UMA ABORDAGEM PARA EXTRAÇÃO DE RELAÇÕES ENTRE ENTIDADES NOMEADAS UTILIZANDO AUTOENCODER E GRADIENT BOOSTING

Ricardo Luiz da Silva Melo

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2022

UMA ABORDAGEM PARA EXTRAÇÃO DE RELAÇÕES ENTRE ENTIDADES NOMEADAS UTILIZANDO AUTOENCODER E GRADIENT BOOSTING

Ricardo Luiz da Silva Melo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Geraldo Bonorino Xexéo
Prof. Jano Moreira de Souza
Prof. Carla Amor Divino Delgado

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2022

Melo, Ricardo Luiz da Silva

Uma abordagem para extração de relações entre entidades nomeadas utilizando autoencoder e gradient boosting/Ricardo Luiz da Silva Melo. – Rio de Janeiro: UFRJ/COPPE, 2022.

IX, 34 p. 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2022.

Referências Bibliográficas: p. 26 – 30.

1. Named Entities. 2. Relation extraction. 3. Autoencoder. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Quero agradecer, em primeiro lugar, a Deus, por ter me dado forças para chegar ao fim dessa jornada.

Dedico este trabalho especialmente a minha mãe Angela (in memoriam), que apesar de não estar presente neste momento, sempre foi e continua sendo fonte de inspiração para a minha vida.

Agradeço a meu pai Ricardo e minha irmã Christine por terem me dado todo apoio e incentivo para que eu pudesse realizar meus sonhos. Também agradeço aos meus avós, Paulo e Neide, e a todos os meus familiares que estiveram presentes em minha vida.

Ao Professor Geraldo Xexéo, meu orientador, pela oportunidade e paciência na realização deste trabalho, que espero fazer jus a confiança dedicada a mim.

Ao Professor Fellipe Duarte e Gustavo Guedes, pela paciência, orientação, confiança e incentivo durante todo o mestrado.

Aos professores Jano Moreira de Souza e Carla Amor Divino Delgado, por aceitarem fazer parte da banca avaliadora desse trabalho, abrindo mão de seus compromissos e sacrificando o tempo de sua agenda.

A todos os colegas de pesquisa do LINE pelo trabalho em conjunto e ajuda no desenvolvimento desse trabalho.

Aos meus amigos da Rural, que mesmo de longe, me incentivaram sempre.

Ao Programa de Engenharia de Sistemas e Computação, em especial a linha de Engenharia de Dados e Conhecimento, pela oportunidade a mim dada para o desenvolvimento deste trabalho.

E a todos aqueles que me auxiliaram de alguma forma durante o curso.

O presente trabalho foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UMA ABORDAGEM PARA EXTRAÇÃO DE RELAÇÕES ENTRE ENTIDADES NOMEADAS UTILIZANDO AUTOENCODER E GRADIENT BOOSTING

Ricardo Luiz da Silva Melo

Setembro/2022

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

O avanço da tecnologia, dos sistemas digitais e das comunicações tem levado a uma geração contínua de grandes quantidades de dados, em sua maioria textos não estruturados, tornando inviável processá-los e analisá-los manualmente. A necessidade de ferramentas para extrair informações de forma automática de dados semiestruturados ou não estruturados deu origem ao campo da Extração de Informação (EI). A EI possui diversas sub-tarefas, e uma delas é a extração de relações (ER). ER visa identificar relações semânticas entre entidades nomeadas que foram previamente identificadas em textos por meio da tarefa reconhecimento de entidades nomeadas (REN). A tarefa de extração de relações é usada principalmente como base para criação de soluções para sistemas de perguntas e respostas, vinculação textual e busca semântica. A solução proposta é composta por algumas etapas que podem ser definidas como a criação de *embeddings* para representação das entidades nomeadas no espaço vetorial, o processo de geração de *embeddings* para relações entre os pares de entidades nomeadas por meio de um *autoencoder* e, por fim, o processo de treinamento do modelo de classificação binário, baseado na técnica de *gradient boosting trees*, que tem como objetivo identificar se há ou não relação entre os pares de entidades nomeadas. A solução proposta foi avaliada em dois conjuntos de dados nos idiomas português e inglês, e então foi comparada com um *framework* de referência na literatura, o *OpenNRE*. Em comparação com o *OpenNRE*, os resultados mostraram que a proposta obteve um valor de F_1 8.1% para o português e 16.7% para o idioma inglês.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AN APPROACH TO RELATION EXTRACTION BETWEEN NAMED ENTITIES USING AUTOENCODER AND GRADIENT BOOSTING

Ricardo Luiz da Silva Melo

September/2022

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

The advance of technology, digital systems and communications has led to a continuous generation of massive amounts of data, mostly unstructured text, becoming infeasible to process and analyze them manually. The need for tools to automatically extract information from semistructured or uninstructed data gave rise to the field of Information Extraction (IE). IE has many sub-tasks, and one of them, among many others, is the Relation Extraction (RE). RE is the task of automatically extracting semantic relationships between named entities that were previously recognized in texts by means of the Named Entity Recognition (NER) task. The RE task is mainly used as a base for Question Answering (QA), Textual Entailment (TE) systems and Semantic Search. The solution proposed for the RE task is based on supervised learning and is composed by three steps. In the first step, embeddings are generated for each entity annotated in the data-set. In the second step, an auto-encoder is used to generate relation embeddings from each pair of entities that have a semantic relationship in the data-set. Finally, the embeddings generated in the first and second steps are then combined as features to build a classification model based on gradient boosting trees technique. The proposed solution was evaluated on two data-sets from different languages, Portuguese and English, and then compared to one of the benchmark solutions available in the literature, the OpenNRE. Compared to OpenNRE, the results showed that the proposal achieved an 8.1% higher F_1 score for the Portuguese language and 16.7% higher F_1 score for the English language.

Sumário

1 Introdução **1**
1.1 Problema e Motivação 1
1.2 Objetivo 2
1.3 Estrutura deste trabalho 2

2 Revisão **4**
2.1 Reconhecimento de Entidades Nomeadas 4
2.2 Extração de Relações 5
2.3 Word Embeddings 6
2.4 Auto encoders 8
2.5 Classificadores 9
2.5.1 Gradient Boosting Trees 10
2.5.2 K-Nearest Neighbors algorithm (KNN) 10
2.5.3 Stochastic gradient descent (SGD) 11
2.5.4 Support-Vector Machine (SVM) 12
2.6 Trabalhos Relacionados 13

3 Proposta **14**
3.1 Pré-processamento dos dados 15
3.2 *Embeddings* de entidades nomeadas 15
3.3 Embedding das relações 16
3.4 Modelo de classificação 17

4 Experimentos **18**
4.1 Conjuntos de dados 18
4.2 Métricas de Avaliação 19
4.3 Parâmetros dos experimentos 20
4.4 Resultados 21
4.4.1 Resultados para o idioma português 22
4.4.2 Resultados para o idioma inglês 23

5 Conclusão **25**

Referências Bibliográficas **26**

A Resultados complementares **31**

Lista de Figuras

2.1 Exemplo de entidades nomeadas. 5
2.2 Exemplo de extração de relações entre entidades nomeadas. 6
2.3 Exemplo de *word embedding* no espaço vetorial. Imagem baseada em BIRAJDAR (2021) 7
2.4 Exemplo das arquiteturas CBOW e Skip-gram. Imagem baseada em (MIKOLOV *et al.*, 2013a) 8
2.5 Arquitetura genérica do autoencoder. Imagem baseada em BANK *et al.* (2020) 9
2.6 Arquitetura de geração de árvores do Gradient Boosting Tree. Imagem baseada em GAURAV (2021) 10
2.7 Exemplo de aplicação do KNN. Imagem baseada em SAJI (2021) . . 11
2.8 Exemplo de aplicação do SGD para classificação. Imagem baseada em SGD (2017) 12
2.9 Exemplo de aplicação do SVM para classificação. Imagem baseada em (VAZ, 2018) 12

3.1 Fluxograma da proposta 14
3.2 *Autoencoder* proposto 16
3.3 Representação do conjunto de dados 17

4.1 Resultados do *OpenNRE* para o idioma português 22
4.2 *Learning Rate* para o *XGBoost* em português 23
4.3 Resultados do OpenNRE para o idioma inglês 23
4.4 *Learning Rate* para o *XGBoost* em inglês 24

Lista de Tabelas

2.1 Entidades nomeadas. 5

4.1 Parâmetros do modelo *OpenNRE* 20
4.2 Parâmetros do modelo *XGBoost* 20
4.3 Resultados dos experimentos para a escolha do classificador da proposta 21
4.4 Resultados do *XGBoost* para o idioma português 22
4.5 Resultados do *XGBoost* para o idioma inglês 24

A.1 Resultados do *XGBoost* para o idioma português 31
A.2 Resultados do *XGBoost* para idioma o inglês 32

Capítulo 1

Introdução

1.1 Problema e Motivação

O avanço dos sistemas e comunicações digitais tem levado a uma geração contínua de grandes quantidades de dados, principalmente na forma de texto não estruturado, tornando inviável processá-los e analisá-los manualmente. A necessidade de ferramentas para extrair automaticamente informações de dados semiestruturados ou não estruturados levou à criação do área de Extração de Informação (EI).

A área de Extração de Informação possui diversas subtarefas, e uma delas, entre muitas outras, é a subtarefa de Extração de Relações (ER). ER é a subtarefa que visa extrair automaticamente relações semânticas entre entidades nomeadas, que foram previamente reconhecidas em textos por meio da tarefa de Reconhecimento de Entidades Nomeadas (REN), e as identifica em documentos textuais e as classifica em categorias predefinidas. A tarefa ER é usada principalmente como base para as áreas de busca semântica, sistemas de perguntas e respostas e de vínculo textual.

Em uma busca semântica baseada em grafos, por exemplo, os documentos podem ser indexados por suas entidades nomeadas e relações. E para realizar a busca, a questão é transformada em um grafo, que é então usado para encontrar correspondências entre documentos com base em uma pontuação de similaridade. Outro exemplo, desta vez aplicado a sistemas de resposta a perguntas, está no uso de extração de relações para selecionar frases em passagens de texto que melhor respondem a uma determinada pergunta. Isso é feito extraindo relações com técnicas de desambiguidade e análises morfológicas.

