UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Avaliação de representações *embeddings* para similaridade sentencial no Português

Ana Carolina Rodrigues

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Ana Carolina Rodrigues

Avaliação de representações *embeddings* para similaridade sentencial no Português

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Ricardo Marcondes Marcacini

USP – São Carlos
Maio de 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

R696a

Rodrigues, Ana Carolina
Avaliação de representações embeddings para similaridade sentencial no Português / Ana Carolina Rodrigues; orientador Ricardo Marcondes Marcacini. -- São Carlos, 2023.
96 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. processamento de língua natural. 2. aprendizado de máquina. 3. similaridade de sentença. 4. embeddings. 5. STS. I. Marcondes Marcacini, Ricardo , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Ana Carolina Rodrigues

Evaluation of *embedding* representations for sentence similarity in Portuguese

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Ricardo Marcondes Marcacini

USP – São Carlos
May 2023

AGRADECIMENTOS

A realização deste trabalho contou com o apoio e incentivos de diferentes formas. Às vezes uma palavra, às vezes uma explicação mais técnica, às vezes uma dica e às vezes uma comida quentinha que trazia ânimo. Foi a junção das pequenas coisas, muitas vezes feitas sem pretensão, que foram construindo o caminho, todas pelas quais sou muito grata.

À minha mãe, que sempre esteve ao meu lado, mesmo quando fiz escolhas que fugiam ao senso comum e a quem seria impossível agradecer o suficiente. Ao meu pai, que com seu olhar analítico e perspicaz do mundo sempre me ajudou. À Nilzete, por todas as vezes que me acolheu em sua casa, de onde sempre saí mais feliz do que quando entrei. Aos meus sogros que tantas vezes me receberam e me hospedaram carinhosamente.

Ao professor Ricardo Marcondes Marcacini, por sua orientação, pelas sugestões e críticas valiosas ao trabalho, pela confiança na minha capacidade, pela disposição amistosa mesmo frente a percalços e pelo total apoio.

Ao professor Roberto Hirata Junior, por me orientar nos primeiros passos que dei na Computação, quando quase nada sabia e sem me julgar mal por eu vir de outra área, e aos professores Marcelo Ferreira e Marcos Lopes, por proporcionarem o primeiro contato que tive com a área de Linguística Computacional, além das excelentes aulas as quais até hoje faço referências mentais.

Aos docentes do Núcleo Interinstitucional de Linguística Computacional (NILC) e do Laboratório de Inteligência Computacional (LABIC) que diretamente e indiretamente me ajudaram, em particular a professora Graça Nunes, com quem dei o pontapé inicial neste trabalho.

Também aos amigos que fiz no NILC e no LABIC. Sou grata e feliz pelos momentos que compartilhamos, dentro e fora do laboratório, trabalhando e rindo.

Além da companhia, agradeço em especial pelas recomendações e momentos de troca acadêmica com Ana Caroline Medeiros Brito, Adèle Ribeiro, Edresson Casanova, Márcio Lima, Marco Cabezudo, Sidney Leal, Ângelo Cesar Mendes da Silva e Marcos Paulo Silva Gôlo.

Ao apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), código PROEX-5660587/M, pelo apoio financeiro.

E principalmente ao Gabriel, por tudo.

RESUMO

RODRIGUES, A. C. **Avaliação de representações *embeddings* para similaridade sentencial no Português**. 2023. 96 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

