação dessas duas características resultam em uma abordagem satisfatória para a tarefa, já que a mesma consegue unir as vantagens da abordagem TF-IDF e *Word2Vec*.

Os resultados dispostos através de matrizes de confusão evidenciam que algumas classes de

perguntas se confundem no momento da classificação. Destaca-se a confusão entre as classes *PERSON* e *ORGANIZATION*, em que se acredita que as estruturas lexicais de perguntas dessas classes são muitas vezes idênticas, diferenciando entre si somente através de características semânticas ou do conhecimento externo de mundo. Por exemplo, as perguntas "Quem comanda o poder judiciário?"e "Quem comanda a seleção brasileira?", são de classes diferentes, porém apresentam estruturas lexicais muito parecidas. Também, percebe-se confusão entre as classes *LOCATION* e *ORGANIZATION*, em que se acredita que o motivo está relacionado com termos homônimos, de modo que dependendo do contexto o termo terá diferente significado. Por exemplo, o estado brasileiro "Goiás"e o clube de futebol "Goiás", podem ser respostas para as seguintes perguntas "Em que estado fica situado o distrito federal?"e "Em que time jogou o jogador Lucinho?".

No experimento com o sistema de QA desenvolvido foram testados as técnicas aplicadas nas diferentes etapas de sua arquitetura. O desempenho geral do sistema de QA foi de aproximadamente 20 pontos percentuais de precisão na tarefa de retornar uma resposta correta. É uma performance abaixo do desempenho do melhores participantes do CLEF utilizando a coleção Chave, como o Priberam (AMARAL et al., 2007), que atingiram performance próximas aos 60 pontos percentuais de precisão.

Além da etapa de classificação de perguntas, outras etapas obtiveram resultados satisfatórios em suas tarefas no sistema de QA. Na etapa de recuperação de documentos, embora os resultados mostrem uma estimativa de sua performance, através de observações empíricas de documentos recuperados, a ferramenta *Solr* mostrou-se promissora na tarefa. Também, o modelo supervisionado CRF, juntamente com a coleção Harem, apresentaram resultados consideráveis para a tarefa de REM, deste modo, o modelo treinado está disponível online [1].

Os resultados do sistema de QA, variando a abordagem de classificação de perguntas e o tamanho do conjunto de treinamento, mostram que o desempenho das abordagens tiveram comportamento semelhante aos resultados do experimento de classificação de perguntas utilizando o modelo SVM. Observa-se que em ambos os experimentos a abordagem híbrida obteve os melhores resultados, em seguida as abordagens *Bag-of-words* e TF-IDF obtiveram os segundos melhores resultados e, por fim, a abordagem *Word2vec* obteve a pior performance. Esses resultados mostram evidências da importância da etapa de classificação de perguntas, uma vez que o desempenho geral do sistema aumenta conforme a capacidade de predição da classe de pergunta é melhorada.

Através de análise empírica do fluxo de processamento de dados, observaram-se erros comuns no sistema de QA. Com estas observações, sugerem-se possíveis melhorias para o sistema. Os principais erros estão listados abaixo:

- **Pergunta:** Em que cidade se encontra a prisão de San Vittore?
  Resposta correta: Milão
  Resposta do sistema: Itália
  O sistema não sabe qual a diferença entre país e cidade. Logo, retorna "Itália"por ser o termo mais frequente.

- **Pergunta:** Com quem se casou Michael Jackson?
  Resposta correta: Irmã de Elvis Presley

[1] https://github.com/eduardogc8/qa-chave

Resposta do sistema: Elvis Presley
O sistema só retorna "Elvis Presley"pois a etapa de REM não inclui "Irmã de"na entidade retornada como resposta.

- **Pergunta:** Quem foi o primeiro presidente dos Estados Unidos?
  Resposta correta: George Washington
  Resposta do sistema: Bill Clinton
  O sistema não encontrou documento com os termos "primeiro presidente dos Estados Unidos". Porém, o documento relevante contém o termo "primeiro presidente norte-americano", deste modo, o sistema não conseguiu identificar que "presidente dos Estados Unidos"e "presidente norte-americano"tem o mesmo significado.

Conforme os erros comuns cometidos que foram listados acima, percebe-se que esses estão relacionados ao problema de falta de conhecimento de mundo. Normalmente, esse problema não é tão frequente em sistemas de QA de domínio específico, uma vez que nestes a base de dados do sistema está estruturada, como uma ontologia. Deste modo, uma das alternativas de melhorias para o sistema seria a aplicação de ontologias, o que atribuiria ao sistema de QA conhecimento externo de mundo. Possivelmente, estas ontologias seriam de conhecimento geral, uma vez que é muito custoso representar um espectro de conhecimento amplo de forma específica. Também, outra solução seria utilização de subclasses de perguntas e de entidades mencionadas. Isso permitiria que o sistema tivesse mais precisão no tipo de entidade a ser buscada em sua base de dados.