Como um exemplo concreto na área de busca semântica, pode-se citar o processo de pesquisa no acervo digital do Centro de Documentação e Imagem (CEDIM) localizado na Universidade Federal Rural do Rio de Janeiro. O CEDIM é um espaço de pesquisa na ramo da História, voltado ao público acadêmico e ao público em geral, e caracteriza-se como um canal sistematizado para a reunião e disponibilização de

documentação textual, visual, iconográfica e sonora (CEDIM, 2016). O processo de pesquisa no CEDIM era realizado através do acesso a um conjunto de diretórios que dividem os arquivos por critérios como tema, época, tipo, dentre outros. Como pode ser observado, o CEDIM não contava com um mecanismo de busca adequado, e consequentemente, o processo de pesquisa acabava sendo feito manualmente pelos usuários, o que acabava levando a uma falta de eficiência no uso do tempo disponível aos usuários. Este problema acabou motivando um projeto de final de conclusão de um curso de graduação, cujo objetivo era criar um mecanismo de busca de forma que fosse possível realizar a busca por meio da indexação desses documentos. Nesse trabalho foi entregue um motor de busca semântica que realizava as buscas por meio de uma indexação baseando-se em entidades nomeadas. Umas das melhorias possíveis naquele projeto seria uma identificação automática das relações entre as entidades nomeadas, por meio de um modelo de aprendizado de máquina, onde seria possível extrair relações semânticas entre as entidades nomeadas mapeadas nos documentos. Essa necessidade acabou motivando a proposta detalhada na presente dissertação de mestrado.

1.2 Objetivo

Esta dissertação tem como objetivo propor uma abordagem para o problema de extração de relações entre entidades nomeadas, e essa abordagem consiste numa proposta a resolução de um problema de classificação. Dado um par de entidades nomeadas, a abordagem proposta tem que ser capaz de identificar ou não a existência de uma relação semântica entre o par.

A fim de solucionar o problema citado, neste trabalho foi proposta a utilização de *embeddings* como uma forma de representação dos dados textuais no espaço vetorial, para que essa representação possa então ser utilizada como atributos de entrada para um modelo de classificação. Para que isso seja possível, uma sequencia de etapas precisam ser executadas e podem ser definidas como a geração de *embeddings* para as entidades nomeadas, geração de embeddings para representar a relação entre um par de entidades nomeadas e, por último, o treinamento de um modelo de classificação de forma supervisionada utilizando os dados gerados na etapa anterior.

1.3 Estrutura deste trabalho

Neste trabalho, os próximos capítulos estão estruturados da seguinte forma: O capítulo 2 apresenta uma visão geral sobre reconhecimento de entidade nomeadas, extração de relações entre entidades nomeadas, *word embeddings*, *autoendoers*, bem como descreve trabalhos relacionados ao tema proposto.

A proposta é apresentada no capítulo 3, descrevendo o processo de criação das *word embeddings* para representação das entidades nomeadas no espaço vetorial, o processo de geração da representação das relações entre os pares de entidades nomeadas por meio de um *autoencoder* e, por fim, o processo de treinamento do modelo de classificação binário para a identificação de relação ou não entre os pares.

O capítulo 4, que detalha os experimentos, descreve a construção dos conjuntos de dados, tanto português quanto inglês, as métricas de avaliação utilizadas, os parâmetros utilizados nos modelos de classificação, assim como os resultados e comparações de desempenho entre a proposta e o modelo de referência, o OpenNRE.

Por último, o capítulo 5 apresenta as conclusões tiradas a partir dos resultados, além de apresentar melhorias e trabalhos futuros a serem realizados.

Capítulo 2

Revisão

A Recuperação da Informação é uma tarefa que pode ser definida como a busca por dados em formato não estruturado, geralmente documentos, que satisfazem a necessidade por uma informação dentro de grandes bases de dados (MANNING *et al.*, 2008). Essa tarefa pode ser executada em bases de dados que possuam diferentes ordens de grandeza, e geralmente a ordem de grandeza dos dados pode ser definida por meio de três escalas distintas: documentos na *Web*, que podem exceder a casa dos bilhões em quantidade; documentos locais encontrados em computadores pessoais; e documentos encontrados em bases de dados centrais que geralmente guardam informações institucionais, empresariais ou de um domínio específico (MANNING *et al.*, 2008).

Dentre as muitas tarefas da Recuperação de Informação, podem ser citadas o Reconhecimento de Entidade Nomeadas (REN) e a Extração de Relações (RE). Para execução dessas tarefas, são utilizadas diversas técnicas de Processamento de Linguagem Natural (PLN)(MARRERO *et al.*, 2013). Neste capítulo, podem ser vistas as definições das tarefas, as técnicas de execução abordadas pelo trabalho e alguns trabalhos relacionados.

2.1 Reconhecimento de Entidades Nomeadas

O Reconhecimento de Entidades Nomeadas (REN) pode ser definido como o processo de extração e classificação de informações de regiões de um texto, que corresponde ao nome de uma entidade (MARRERO *et al.*, 2013). O termo foi introduzido pela primeira vez na sexta edição de uma conferência nomeada como *Message Understanding Conference* (MUC) (muc, 1995), e surgiu como uma forma de identificar nomes próprios (Pessoas, Organizações, Locais, etc) e expressões numéricas (tempo, data, quantidades, percentuais, etc) (ZHOU e SU, 2002). As tarefas NER e RE são comumente usadas para dar suporte aos sistemas *Question Answering* (QA) e *Textual Entailment* (TE).

A figura 2.1 apresenta um exemplo de frase e a classificação de alguns termos após o processo de Reconhecimento de Entidades Nomeadas (REN). No exemplo podem ser vistos os termos: "Julio do Valle", "pracinhas"e "II Guerra"; e, respectivamente, suas entidades classificadas: "Pessoa", "Grupo"e "Evento".

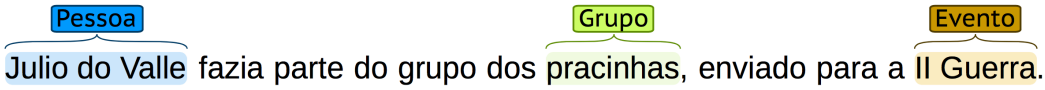


Figura 2.1: Exemplo de entidades nomeadas.

A tabela 2.1 apresenta alguns exemplos de entidades nomeadas que podem ser extraídas de um determinado documento.

Nomenclatura	**Descrição**
Pessoa	Nomes próprios de pessoas, sendo estes nomes completos ou parciais. (ex.: Julio do Valle; Julio; Valle)
Evento	Citação de eventos históricos importantes. (ex.: Segunda Guerra Mundial).
Local	Nomes próprios de locais (ex.: Rio de Janeiro; Alemanha).
Data	Datas, completas ou parciais, de acontecimentos históricos (ex.: 11 de novembro de 1942; maio de 1945).
AutorReporter	Nomes próprios de autores e/ou repórteres envolvidos na elaboração do noticiado.
Pesquisador	Nomes próprios de pesquisadores citado no noticiário.
Organização	Nomes de organizações, empresas, instituições, etc. (ex.: FEB; Polícia Militar).
Fonte	Citação da fonte da informação.
TempoFonte	Data da publicação da notícia.
URLFonte	Endereço da fonte disponibilizado pela rede.
Artefato	Nome dado a um mecanismo construído para um fim determinado. (ex.: Bomba Atômica)
Quantidade	Indicador de quantidade para algum evento pertinente (ex.: 8 mil mortos; mais de 100 pessoas feridas).

Tabela 2.1: Entidades nomeadas.

2.2 Extração de Relações

O processo de Extração da Informação (EI) possui muitas sub-tarefas, e uma delas, é a Extração de Relações (RE). A RE é a tarefa que tem como objetivo extrair automaticamente as relações semânticas entre as entidades nomeadas, que foram previamente reconhecidas por meio da tarefa de Reconhecimento de Entidades Nomeadas (REN) HIRSCHBERG e MANNING (2015).

O termo “Extração de Relação” é frequentemente usado na literatura para se referir a RE a nível global ou RE a nível de menção. No método RE a nível global, espera-se um corpus de texto como entrada e produção de uma lista de pares que possuam um certo grau de relação semântica como saída. Por outro lado, no método RE a nível de menção, espera-se um par acompanhado por algum tipo de informação contextual, que geralmente é a sentença que os contém.

A figura 2.2 apresenta um exemplo de frase, onde as entidades previamente classificadas pelo processo de REN são relacionadas entre si pelo processo de Extração de Relações (RE).No exemplo podem ser vistas as relações entre as entidades: "Pessoa", "Grupo"e "Evento".

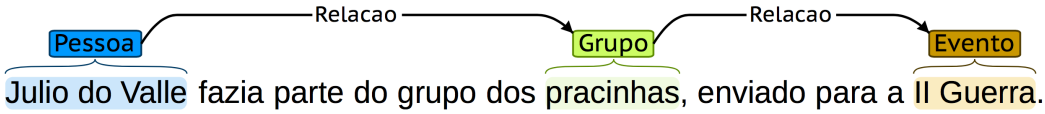


Figura 2.2: Exemplo de extração de relações entre entidades nomeadas.

Segundo PAWAR *et al.* (2017), as abordagens encontradas na literatura estão majoritariamente contidas nos seguintes tipos de abordagens: supervisionadas, não-supervisionadas e semi-supervisionadas. Nas abordagens supervisionadas, o aprendizado é realizado utilizando uma base de entidades e relações previamente anotadas, onde cada par de entidades nomeadas dado como entrada no modelo é associado a um tipo de relação baseado em um conjunto pré-definido de relações.

Nas abordagens não-supervisionadas, o aprendizado é realizado sobre uma base de dados sem anotação prévia de relações, onde as relações são geralmente extraídas por meio de técnicas que se baseiam em detecção de padrões ou tendências.

Nas abordagens semi-supervisionadas, o aprendizado é feito combinando abordagens de aprendizado supervisionado e não-supervisionado, onde a base previamente anotada é utilizada como uma referência de padrão de relações para a extração de relações na base não anotada.

2.3 Word Embeddings

No campo de Processamento de Linguagem Natural (PLN), muitos modelos de aprendizado de máquina recebem como entrada documentos representados por meio de vetores de palavras (*word vectors*). Os *word vectors* são o resultados do mapeamento das palavras ou das sentenças do documento para o espaço vetorial dos números reais. Por muito tempo a representação *Bag of Words* (BoW) foi utilizada como o método padrão para representar os documentos no espaço vetorial ZHANG *et al.* (2010) mas, atualmente, a representação dos documentos por meio do *word embedding* foi introduzida como uma nova alternativa MIKOLOV *et al.* (2013b).

A principal diferença entre esses dois métodos de representação dos documentos no espaço vetorial é: o *bag of word* utiliza a representação por meio de vetores esparsos, enquanto que o *word embdding* usa a representação por meio de vetores pertencentes a um espaço vetorial de representação reduzida.

De acordo com LI e YANG (2018); GLOBERSON *et al.* (2007), *word embedding* é o conjunto de métodos de modelo de linguagem ou métodos de escolha de características, no qual o seu principal objetivo é mapear palavras ou frases para o espaço contínuo de baixa dimensionalidade. Para LAI *et al.* (2016), o *word embedding* tem a capacidade de obter informações sintáticas e semânticas de uma palavra, onde o conhecimento semântico está atrelado ao significado da palavra, e a sua função sintática está relacionada com as regras estruturais do contexto que está inserida.

Palavras com conceitos parecidos são propensas a apresentarem comportamento semelhante no espaço contínuo como afirma HINTON *et al.* (1986); ou seja, as *word embeddings* das palavras com o mesmo conceito, tendem a estarem próximas no espaço contínuo R^n. Colaborando com a afirmação de HINTON *et al.* (1986), SUN *et al.* (2015) menciona que as palavras encontradas em contextos semelhantes são propensas a apresentarem semântica ou função sintática semelhante.

A figura 2.4 apresenta exemplos de termos em um espaço contínuo R^n qualquer, onde os termos semelhantes semanticamente e sintaticamente estão próximos. No caso semântico, podem ser visto os termos *Woman* (mulher) e *Man* (homem) próximos em uma dimensão e *Woman* (mulher) e *Queen* (rainha) em outra dimensão. No caso sintático, podem ser vistos os termos *Small* (pequeno) e *Smallest* (menor) próximos em uma dimensão e *Small* (pequeno), e *Big* (grande) em outra.

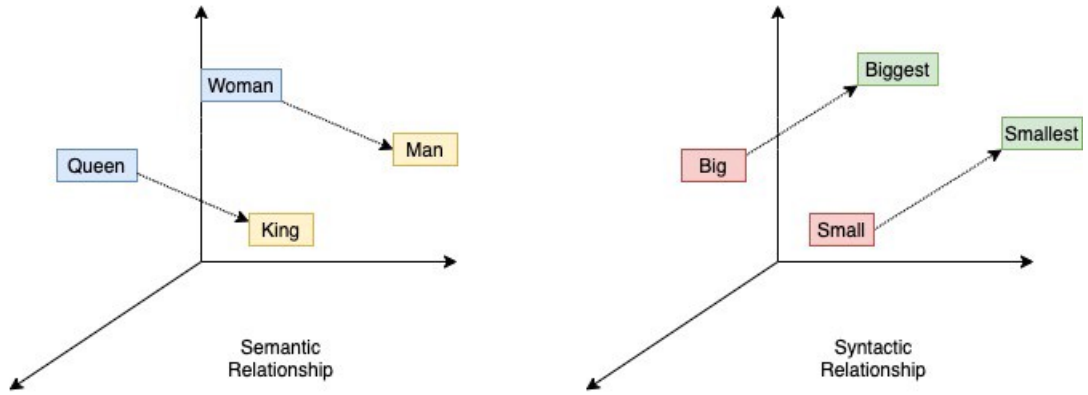


Figura 2.3: Exemplo de *word embedding* no espaço vetorial. Imagem baseada em BIRAJDAR (2021)

Ao longo do tempo foram surgindo diversos métodos que implementam o conceito de *word embeddings*, sendo o *Word2Vec* (MIKOLOV *et al.*, 2013c), *Glove* (PENNINGTON *et al.*, 2014) e *BERT* (DEVLIN *et al.*, 2019) os mais utilizados. O *Word2Vec* constrói um vocabulário a partir do conjunto de treinamento para então

gerar as representações vetoriais das palavras, e faz isso por meio da implementação de duas arquiteturas para a geração de representações de palavras no espaço vetorial, uma denominada *CBOW* e outra *continuous skip-gram*.

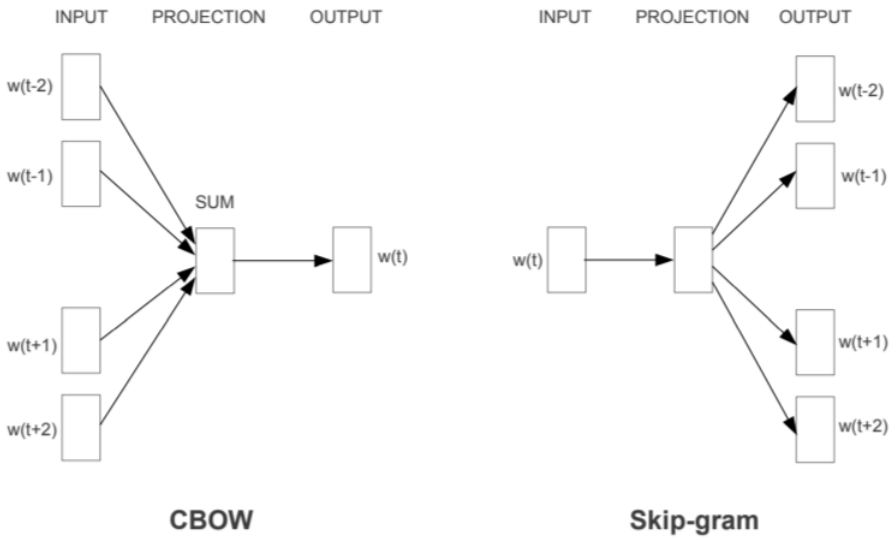


Figura 2.4: Exemplo das arquiteturas CBOW e Skip-gram. Imagem baseada em (MIKOLOV *et al.*, 2013a)

O método *continuous bag-of-words* (CBOW) se baseia em uma representação distribuída continua do contexto e, neste método, a representação de uma palavra é gerada a partir da utilização do contexto das palavras que a antecedem e a sucedem MIKOLOV *et al.* (2013b), como pode ser visto na figura 2.4. Semelhante ao *CBOW*, o método *skip-gram* ao invés de prever a palavra atual com base no contexto, ele tenta maximizar a classificação de uma palavra com base em outra palavra na mesma frase. Mais precisamente, usamos cada palavra atual como entrada para um classificador *log-linear* com camada de projeção para prever palavras dentro de um determinado intervalo antes e depois da palavra atualMIKOLOV *et al.* (2013c), como pode ser visto na figura 2.4.

2.4 Auto encoders

Os *autoencoders*, de maneira simplificada, são circuitos de aprendizado de máquina que têm por objetivo transformar entradas em saídas com a menor distorção possível BALDI (2012). Uma outra definição para *autoencoders*, segundo BANK *et al.* (2020), é que são uma espécie de rede neural de aprendizado não supervisionado, designada para codificar a entrada em uma representação significativa e comprimida, e então ser capaz de decodificá-la de forma mais fiel possível à entrada original.

A arquitetura do *autorencoder*, de maneira geral, apresenta-se da seguinte forma: um codificador responsável por receber e codificar a entrada original criando uma representação comprimida; um decodificador que recebe a representação comprimida e a decodifica criando uma reconstrução da entrada original o mais similar possível à entrada, como apresenta a figura2.5. A figura exemplifica uma codificação de uma imagem, porém as propriedades dos *autoencoders* também se aplicam a dados textuais.

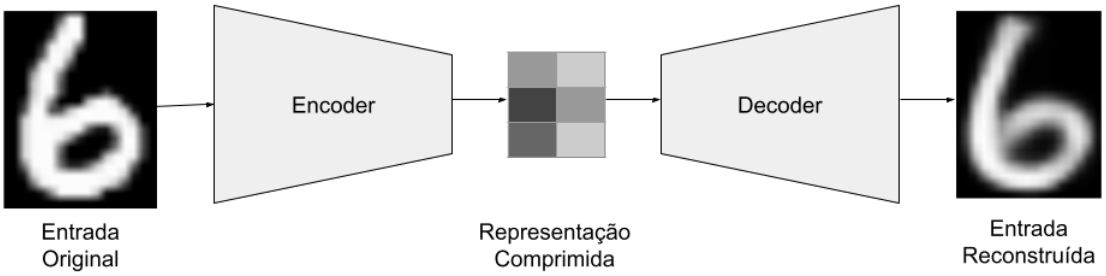


Figura 2.5: Arquitetura genérica do autoencoder. Imagem baseada em BANK *et al.* (2020)

Dentro da literatura acadêmica existem alguns tipos de *autorencoders* como: *sparse autroencoders* que lidam com o *tradeoff* viés-variância por meio de multi-camadas, onde em determinadas camadas nem todos os neurônios transmitirão o seu sinal para os neurônios da camada posterior desse modo forçando a esparsidade. Isso ocorre por conta das funções de ativação que podem ser aplicadas aos neurônios, como *ReLU*, *Sigmoid*, *Softplus* e entre outras; o outro *autoencoder* é o *denoising* que tem por finalidade minimizar o erro entre a entrada reconstruída e a entrada original, e isso é feito por meio de adição de ruídos aos dados de entrada na expectativa que consiga reconstruir a entrada original; além desses existem outros *autoencoders* como os do tipo *Contractive*, *Marginalized De-noising* e o *Variational* BANK *et al.* (2020); ARPIT *et al.* (2015).

Os *autorencoders* podem ser aplicados à diversas áreas de acordo com a literatura como na tarefa de classificação auxiliando como extrator de atributos; para gerar representação comprimida pelos na tarefa de grupamento, visto que o processo de agrupamento é sensível à alta dimensionalidade; como redutores de dimensionalidade; além de diversas outras aplicações que não foram citadas BANK *et al.* (2020); ARPIT *et al.* (2015).

2.5 Classificadores

Nesta seção serão apresentados alguns métodos de classificação de treinamento supervisionado, onde os modelos são treinados a partir de um conjunto de dados de

exemplos, juntamente com a variável objetivo que se deseja aprender. Os classificadores apresentados serão utilizados nos experimentos do capítulo 4 e são os seguintes: *Stochastic Gradient Descent* (SGD), *K-Nearest Neighbors* (KNN), *Support Vector Machine*s (SVM) e *Gradient Boosting Trees*.