Dado os resultados, suas análises e a pergunta de pesquisa, "A união de características lexicais e semânticas do texto de perguntas, aplicada em modelos supervisionados, são suficientes para atingir resultados satisfatórios na tarefa de classificação de perguntas em sistemas de QA, variando o tamanho do conjunto de treinamento?", conclui-se que a abordagem híbrida proposta consegue atingir resultados satisfatórios na tarefa para a língua portuguesa, em diferentes tamanhos de conjunto de treinamento, para os modelos supervisionados SVM, MLP e LSTM.

6 CONSIDERAÇÕES FINAIS

Este trabalho de pequisa propôs uma abordagem híbrida para a etapa de classificação de perguntas de sistemas de QA. A abordagem consiste em unir características lexicais e semânticas do texto da pergunta através de vetores de representação TF-IDF e *Word2vec*. Ambas características não requerem recursos complexos de aquisição, facilitando a aplicação da abordagem em línguas não inglesa. Deste modo, a abordagem proposta busca ser robusta com diferentes tamanhos de conjuntos de treinamento.

De modo a comparar os resultados obtidos com a abordagem híbrida, foram utilizadas as técnicas de representação de texto *Bag-of-words*, TF-IDF e *Word2vec* como abordagens *baselines*. A coleção Chave foi utilizada para os testes, em que todas as abordagens foram testadas nos modelos supervisionados SVM, MLP e LSTM, variando o tamanho do conjunto de treinamento. Por fim, de modo a testar o impacto de cada abordagem em um sistema completo de QA, foi desenvolvido um sistema para a língua portuguesa capaz de responder perguntas do tipo factoide. Assim, foram realizados diversos testes com o sistema desenvolvido, variando o tamanho do conjunto de treinamento.

Os resultados apresentados neste trabalho mostram que a abordagem híbrida proposta obteve resultados promissores, uma vez que em todos os testes, essa obteve os resultados estatisticamente melhores ou próximos à melhor abordagem. Isso indica que a abordagem tem a característica de generalização, evidenciando que a combinação de características lexicais e semânticas têm performance consideravelmente robusta em diferentes modelos supervisionados de aprendizado de máquina para a tarefa de classificação de perguntas em português.

Os testes com o sistema de QA desenvolvido mostram que o desempenho das abordagens tiveram comportamento semelhante aos resultados do experimento de classificação de perguntas utilizando o modelo SVM. Observa-se que em ambos os experimentos a abordagem híbrida obteve os melhores resultados, em seguida as abordagens *Bag-of-words* e TF-IDF obtiveram os segundos melhores resultados e, por fim, a abordagem *Word2vec* obteve a pior performance. Esses resultados mostram evidências da importância da etapa de classificação de perguntas, uma vez que o desempenho geral do sistema aumenta conforme a capacidade de predição da classe de pergunta é melhorada.

Como trabalhos futuros, sugerem-se as seguintes atividades:

- Os testes realizados utilizaram as coleções para a língua portuguesa Chave e UIUC traduzida. Para ambas, não foi possível comparar os resultados de classificação de perguntas apresentados neste trabalho com outras abordagens da literatura. Deste modo, sugere-se a realização de testes dessas abordagens com outras coleções de língua não-inglesa, de modo a comparar os resultados com demais abordagens da literatura.

- Embora que os resultados com o sistema de QA apresentaram comportamento semelhante aos experimentos de classificação de perguntas, o desempenho geral do sistema foi de aproximadamente 20 % de precisão. Além disso, através da análise dos resultados das etapas do sistema, observa-se que o sistema obteve os piores desempenhos na etapa de recuperação de sentenças e no módulo de Processamento da Pergunta. Assim, sugerem-se melhorias nesta etapa, como também, a utilização de técnicas mais sofisticadas no módulo final, como redes neurais profundas para compreensão de leitura.

6.1 Publicação

O presente trabalho de pesquisa foi submetido e publicado na *International Conference on Computational Processing of the Portuguese Language* realizado em Gramado-RS, Brasil (CORTES; WOLOSZYN; BARONE, 2018).

REFERÊNCIAS

ABBAS, F. et al. Wikiqa—a question answering system on wikipedia using freebase, dbpedia and infobox. In: IEEE. **Innovative Computing Technology (INTECH), 2016 Sixth International Conference on**. [S.l.], 2016. p. 185–193.

AMARAL, C. et al. Priberam's question answering system in qa@ clef 2007. In: SPRINGER. **Workshop of the Cross-Language Evaluation Forum for European Languages**. [S.l.], 2007. p. 364–371.