2.5.1 Gradient Boosting Trees

O *Gradient Boosting Trees* é uma técnica de *ensemble* baseada no uso de estimadores mais fracos, ou seja, árvore de decisão de baixa acurácia. Pode ser usado tanto para classificação quanto para regressão. A cada etapa de execução uma árvore de decisão é criada para treinar a partir do resíduo da anterior, onde ruído significa exemplos que foram classificados erroneamente, esse processo é repetido até todos os exemplos sejam classificados de forma correta, ou até algum critério de parada. O algoritmo utiliza gradientes (derivadas) para computar os resíduos. A cada etapa, de forma incremental, os resultado da classificação de cada árvore é agregado (GAURAV, 2021). Na figura 2.6 pode ser visto como o modelo funciona ao criar árvores através dos exemplos residuais.

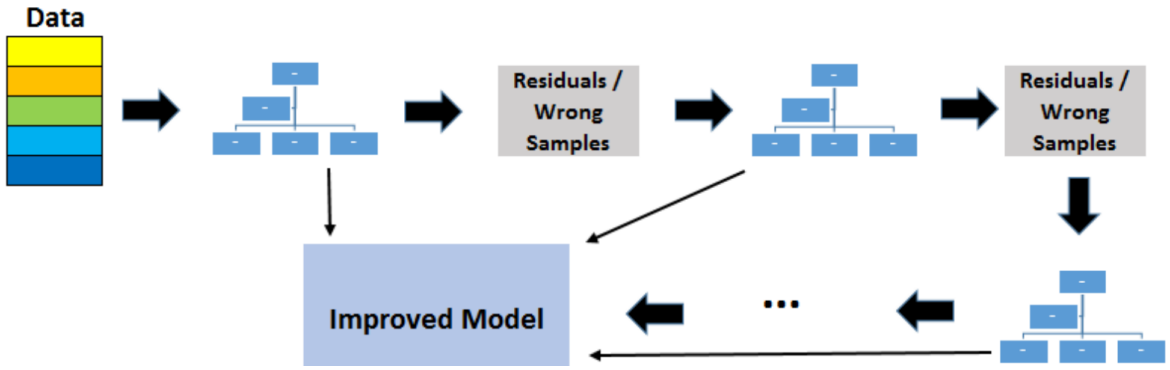


Figura 2.6: Arquitetura de geração de árvores do Gradient Boosting Tree. Imagem baseada em GAURAV (2021)

2.5.2 K-Nearest Neighbors algorithm (KNN)

K-Nearest Neighbor (KNN) é um dos algoritmos de aprendizado de máquina mais simples baseado na técnica de aprendizado supervisionado. O KNN tem como objetivo classificar um novo dado a partir de quão similar um dado é quanto a outrosHASTIE *et al.* (2009); COVER e HART (1967).

A aplicação do KNN pode ser definida pelos seguintes passos SAJI (2021); HASTIE *et al.* (2009): o recebimento de um dado de entrada x não classificado; a medição da distância Euclidiana entre o dado x e todos os outros dados do conjuntos previamente classificado; a obtenção das menores k distâncias; a verificação de qual classe

a maioria dos dados classificados pertence; e para finalizar, a classificação do novo dado de acordo com a classe da maioria.

O valor do k deve ser escolhido previamente para a classificação adequada do novo dado. Na figura 2.7 pode ser visto um exemplo de aplicação do KNN e da importância da escolha do k. Um dado novo qualquer (estrela vermelha) é classificado como *class A* (bolas azuis), caso $k = 6$ e como *class B* (bolas verdes), caso $k = 3$ HASTIE *et al.* (2009).

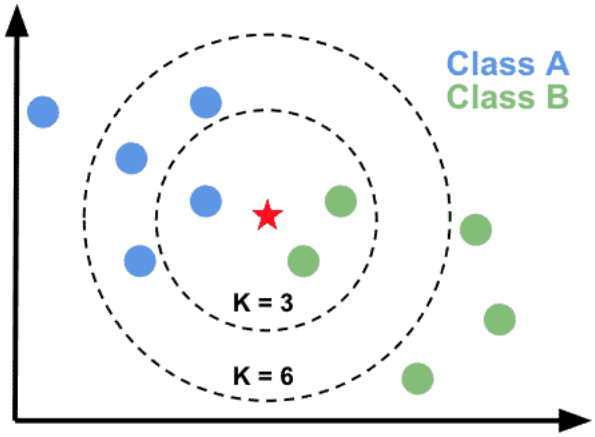


Figura 2.7: Exemplo de aplicação do KNN. Imagem baseada em SAJI (2021)

2.5.3 Stochastic gradient descent (SGD)

O *Stochastic Gradient Descent* (SGD) é um método iterativo que tem como objetivo, otimizar uma função objetivo escolhendo uma instância aleatória do conjunto de treinamento em cada etapa e calculando o gradiente baseado nessa única instância. Isso torna o algoritmo rápido, possibilitando o uso em dados esparsos, como de problemas classificação de texto e no processamento de linguagem natural. Na figura 2.8 pode ser visto um exemplo de implementação da classificação usando o SGD (GÉRON, 2019).

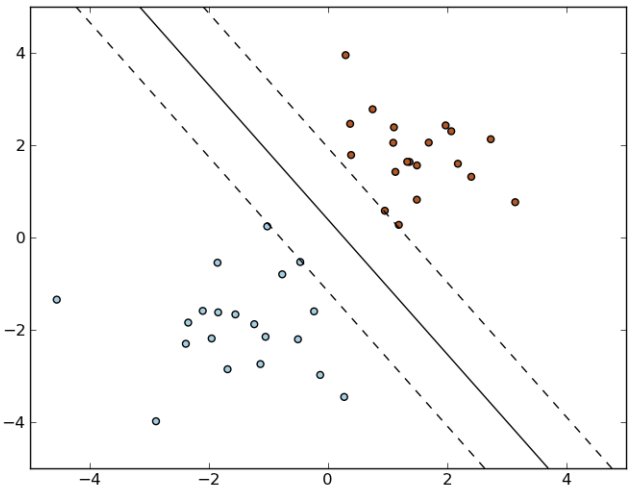


Figura 2.8: Exemplo de aplicação do SGD para classificação. Imagem baseada em SGD (2017)

2.5.4 Support-Vector Machine (SVM)

O Support Vector Machine (SVM) é um modelo de aprendizado de máquina capaz de realizar classificação, regressão e até detecção de *outliers*. Para classificação, o SVM pode realizar tanto uma classificação linear, como não linear (GÉRON, 2019).

Na figura 2.9 pode ser visto um exemplo da aplicação do SVM em classificação linear, onde um hiperplano é traçado com a maior distância até dados mais próximos das classes, criando um limite entre os grupos de classes. A margem é definida após diversas iterações (GÉRON, 2019).

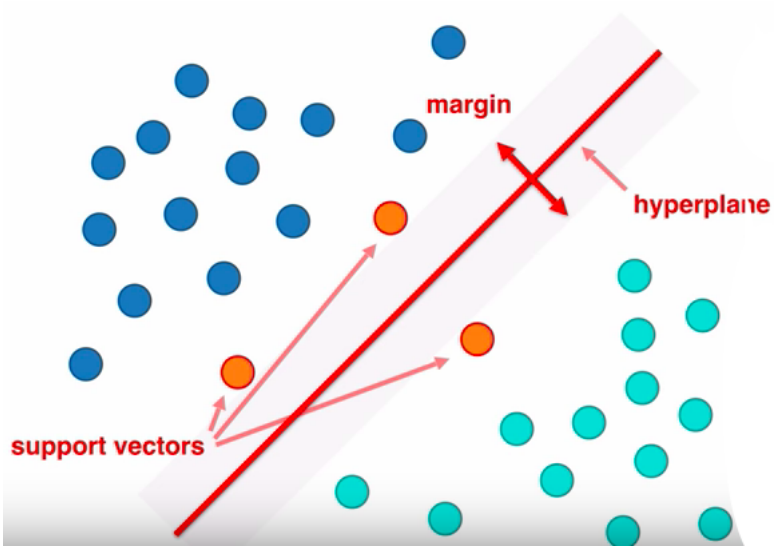


Figura 2.9: Exemplo de aplicação do SVM para classificação. Imagem baseada em (VAZ, 2018)

2.6 Trabalhos Relacionados

Esta seção apresenta de forma sucinta quais os métodos empregados em algumas soluções na literatura para a resolução do problema de extração de relações entre entidades nomeadas. Mais especificamente, o OpenNRE foi escolhido como solução de referência devido a disponibilização dos códigos para execução da proposta para fins de comparação na época dos experimentos.

OpenNRE é um *framework* para extração de relações de forma supervisionada distante, baseada em uma rede neural convolucional com mecanismo de atenção. Este *framework* propõe um *pipeline* de aprendizado. Na estrutura, os embeddings de sentenças são combinados para gerar representações de relações semânticas. Como a supervisão distante introduz algum ruído no processo de anotação, o mecanismo de atenção ao nível de sentença em várias instâncias (PCNN+ATT) é aplicado a fim de mitigar o problema. A partir dos resultados apresentados, a solução proposta foi melhor do que as propostas por Mintz (MINTZ *et al.*, 2009), MultiR (HOFFMANN *et al.*, 2011) e MIML (SURDEANU *et al.*, 2012) na época do experimento.

HRERE é um *framework* de extração de relações (ER) que usa informações de uma base conhecimento para melhorar a tarefa de ER (XU e BARBOSA, 2019). O objetivo é alcançado aprendendo as representações tanto de linguagem quanto de conhecimento, simultaneamente, durante o treinamento do modelo. O modelo de aprendizagem subjacente é uma rede LSTM bidirecional com três funções de perda, uma para cada uma das duas representações citadas e uma terceira para garantir que elas não divirjam. O *framework* foi avaliado no conjunto de dados do NYT (Riedel et al., 2010), e a partir dos resultados, superou o melhor modelo disponível no *framework* OpenNRE, PCNN+ATT (HAN *et al.*, 2019).

O *framework* RESIDE propõe uma solução que usa informações do tipo *side embedding*, obtidas de uma base de conhecimento e combinadas com informações sintáticas codificadas obtidas de uma rede neural convolucional baseada em grafo (VASHISHTH *et al.*, 2018). Essa combinação mostrou melhorias na tarefa de Extração de Relações. Analisando os resultados, a solução proposta apresentou melhor desempenho que o modelo PCNN+ATT do framework OpenNRE (HAN *et al.*, 2019).

Capítulo 3

Proposta

Este trabalho se propõe a apresentar uma solução para o problema de extração de relações entre entidades nomeadas que, como mencionado no Capítulo 2, é uma tarefa que tem como alvo a identificação de existência ou não de relações semânticas entre entidades nomeadas. A proposta deste trabalho possui foco principal no idioma português, mais especificamente no campo de conhecimento da História, porém a solução também foi avaliada no idioma inglês para fins de comparação. O resultado final da proposta pode ser definido como a criação de um modelo de classificação que, dado um par de entidades nomeadas como entrada, é capaz de indicar se existe ou não uma relação semântica entre o par.

Para isso, uma sequência de etapas envolvendo um conjunto de técnicas de aprendizado de máquina foram executadas seguindo o seguinte fluxo, que também pode ser visto na figura 3.1: pré-processamento dos textos anotados; representação das entidades nomeadas no espaço vetorial por meio de *embeddings*; criação de *embeddings* para as relações por meio de uma codificação de dimensionalidade reduzida obtida através de um *autoencoder*; e por fim, a criação de um modelo de classificação binária para extração de relações semânticas entre pares de entidades nomeadas, utilizando as *embeddings* das relações como atributos de treinamento do modelo.

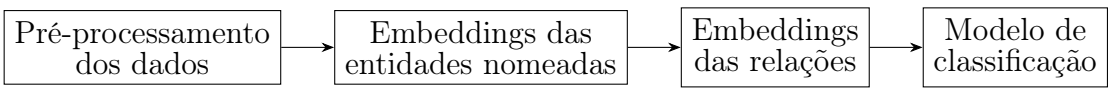


Figura 3.1: Fluxograma da proposta

As *embeddings* das entidades nomeadas são obtidas treinando as entidades como *tokens* únicos ou compostos. Após a geração dos *embeddings* das entidades, o segundo passo é geração das *embeddings* das relações. Nesta etapa, os pares de *embeddings* entre entidades que possuem relação são submetidas a um *autoencoder*, e os vetores resultantes são as *embeddings* que representam a relação entre os pares.

3.1 Pré-processamento dos dados

Dois conjunto de dados, em dois idiomas, foram usados para realizar os experimentos, um em português e outro em inglês. Para os experimentos, ambos conjuntos de dados passaram por algumas etapas de preparação descritas nos próximos parágrafos.

O conjunto de dados em português foi criado a partir de um processo de anotação manual para um projeto de pesquisa em conjunto com o CEDIM. No presente trabalho, o conjunto de dados passou por um processo de revisão, que resultou em algumas correções nas anotações tanto de entidades nomeadas quanto de relações. Os dados anotados, como exemplificados nas figuras 2.1 e 2.2, não possuem anotações de exemplos de não relação, fazendo com que fosse necessário gerar exemplos para essa categoria originalmente não existente. Para isso foi adotada a premissa de que uma não relação pode ser definida como um par de duas entidades nomeadas que não nunca foram anotadas como tendo uma relação semântica entre si no conjunto de dados. O conjunto de não relação foi então gerado através da combinação de pares, obedecendo a premissa mencionada.

O conjunto de dados NYT10, disponibilizado por (RIEDEL *et al.*, 2010), foi escolhido para os experimentos em inglês dado que ele foi o conjunto de dados utilizado pelo *OpenNRE*, o modelo de referência utilizado para comparação nos experimentos. Foi necessário adequar o conjunto de dados ao escopo do experimento. A adequação realizada foi a remoção do qualificador das relações para que fossem agrupadas em duas categorias, relação e não relação.

3.2 *Embeddings* de entidades nomeadas

Ambos os experimentos codificam as entidades nomeadas por meio de *embeddings* a fim de representá-las no espaço vetorial. Para o conjunto de dados em português, um modelo de *word embedding* foi treinado usando o método *Word2vec*.

Para a criação do conjunto de dados em português, os documentos primeiramente passaram por uma etapa de pré-processamento, na qual cada documento foi submetido a um processo de tokenização, onde cada palavra foi representada como um *token*, com exceção às entidades nomeadas compostas que foram preservadas como palavras compostas. A fim de se obter pelo menos uma representação para cada entidade nomeada, foi definida uma contagem de frequência que considera que cada *token* elegível à geração de uma *embedding* tenha aparecido pelo menos uma vez no texto. A partir desses parâmetros, o modelo foi treinado para gerar *embeddings* de tamanho 50 utilizando uma janela contextual de tamanho 5. O método de treinamento baseado em *skip-gram* foi escolhido por gerar boas representações para

tokens pouco frequentes, além de gerar representações melhores do que o *CBOW* em conjuntos de dados relativamente pequenos. A escolha desses parâmetros foi de encontro às características do conjunto de dados em português, que foi anotado manualmente para um contexto específico. O capítulo 4.1 possui um detalhamento sobre a construção do conjunto de dados em português.

3.3 Embedding das relações

Após a criação das *embeddings* das entidades nomeadas na etapa anterior, se inicia a etapa de criação das *embeddings* que representam as relações entre os pares de entidades nomeadas. Para isso, foi proposto um modelo de *autoencoder* para que se pudesse obter *embeddings* de relações utilizando as *embeddings* dos pares de entidades nomeadas como entrada para o modelo.

Dada a proposta de se criar *embeddings* para a representação das relações, ou seja, representações em um espaço vetorial de dimensão reduzida, o *autoencoder* proposto foi do tipo simétrico, multi-camada e regularizado. Esse tipo de *autoencoder* foi escolhido pois seu objetivo principal não é gerar uma decodificação fiel à entrada, mas sim uma codificação de dimensionalidade reduzida, mantendo as informações mais relevantes para a representação do par, geradas a partir da camada intermediária de codificação. A figura a seguir representa a estrutura do *autoencoder* proposto.

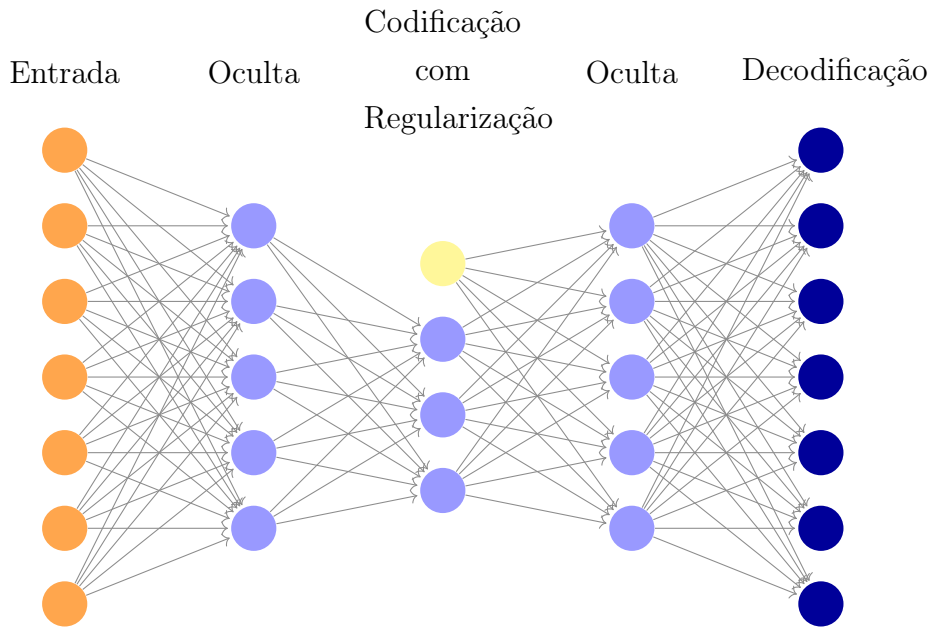


Figura 3.2: *Autoencoder* proposto

A composição de camadas da rede neural ficou definida da seguinte forma: uma camada de entrada de tamanho 100, uma camada oculta de tamanho 75 com função de ativação $ReLU$; uma camada oculta de tamanho 50 com função de ativação

ReLU e um método de regularização (camada de codificação); uma camada oculta de tamanho 75 com função de ativação *ReLU*; e uma camada de saída de tamanho 100 (camada de decodificação).

No *autoencoder* proposto, a entrada da rede neural foi definida como a concatenação entre os vetores de embeddings do par de entidade nomeadas. Baseando-se nos experimentos e achados de ARPIT *et al.* (2016), a função de ativação *ReLU* foi utilizada nas camadas intermediárias a fim de garantir a esparsidade na geração da *embedding* da relação. Além disso, na camada intermediária de codificação foi aplicado um método de regularização a fim de se evitar que o modelo tenha sua capacidade de generalização reduzida.

3.4 Modelo de classificação

Após a geração das *embeddings* das relações dos pares de entidades nomeadas na etapa anterior, as *embeddings* geradas foram então utilizadas como atributos de entrada de um modelo de classificação, que tem como objetivo identificar se existe relação ou não entre um par de entidades nomeadas. Nesta etapa, cada exemplo do conjunto de treinamento é representado por um vetor ($\vec{v}$), que é a própria *embedding* da relação, juntamente com uma variável resposta (y) para a realização do treinamento supervisionado do modelo. A figura 3.3 apresenta uma notação do conjunto de treinamento do modelo.

$$\begin{bmatrix} \vec{v_1} \\ \vec{v_2} \\ \vdots \\ \vec{v_{n-1}} \\ \vec{v_n} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

Figura 3.3: Representação do conjunto de dados

Dada a construção do conjunto de dados para o treinamento de um modelo de classificação, inicia-se a etapa de escolha do classificador. Para os experimentos foram utilizados os seguintes classificadores: *SGD*, *KNN*, *SVM* e o *XGBoost*, sendo o último uma implementação da técnica de *Gradient Boosting Trees*.

Capítulo 4

Experimentos

Neste capítulo é descrito como os conjuntos de dados para os experimentos foram preparados, quais métricas de avaliação e desempenho foram utilizadas, como os modelos dos experimentos foram parametrizados, como foi realizado o processo de escolha do modelo de classificação da proposta e quais foram os resultados obtidos. Além disso também é apresentada uma discussão sobre os resultados obtidos pelos experimentos.