AMARAL, C. et al. Priberam's question answering system in qa@ clef 2008. In: SPRINGER. **Workshop of the Cross-Language Evaluation Forum for European Languages**. [S.l.], 2008. p. 337–344.

AOUICHAT, A.; AMEUR, M. S. H.; GEUSSOUM, A. Arabic question classification using support vector machines and convolutional neural networks. In: SPRINGER. **International Conference on Applications of Natural Language to Information Systems**. [S.l.], 2018. p. 113–125.

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

BAO, J. et al. Knowledge-based question answering as machine translation. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [S.l.: s.n.], 2014. v. 1, p. 967–976.

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

CAO, Y. et al. Askhermes: An online question answering system for complex clinical questions. **Journal of biomedical informatics**, Elsevier, v. 44, n. 2, p. 277–288, 2011.

CHIU, J. P.; NICHOLS, E. Named entity recognition with bidirectional lstm-cnns. **Transactions of the Association for Computational Linguistics**, MIT Press, v. 4, p. 357–370, 2016.

CIAMPAGLIA, G. L. et al. Computational fact checking from knowledge networks. **PloS one**, Public Library of Science, v. 10, n. 6, p. e0128193, 2015.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

CORTES, E. G.; WOLOSZYN, V.; BARONE, D. A. When, where, who, what or why? a hybrid model to question answering systems. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. [S.l.], 2018. p. 136–146.

COSTA, Â. et al. An english-portuguese parallel corpus of questions: translation guidelines and application in smt. In: **Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)**. [S.l.: s.n.], 2012.

DANG, H. T.; KELLY, D.; LIN, J. J. Overview of the trec 2007 question answering track. In: **Trec**. [S.l.: s.n.], 2007. v. 7, p. 63.

FREITAS, C. et al. Second harem: advancing the state of the art of named entity recognition in portuguese. In: EUROPEAN LANGUAGE RESOURCES ASSOCIATION. **quot; In Nicoletta Calzolari; Khalid Choukri; Bente Maegaard; Joseph Mariani; Jan Odijk; Stelios Piperidis; Mike Rosner; Daniel Tapias (ed) Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010)(Valletta 17-23 May de 2010) European Language Resources Association**. [S.l.], 2010.

GRAVES, A. Supervised sequence labelling with recurrent neural networks. 2012. **ISBN 9783642212703. URL http://books. google. com/books**, 2012.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **arXiv preprint arXiv:1708.06025**, 2017.

HOVY, E.; HERMJAKOB, U.; RAVICHANDRAN, D. A question/answer typology with surface text patterns. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the second international conference on Human Language Technology Research**. [S.l.], 2002. p. 247–251.

HSU, C.-W.; LIN, C.-J. A comparison of methods for multiclass support vector machines. **IEEE transactions on Neural Networks**, IEEE, v. 13, n. 2, p. 415–425, 2002.

JR, B. F. G. et al. Baseball: an automatic question-answerer. In: ACM. **Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference**. [S.l.], 1961. p. 219–224.

JURAFSKY, D.; MARTIN, J. H. **Speech and language processing**. [S.l.]: Pearson London, 2014.

KANDO, N. et al. Overview of ir tasks at the first ntcir workshop. In: **Proceedings of the first NTCIR workshop on research in Japanese text retrieval and term recognition**. [S.l.: s.n.], 1999. p. 11–44.

KAWAKAMI, K. **Supervised Sequence Labelling with Recurrent Neural Networks**. Thesis (PhD) — PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

KOLOMIYETS, O.; MOENS, M.-F. A survey on question answering technology from an information retrieval perspective. **Information Sciences**, Elsevier, v. 181, n. 24, p. 5412–5434, 2011.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

LAHBARI, I.; OUATIK, S. E. A.; ZIDANI, K. A. Arabic question classification using machine learning approaches. 2017.

LEE, J. Y.; DERNONCOURT, F. Sequential short-text classification with recurrent and convolutional neural networks. **arXiv preprint arXiv:1603.03827**, 2016.

LEHMANN, J. et al. Deqa: Deep web extraction for question answering. In: SPRINGER. **International Semantic Web Conference**. [S.l.], 2012. p. 131–147.

LI, X.; ROTH, D. Learning question classifiers. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 19th international conference on Computational linguistics-Volume 1**. [S.l.], 2002. p. 1–7.

LIN, J. The web as a resource for question answering: Perspectives and challenges. In: CITESEER. **In Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002**. [S.l.], 2002.

LONI, B. A survey of state-of-the-art methods on question classification. Citeseer, 2011.

MA, M. et al. Group sparse cnns for question classification with answer sets. **arXiv preprint arXiv:1710.02717**, 2017.

MAGNINI, B. et al. Overview of the clef 2006 multilingual question answering track. In: SPRINGER. **Workshop of the Cross-Language Evaluation Forum for European Languages**. [S.l.], 2006. p. 223–256.