4.1 Conjuntos de dados

O conjunto de dados em português contém 1.522 relações e 8.256 não relações, totalizando 9.778 registros anotados de um conjunto de artigos da Wikipédia. Após análises manuais sobre o balanceamento da base, foi realizado um *undersampling*, selecionando não relações aleatoriamente sem reposição até que a categoria de não relações tivesse o mesmo número de registros da de relações. Com esse processamento, a base final do experimento passou a possuir um total de 1.522 relações e 1.522 não relações, totalizando 3.044 registros.

O conjunto de dados NYT10, que contém notícias em inglês do portal jornalistico *The New York Times* sobre diversos temas, foi criado por meio de anotação utilizando o método *distant supervision* a fim de aprender relações semânticas de textos, de forma supervisionada, apoiado por uma base de conhecimento (RIEDEL *et al.*, 2010). Este conjunto de dados contém 570.087 relações e 413.404 não relações, totalizando 983.491 registros. Devido ao fato do mapeamento das *embeddings* ser feito utilizando um modelo pré-treinado, nem todas as entidades nomeadas possuiam *embeddings* para representá-las. Devido a isso, para o experimento foram considerados apenas os registros que possuíam *embeddings* mapeadas para cada entidade nomeada, fazendo com que a base fosse reduzida para um total de 7.982 relações e 190.986 não relações, totalizando 198.968 exemplos. Assim como no conjunto de dados em português, foi realizado um *undersampling* a fim de balancear as categorias,

tendo a base final do experimento passado a possuir um total de 15.964 registros, composto por 7.982 relações e 7.982 não relações.

4.2 Métricas de Avaliação

A fim de se avaliar os experimentos, foi necessária a definição de uma metologia de avaliação para que se garanta que os experimentos fossem avaliados com um mesmo método de validação baseado em uma métrica de desempenho do modelo. A métrica principal de desempenho escolhida para os experimentos foi a métrica F_1, que é baseada em outras duas métricas, a precisão e a cobertura.

A precisão (P) pode ser definida como a taxa de itens classificados corretamente em função do total itens que foram classificados, conforme mostrado na fórmula 4.1, onde vp significa verdadeiro positivo e fp, falso positivo. A cobertura (C) pode ser definida como a taxa de itens que foram classificados corretamente em função de todos os itens da amostra que deveriam ter sido classificados, conforme mostrado na fórmula 4.2 onde fn significa falso negativo. A F_1, como definido na 4.3, pode ser entendida como uma forma de representação do equilíbrio entre precisão e cobertura por ser calculada como média ponderada entre essa duas métricas mencionadas.

$$P = \frac{vp}{vp + fp} \tag{4.1}$$

$$C = \frac{vp}{vp + fn} \tag{4.2}$$

$$\mathrm{F}_1 = 2 * \left(\frac{P * C}{P + C} \right) \tag{4.3}$$

No processo de validação foi utilizado o método de validação cruzada, onde o conjunto de dados é dividido em n partes iguais, que então são usadas para a realização de n processos de treinamento. Em cada treinamento as partes são distribuídas da seguinte forma: cada parte é utilizada uma vez com um subconjunto de validação de um modelo que é treinado utilizando as $n - 1$ partes restantes. Neste trabalho foram utilizadas 10 partes, que é um valor comumente utilizado na literatura. Esse método de validação permite uma análise de capacidade de generalização do modelo de classificação em geral.

4.3 Parâmetros dos experimentos

A realização do experimento foi dividida em quatro partes: testes dos modelos de classificação da solução proposta e do *OpenNRE* com os conjuntos de dados em português e inglês. Para todos os experimentos, a otimização de hiper-parâmetros foi usada para encontrar a melhor combinação de parâmetros para cada modelo. Além disso, cada modelo treinado a partir de cada possibilidade combinação de parâmetros foi validado utilizando a validação cruzada juntamente com a métrica F_1 como critério de comparação de desempenho.

Para os classificadores, a faixa de valores para cada parâmetro foi escolhida com base nas características dos classificadores escolhidos, além de uma análise exploratória manual. Por outro lado, para o *OpenNRE*, o intervalo de valores para cada parâmetro foi escolhido gerando limites inferiores e superiores a partir dos valores ótimos dos resultados do conjunto de dados original. Nos experimentos, as mesmas faixas de valores para cada parâmetro dos classificadores foram utilizadas para ambos os idiomas, onde o melhor resultado para cada idioma pôde ser definido como uma escolha de combinação de parâmetros que resultar no maior valor de F_1. As faixa de valores utilizadas para cada parâmetro do modelo *OpenNRE* estão apresentadas na tabela 4.1 e, para o modelo proposto com o *XGBoost*, na tabela 4.2.

Learning rate	[0.25, 0.5, 0.75, 0.1]
Max epochs	[100, 150, 230]

Tabela 4.1: Parâmetros do modelo *OpenNRE*

Learning rate	[0.1, 0.25, 0.5, 0.75]
Max depth	[6, 9, 12]
Subsample	[0.8, 1.0]
Number of estimators	[100, 1000]

Tabela 4.2: Parâmetros do modelo *XGBoost*

No experimento com o *XGBoost*, foram escolhidos 4 parâmetros a serem explorados, e podem ser definidos da seguinte forma: o *learning rate* (taxa de aprendizado) define a que ritmo os pesos do modelo são atualizados, o *max depth* (profundidade máxima) define a profundidade das árvores do modelo, o *subsample* (subamostra) define a porcentagem do número de exemplos que cada árvore irá treinar e o *number of estimators* quantidade de modelos define a quantidade máxima de modelos. No caso do experimento com o *OpenNRE*, os 2 parâmetros escolhidos podem ser definidos da seguinte forma: o *learning rate* (taxa de aprendizado) define a que ritmo os pesos do modelo são atualizados e o *max epochs* (número máximo de épocas)

define o máximo de vezes que o algoritmo de aprendizado processa todos os dados do conjunto de treinamento.

O desenvolvimento de todas as etapas da proposta foi realizado na linguagem *Python*, utilizando as bibliotecas *SK-learn*(PEDREGOSA *et al.*, 2011) e *XGBoost*(CHEN e GUESTRIN, 2016) para os classificadores, *Gensin*(REHUREK e SOJKA, 2011) para geração das embeddings das entidades, *Keras*(CHOLLET *et al.*, 2015) para a criação do *autoencoder*. Além disso, a execução de todas as etapas da proposta foram aceleradas via GPU, utilizando a tecnologia CUDA(NVIDIA *et al.*, 2020) oferecida pela fabricante *NVIDIA*.

Os experimentos foram executados utilizando os serviços de computação em nuvem da *Amazon Web Services* (AWS). Pelas características dos experimentos, como por exemplo, o treinamento de uma rede neural do tipo *deep learning* que se beneficia de aceleração por GPU, a máquina virtual escolhida possui a seguinte configuração: 8 núcleos de CPU do tipo Intel Xeon, 1 GPU Nvidia T4, 32Gb de memória RAM e armazenamento do tipo SSD NVMe com 225Gb de capacidade.

4.4 Resultados

Nesta seção, os resultados dos experimentos foram separados em uma etapa de experimentação preliminar para escolha do modelo da proposta em ambos os idiomas português e inglês. Dada a escolha do melhor classificador para proposta, foi realizada uma análise de hiper-parâmetros para a proposta e para o *OpenNRE*, sendo que essa análise foi dividida por idioma a fim de facilitar as comparações.

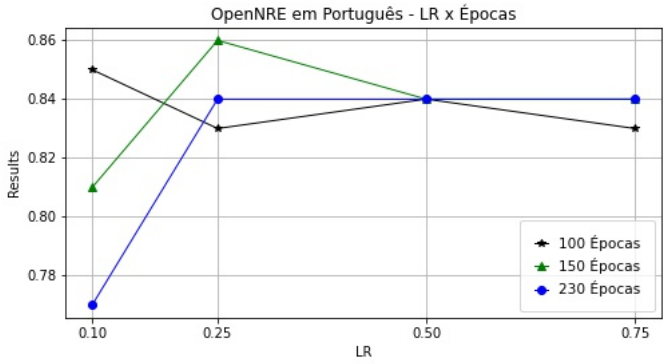
A partir da análise dos resultados dos experimentos preliminares para a escolha do classificador da proposta, como pode ser visto na tabela 4.3, onde *std* significa desvio padrão. Os melhores resultados de F_1 foram menores do que o do *XGBoost* para ambos os idiomas analisados, como pode ser visto nas tabelas 4.5 e 4.2. Devido a isso, o *XGBoost* foi escolhido como o classificador da proposta e foi submetido, juntamente com o *OpenNRE*, à etapa de otimização de hiper-parâmetros a fim de se obter o máximo de desempenho para ambas soluções.

Classificador	Português (F_1 | std)	Inglês (F_1 | std)
SGD	0.54 (0.08)	0.70 (0.01)
KNN	0.81 (0.01)	0.84 (0.01)
SVM	0.77 (0.02)	0.57 (0.03)

Tabela 4.3: Resultados dos experimentos para a escolha do classificador da proposta

4.4.1 Resultados para o idioma português

Após a execução dos experimentos para o idioma português, os resultados de ambas soluções puderam ser analisados. Os resultados para execução de otimização de hiper-parâmetros do *OpenNRE*, como podem ser vistos na tabela da figura 4.1, mostram que para o conjunto de dados em português, o maior valor de F_1 obtido foi de 0.86 com uma taxa de aprendizado de 0.25 e 150 épocas. No modelo proposto, a combinação dos parâmetros taxa de aprendizado de 0.10, profundidade máxima de 9, quantidade de modelos de 1000 e subamostra de 0.80, resultou no maior valor de F_1, que foi de 0.93, e isso representa um ganho de aproximadamente 8.1% em relação ao melhor resultado do *OpenNRE*.



LR	Épocas 100	150	230
0.1	0.85	0.81	0.77
0.25	0.83	**0.86**	0.84
0.5	0.84	0.84	0.84
0.75	0.83	0.84	0.84

Figura 4.1: Resultados do *OpenNRE* para o idioma português

learning_rate	F_1	std
0.10	0.9332	0.0073
0.25	0.9316	0.0066
0.50	0.9272	0.0080
0.75	0.9218	0.0092

Tabela 4.4: Resultados do *XGBoost* para o idioma português

Analisando-se o gráfico da figura 4.2, que apresenta alguns dados do modelo *XGBoost* da proposta, pode se notar o impacto do parâmetro *learning rate* nos valores de F_1, onde mesmo com uma diferença numérica relativamente pequena, é possível notar uma tendência de decréscimo nos valores de F_1 conforme os valores de *learning rate* aumentam. Além disso, os valores relativamente baixos para a coluna *std* (desvio padrão) na tabela 4.4, indicam que o modelo tem uma boa capacidade de generalização.

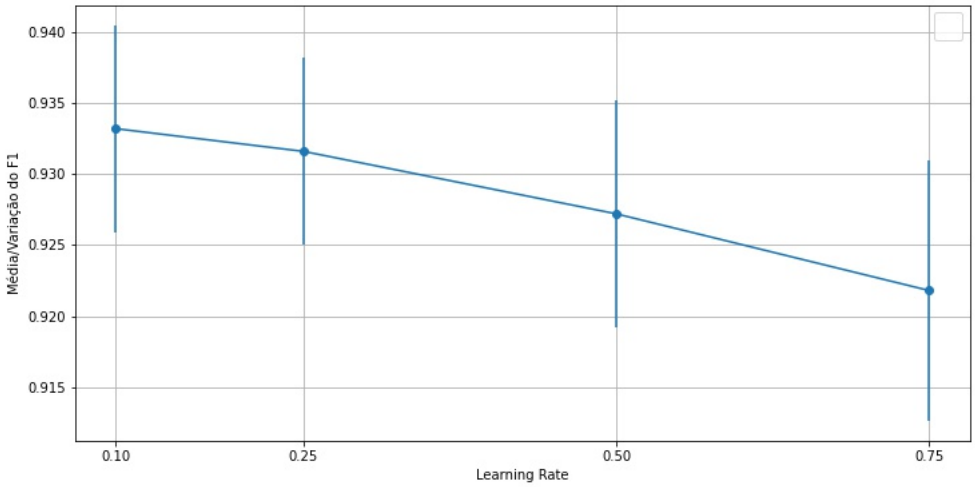
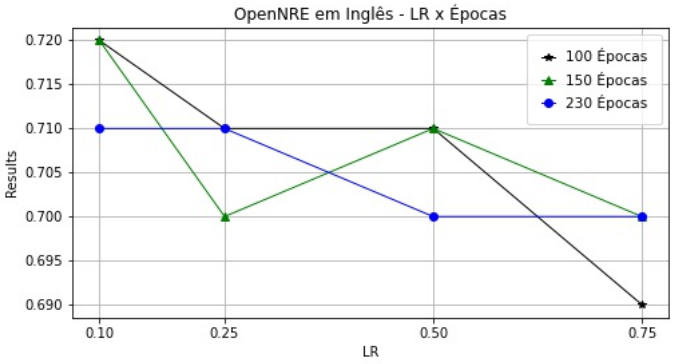


Figura 4.2: *Learning Rate* para o *XGBoost* em português

4.4.2 Resultados para o idioma inglês

Após a execução dos experimentos para o idioma inglês. Os resultados para execução de otimização de hiper-parâmetros do *OpenNRE*, como podem ser visto na tabela 4.3, mostram que para o conjunto de dados em inglês, o maior valor obtido de F_1, que foi de 0.72, foi obtido com uma taxa de aprendizado de 0.1 e 100 ou 150 épocas. No modelo proposto, a combinação dos parâmetros taxa de aprendizado de 0.10, profundidade máxima de 12, quantidade de modelos de 1000 e subamostra de 0.80, resultou no maior valor de F_1, que foi de 0.84, como pode ser visto na tabela 4.5. Comparando-se ao *OpenNRE*, houve um ganho de aproximadamente 16.6% em relação ao melhor resultado do *OpenNRE*.



LR	Épocas 100	150	230
0.1	**0.72**	**0.72**	0.71
0.25	0.71	0.70	0.71
0.5	0.71	0.71	0.70
0.75	0.69	0.70	0.70

Figura 4.3: Resultados do OpenNRE para o idioma inglês

learning_rate	F_1	std
0.10	0.8396	0.0038
0.25	0.8328	0.0050
0.50	0.8219	0.0080
0.75	0.8163	0.0098

Tabela 4.5: Resultados do *XGBoost* para o idioma inglês

Assim como visto no idioma português, para o modelo XGboost da proposta, de acordo com o gráfico da figura 4.4, também pode-se notar o impacto do parâmetro *learning rate* nos valores de F_1, onde mesmo com uma diferença numérica relativamente pequena, é possível também notar uma tendência de decréscimo nos valores de F_1 conforme os valores de *learning rate* aumentam. Além disso, os valores relativamente baixos para a coluna *std* (desvio padrão) na tabela 4.4 também indicam que o modelo tem uma boa capacidade de generalização.

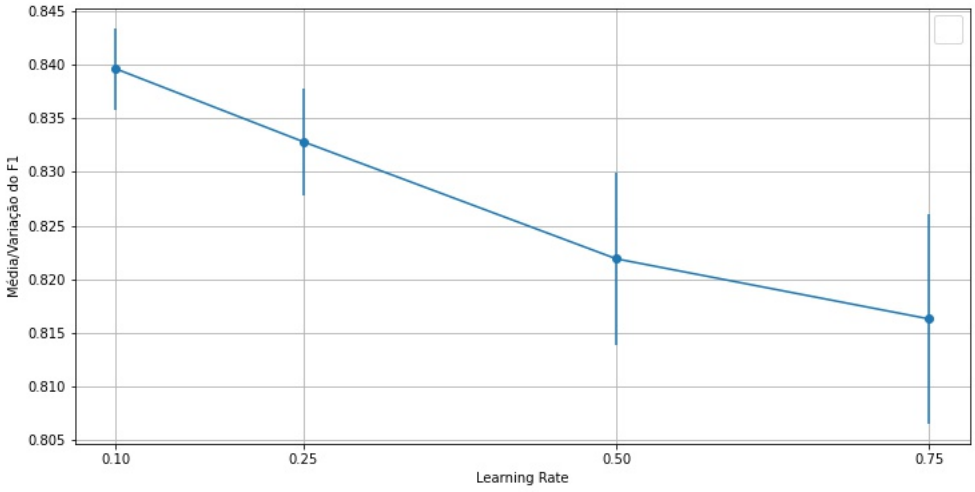


Figura 4.4: *Learning Rate* para o *XGBoost* em inglês

Capítulo 5

Conclusão

Motivado pela continuidade de um projeto que se iniciou no CEDIM, este trabalho aplica técnicas da área de recuperação da informação, propondo uma solução para o problema de extração de relações entre entidades nomeadas. A solução proposta pode ser definida como o processo de criação das embeddings das entidades nomeadas, o processo de geração de embeddings para relações entre os pares de entidades nomeadas por meio de um *autoencoder* e, por fim, o processo de treinamento do modelo de classificação binário com a função de identificar se existe ou não relação entre um par de entidades nomeadas. No contexto do CEDIM, a identificação automática das relações aceleraria o processo de indexação da base de documentos.

Como contribuição, fica a revisão e disponibilização de um conjunto de dados em português contento entidades nomeadas e relações anotadas dentro do campo de conhecimento da história. Além disso, como proposta de solução para o problema apresentado, foi criada uma representação para as entidades nomeadas por meio de *word embeddings* e, posteriormente, foi proposto uma representação das relações entre pares de entidades nomeadas por meio de um *autoencoder*, gerando assim um conjunto de dados para treinamento de modelos de classificação capazes de aprender de forma supervisionada, se pares de entidades nomeadas possuem ou não algum tipo de relação semântica. Tanto a proposta quanto o *OpenNRE* foram testados nos idiomas português e inglês, e o resultado obtido pela proposta para o idioma português foi aproximadamente 8.1% melhor do que o modelo de referência *OpenNRE* e, para o idioma inglês, foi aproximadamente 16.7% melhor.

Como melhorias e trabalhos futuros, ficam experimentos com outros conjuntos de dados, geração de novas representações para as entidades nomeadas, utilizando o *BERT*, por exemplo. Além disso, mais testes com outras modelagens de *autoencoders* poderiam ser realizados, assim como testes com outros classificadores.

Referências Bibliográficas

BIRAJDAR, N. "Word2Vec Research Paper Explained". https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30, 2021. Acessado: 2022-08-30.

MIKOLOV, T., LE, Q. V., SUTSKEVER, I. "Exploiting similarities among languages for machine translation", *arXiv preprint arXiv:1309.4168*, 2013a.

BANK, D., KOENIGSTEIN, N., GIRYES, R. "Autoencoders". 2020. Disponível em: <https://arxiv.org/abs/2003.05991>.

GAURAV. "An Introduction to Gradient Boosting Decision Trees". https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees/, 2021. Acessado: 2022-08-30.

SAJI, B. "A Quick Introduction to K – Nearest Neighbor (KNN) Classification Using Python". https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python 2021. Acessado: 2022-08-30.

"Stochastic Gradient Descent". https://scikit-learn.org/stable/modules/sgd.html, 2017. Acessado: 2022-08-30.

VAZ, A. L. "Classificando o paladar de receitas — SVM". https://arthurlambletvaz.medium.com/classificando-o-paladar-das-receitas-svm-bf0fbb185b10, 2018. Acessado: 2022-08-30.

CEDIM. "Centro de Documentação e Imagem". https://goo.gl/gfC3Xg, Agosto, 2016.

MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H., et al. *Introduction to information retrieval*, v. 1. Cambridge university press Cambridge, 2008.

MARRERO, M., URBANO, J., SÁNCHEZ-CUADRADO, S., et al. "Named entity recognition: fallacies, challenges and opportunities", *Computer Standards & Interfaces*, v. 35, n. 5, pp. 482–489, 2013.

"Appendix C: Named Entity Task Definition (v2.1)". In: *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*, 1995.

ZHOU, G., SU, J. "Named entity recognition using an HMM-based chunk tagger". In: *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 473–480. Association for Computational Linguistics, 2002.

HIRSCHBERG, J., MANNING, C. D. "Advances in natural language processing", *Science*, v. 349, n. 6245, pp. 261–266, 2015. ISSN: 0036-8075. doi: 10.1126/science.aaa8685. Disponível em: <`https://science.sciencemag.org/content/349/6245/261`>.

PAWAR, S., PALSHIKAR, G. K., BHATTACHARYYA, P. "Relation extraction: A survey", *arXiv preprint arXiv:1712.05191*, 2017.