MAGNINI, B. et al. The multiple language question answering track at clef 2003. In: SPRINGER. **Workshop of the Cross-Language Evaluation Forum for European Languages**. [S.l.], 2003. p. 471–486.

MEYER, D.; WIEN, F. T. Support vector machines. **R News**, v. 1, n. 3, p. 23–26, 2001.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 3111–3119.

MISHRA, A.; JAIN, S. K. A survey on question answering systems with classification. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 28, n. 3, p. 345–361, 2016.

MOHD, M.; HASHMY, R. Question classification using a knowledge-based semantic kernel. In: **Soft Computing: Theories and Applications**. [S.l.]: Springer, 2018. p. 599–606.

MONTEIRO, R. A. et al. Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. [S.l.], 2018. p. 324–334.

MOTHE, J. et al. Experimental ir meets multilinguality, multimodality, and interaction. In: SPRINGER. **Sixth International Conference of the CLEF Association, CLEF**. [S.l.], 2015. v. 15, p. 8–11.

PEÑAS, A. et al. Overview of the clef question answering track 2015. In: SPRINGER. **International Conference of the Cross-Language Evaluation Forum for European Languages**. [S.l.], 2015. p. 539–544.

PETERS, C. et al. **Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark,... Applications, incl. Internet/Web, and HCI**. [S.l.]: Springer Publishing Company, Incorporated, 2009.

RAJPURKAR, P. et al. Squad: 100,000+ questions for machine comprehension of text. **arXiv preprint arXiv:1606.05250**, 2016.

ROSA, F. B.; BARONE, D. Study of a deep learning approach to named entity recognition for portuguese. 2018.

SANGODIAH, A.; MUNIANDY, M.; HENG, L. E. Question classification using statistical approach: A complete review. **Journal of Theoretical & Applied Information Technology**, v. 71, n. 3, 2015.

SARROUTI, M.; ALAOUI, S. O. E. A machine learning-based method for question type classification in biomedical question answering. **Methods of information in medicine**, Schattauer GmbH, v. 56, n. 03, p. 209–216, 2017.

SASIKUMAR, U.; SINDHU, L. A survey of natural language question answering system. **International Journal of Computer Applications**, Foundation of Computer Science, v. 108, n. 15, p. 975–8887, 2014.

SCHLAEFER, N. et al. **OpenEphyra open source QA system**.

SCHULZE, F. et al. Hpi question answering system in bioasq 2016. In: **Proceedings of the Fourth BioASQ workshop**. [S.l.: s.n.], 2016. p. 38–44.

SCOTT, S.; MATWIN, S. Text classification using wordnet hypernyms. **Usage of WordNet in Natural Language Processing Systems**, 1998.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM computing surveys (CSUR)**, ACM, v. 34, n. 1, p. 1–47, 2002.

SILVA, J. et al. From symbolic to sub-symbolic information in question classification. **Artificial Intelligence Review**, Springer, v. 35, n. 2, p. 137–154, 2011.

VOORHEES, E. M.; DANG, H. T. Overview of the trec 2003 question answering track. In: **TREC**. [S.l.: s.n.], 2003. v. 2003, p. 54–68.

VOORHEES, E. M. et al. The trec-8 question answering track report. In: **Trec**. [S.l.: s.n.], 1999. v. 99, p. 77–82.

WANG, W.; YAN, M.; WU, C. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. **arXiv preprint arXiv:1811.11934**, 2018.

WANG, W. et al. Gated self-matching networks for reading comprehension and question answering. In: **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [S.l.: s.n.], 2017. v. 1, p. 189–198.

WILKENS, R. et al. Comunica: a question answering system for brazilian portuguese. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations**. [S.l.], 2010. p. 21–24.

WILKENS, R. S.; VILLAVICENCIO, A.; VICARI, R. A study of the use of natural language processing for conversational agents. 2016.

YAMAMOTO, T. et al. Overview of the ntcir-12 imine-2 task. In: **Proc. 12th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Quesiton Answering, And Cross-Lingual Information Access**. [S.l.: s.n.], 2016. p. 94–123.

YANG, Y.; PEDERSEN, J. O. A comparative study on feature selection in text categorization. In: **Icml**. [S.l.: s.n.], 1997. v. 97, p. 412–420.

YAO, X. Lean question answering over freebase from scratch. In: **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations**. [S.l.: s.n.], 2015. p. 66–70.

ZAREMBA, W.; SUTSKEVER, I.; VINYALS, O. Recurrent neural network regularization. **arXiv preprint arXiv:1409.2329**, 2014.

ZHOU, C. et al. A c-lstm neural network for text classification. **arXiv preprint arXiv:1511.08630**, 2015.