ZHANG, Y., JIN, R., ZHOU, Z.-H. "Understanding bag-of-words model: a statistical framework", *International Journal of Machine Learning and Cybernetics*, v. 1, n. 1-4, pp. 43–52, 2010.

MIKOLOV, T., CHEN, K., CORRADO, G., et al. "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013b.

LI, Y., YANG, T. "Word embedding for understanding natural language: a survey". In: *Guide to Big Data Applications*, Springer, pp. 83–104, May 2018.

GLOBERSON, A., CHECHIK, G., PEREIRA, F., et al. "Euclidean embedding of co-occurrence data", *Journal of Machine Learning Research*, v. 8, n. Oct, pp. 2265–2295, October 2007.

LAI, S., LIU, K., HE, S., et al. "How to generate a good word embedding", *IEEE Intelligent Systems*, v. 31, n. 6, pp. 5–14, May 2016.

HINTON, G. E., OTHERS. "Learning distributed representations of concepts". In: *Proceedings of the eighth annual conference of the cognitive science society*, v. 1, p. 12. Amherst, MA, August 1986.

SUN, F., GUO, J., LAN, Y., et al. "Learning word representations by jointly modeling syntagmatic and paradigmatic relations". In: *Proceedings of the*

53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 136–145, July 2015.

MIKOLOV, T., SUTSKEVER, I., CHEN, K., et al. "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems*, v. 26, 2013c.

PENNINGTON, J., SOCHER, R., MANNING, C. D. "Glove: Global Vectors for Word Representation." In: *EMNLP*, v. 14, pp. 1532–1543, 2014.

DEVLIN, J., CHANG, M.-W., LEE, K., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, jun. 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. Disponível em: <https://aclanthology.org/N19-1423>.

BALDI, P. "Autoencoders, Unsupervised Learning, and Deep Architectures". In: Guyon, I., Dror, G., Lemaire, V., et al. (Eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, v. 27, *Proceedings of Machine Learning Research*, pp. 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR. Disponível em: <https://proceedings.mlr.press/v27/baldi12a.html>.

ARPIT, D., ZHOU, Y., NGO, H., et al. "Why Regularized Auto-Encoders learn Sparse Representation?" 2015. Disponível em: <https://arxiv.org/abs/1505.05561>.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H., et al. *The elements of statistical learning: data mining, inference, and prediction*, v. 2. Springer, 2009.

COVER, T., HART, P. "Nearest neighbor pattern classification", *IEEE transactions on information theory*, v. 13, n. 1, pp. 21–27, 1967.

GÉRON, A. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. Alta Books, 2019.

MINTZ, M., BILLS, S., SNOW, R., et al. "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the*

47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 1003–1011, 2009.

HOFFMANN, R., ZHANG, C., LING, X., et al. "Knowledge-based weak supervision for information extraction of overlapping relations". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 541–550, 2011.

SURDEANU, M., TIBSHIRANI, J., NALLAPATI, R., et al. "Multi-instance multi-label learning for relation extraction". In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pp. 455–465, 2012.

XU, P., BARBOSA, D. "Connecting Language and Knowledge with Heterogeneous Representations for Neural Relation Extraction". In: *The 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*. ACL, June 2019.

HAN, X., GAO, T., YAO, Y., et al. "OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction". In: *Proceedings of EMNLP-IJCNLP: System Demonstrations*, pp. 169–174, 2019. doi: 10.18653/v1/D19-3029. Disponível em: <`https://www.aclweb.org/anthology/D19-3029`>.

VASHISHTH, S., JOSHI, R., PRAYAGA, S. S., et al. "RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1257–1266, Brussels, Belgium, out.-nov. 2018. Association for Computational Linguistics. Disponível em: <`http://aclweb.org/anthology/D18-1157`>.

RIEDEL, S., YAO, L., MCCALLUM, A. "Modeling relations and their mentions without labeled text". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 148–163. Springer, 2010.

ARPIT, D., ZHOU, Y., NGO, H., et al. "Why regularized auto-encoders learn sparse representation?" In: *International Conference on Machine Learning*, pp. 136–144. PMLR, 2016.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine learning in Python", *Journal of machine learning research*, v. 12, n. Oct, pp. 2825–2830, 2011.

CHEN, T., GUESTRIN, C. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN: 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. Disponível em: <http://doi.acm.org/10.1145/2939672.2939785>.

REHUREK, R., SOJKA, P. "Gensim–python framework for vector space modelling", *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, v. 3, n. 2, 2011.

CHOLLET, F., OTHERS. "Keras". https://keras.io, 2015.

NVIDIA, VINGELMANN, P., FITZEK, F. H. "CUDA, release: 10.2.89". 2020. Disponível em: <https://developer.nvidia.com/cuda-toolkit>.

Apêndice A

Resultados complementares

Tabela A.1: Resultados do *XGBoost* para o idioma português

learning_rate	max_depth	n_estimators	subsample	f1	std
0.10	9	1000	0.80	0.9332	0.0073
0.10	12	1000	0.80	0.9331	0.0083
0.10	6	1000	0.80	0.9324	0.0089
0.25	9	1000	1.00	0.9321	0.0082
0.25	12	1000	0.80	0.9317	0.0078
0.25	9	1000	0.80	0.9316	0.0066
0.25	6	1000	0.80	0.9308	0.0083
0.10	9	1000	1.00	0.9304	0.0077
0.25	6	1000	1.00	0.9300	0.0081
0.50	12	1000	1.00	0.9295	0.0088
0.10	12	1000	1.00	0.9295	0.0096
0.25	12	1000	1.00	0.9294	0.0087
0.10	6	1000	1.00	0.9292	0.0106
0.50	6	1000	1.00	0.9273	0.0081
0.50	9	1000	0.80	0.9272	0.0080
0.50	6	1000	0.80	0.9269	0.0083
0.50	9	1000	1.00	0.9264	0.0079
0.50	12	100	1.00	0.9260	0.0078
0.25	12	100	0.80	0.9259	0.0082
0.50	12	1000	0.80	0.9252	0.0051
0.75	9	1000	1.00	0.9245	0.0087
0.75	12	1000	1.00	0.9241	0.0087
0.75	6	1000	1.00	0.9240	0.0126

0.50	9	100	0.80	0.9240	0.0094
0.50	12	100	0.80	0.9236	0.0091
0.25	9	100	1.00	0.9228	0.0097
0.25	12	100	1.00	0.9225	0.0078
0.75	12	1000	0.80	0.9223	0.0095
0.10	12	100	0.80	0.9221	0.0097
0.75	6	1000	0.80	0.9221	0.0108
0.75	9	1000	0.80	0.9218	0.0092
0.25	9	100	0.80	0.9213	0.0094
0.75	12	100	1.00	0.9208	0.0071
0.75	9	100	1.00	0.9200	0.0097
0.50	9	100	1.00	0.9196	0.0073
0.10	12	100	1.00	0.9184	0.0115
0.75	12	100	0.80	0.9180	0.0081
0.50	6	100	0.80	0.9176	0.0074
0.75	9	100	0.80	0.9174	0.0104
0.75	6	100	1.00	0.9164	0.0139
0.10	9	100	0.80	0.9163	0.0093
0.50	6	100	1.00	0.9150	0.0110
0.25	6	100	0.80	0.9140	0.0083
0.75	6	100	0.80	0.9132	0.0105
0.10	9	100	1.00	0.9109	0.0098
0.25	6	100	1.00	0.9106	0.0098
0.10	6	100	0.80	0.8895	0.0123
0.10	6	100	1.00	0.8783	0.0126

Tabela A.2: Resultados do *XGBoost* para idioma o inglês

learning_rate	max_depth	n_estimators	subsample	F1	std
0.10	12	1000	0.80	0.8396	0.0038
0.10	9	1000	0.80	0.8387	0.0063
0.10	12	1000	1.00	0.8369	0.0053
0.10	9	1000	1.00	0.8340	0.0038
0.25	9	1000	1.00	0.8337	0.0090
0.25	12	1000	1.00	0.8333	0.0061
0.25	9	1000	0.80	0.8331	0.0094
0.25	6	1000	1.00	0.8330	0.0059

0.25	12	1000	0.80	0.8328	0.0050
0.10	6	1000	0.80	0.8326	0.0034
0.25	6	1000	0.80	0.8322	0.0044
0.10	6	1000	1.00	0.8280	0.0084
0.50	6	1000	1.00	0.8273	0.0063
0.25	12	100	1.00	0.8272	0.0029
0.50	9	1000	1.00	0.8269	0.0065
0.50	12	1000	1.00	0.8256	0.0062
0.50	9	1000	0.80	0.8253	0.0078
0.10	12	100	1.00	0.8239	0.0055
0.10	12	100	0.80	0.8234	0.0064
0.25	12	100	0.80	0.8229	0.0091
0.25	9	100	0.80	0.8224	0.0089
0.50	6	1000	0.80	0.8224	0.0086
0.75	12	1000	1.00	0.8222	0.0033
0.50	12	1000	0.80	0.8219	0.0080
0.75	6	1000	1.00	0.8214	0.0057
0.75	9	1000	1.00	0.8205	0.0063
0.10	9	100	1.00	0.8196	0.0061
0.25	9	100	1.00	0.8193	0.0094
0.10	9	100	0.80	0.8189	0.0078
0.75	6	1000	0.80	0.8189	0.0056
0.50	12	100	1.00	0.8183	0.0058
0.50	9	100	1.00	0.8183	0.0077
0.75	12	1000	0.80	0.8163	0.0098
0.50	9	100	0.80	0.8160	0.0043
0.50	12	100	0.80	0.8139	0.0079
0.75	9	100	1.00	0.8137	0.0060
0.75	12	100	1.00	0.8136	0.0046
0.75	9	1000	0.80	0.8134	0.0076
0.25	6	100	1.00	0.8096	0.0081
0.10	6	100	0.80	0.8093	0.0051
0.25	6	100	0.80	0.8092	0.0053
0.10	6	100	1.00	0.8069	0.0074
0.75	12	100	0.80	0.8060	0.0090
0.75	9	100	0.80	0.8037	0.0084
0.75	6	100	1.00	0.8027	0.0061

0.50	6	100	1.00	0.8026	0.0112
0.50	6	100	0.80	0.8001	0.0083
0.75	6	100	0.80	0.7965	0.0076