O mapeamento de texto para representações numéricas que possam ser processadas computacionalmente tornou-se uma etapa essencial no processamento de língua natural (PLN). Mais especificamente, representações vetoriais densas de números reais, conhecidas como *embeddings*, associadas ao uso de algoritmos de aprendizado de máquina baseados em arquiteturas de redes neurais ganharam notoriedade na última década com resultados significativos na área. Existem diversos métodos para gerar estas representações e uma forma tradicionalmente empregada para testá-los é através da identificação de similaridade semântica textual (STS), tarefa na qual o objetivo é determinar o valor de similaridade entre duas sentenças, dado pela anotação humana dos dados a partir de uma escala pré-determinada. Nos últimos anos, o estabelecimento de modelos com arquitetura baseada em *Transformers* introduziu uma variedade de modelos de *embeddings* pré-treinados que tem sido utilizados de forma bem-sucedida no Inglês. Para o Português, versões multilíngues e, em menor grau, específicas para língua, recentemente ampliaram as alternativas a serem exploradas para STS. Existem duas formas de empregar modelos de representações pré-treinadas: *embeddings* podem servir como entrada fixa em algoritmo preditivos ou o modelo que a gera acoplado de forma interativa como parte do algoritmo, permitindo que as representações sejam atualizadas para um fim específico. Desta forma, o papel das representações no processamento de língua para similaridade não fica bem definido, uma vez que os resultados são fruto do sistema como um todo, representações mais algoritmo preditivo. Neste trabalho, investigamos modelos de representações na tarefa de STS considerando diferentes aspectos, sendo os principais: (i) Avaliamos o impacto da escolha do modelo de representação nos resultados em relação aos hiperparâmetros do algoritmo preditivo. (ii) Partindo da hipótese que diferentes modelos codificam características distintas do texto as quais podem ser complementarmente relevantes, testamos combinações de modelos de representações sentenciais pré-treinadas como forma de melhorar o desempenho na predição similaridade sentencial no Português. (iii) Testamos a capacidade de generalização dos resultados de STS no Português de dois modelos para além do dataset original. Os principais resultados obtidos indicam que (i) a escolha do modelo de representação é determinante para o desempenho na tarefa, levando à diferentes faixas de resultados (ii) o uso de modelos em conjunto em uma arquitetura simples é uma alternativa para melhorar o desempenho na tarefa em relação ao uso de modelos sozinhos.

Palavras-chave: embeddings, STS, similaridade sentencial, Português, aprendizado de máquina.

ABSTRACT

RODRIGUES, A. C. **Evaluation of *embedding* representations for sentence similarity in Portuguese**. 2023. 96 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Mapping text into numerical representations that can be computationally processed has become an essential step in natural language processing (NLP). More specifically, dense vector representations of real numbers, known as embeddings, and associated with machine learning algorithms based on neural network architectures have gained notoriety in the last decade with significant results in the area. There are several methods to generate these representations and a traditional way to test them is through the identification of semantic textual similarity (STS), a task whose objective is to determine the similarity score between two sentences, given by human annotation based on a pre-defined scale. In recent years, the establishment of models based on Transformers introduced a variety of pre-trained embedding models that have been used successfully in English. Concerning Portuguese, multilingual and, to a lesser extent, language specific versions, recently expanded the alternatives to be explored for STS. There are two ways to make use of pre-trained representation models, embeddings can serve as fixed input in predictive algorithms, or the model that generates them being connected as part of the algorithm in an interactive manner, in which representations are tuned for a specific purpose. Since results come from the entire system, representations plus predictive algorithm, the part of representations in language processing for similarity is not well defined. In this work, we investigated representation models in the STS task considering different aspects, mainly: (i) We evaluated the impact of representation models in the results compared with the hyperparameters of the predictive algorithm. (ii) Starting from the hypothesis that different models encode distinct features of the text, and they can be complementarily relevant, we tested combinations of pre-trained sentence representation models as a way to improve the performance of sentence similarity prediction in Portuguese. (iii) We tested the generalizability of STS results in Portuguese of two models in addition to the original dataset. The main results obtained indicate that (i) the choice of the representation model is decisive for the performance in the task, leading to the distinct ranges of results (ii) the use of multiple models combined in a simple architecture is an alternative to improve performance in the task compared to the use of models alone.

Keywords: embeddings, STS, sentence similarity, Portuguese, machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contexto e motivação

A escrita faz parte da comunicação humana e é utilizada em larga escala em diversos segmentos da sociedade, como na administração de empresas, redes sociais, veículos de notícia e no exercício do Direito. A produção abundante, diversificada e atualmente em grande parte digitalizada, trouxe um interesse crescente no desenvolvimento de sistemas capazes de processar de forma automatizada informações contidas em textos. Contudo, um dos grandes obstáculos do processamento automatizado é conseguir captar relações de significado em representações numéricas que possam ser manipuladas e processadas computacionalmente. Embora avanços tenham sido alcançado na última década, o desenvolvimento de sistemas sensíveis ao conteúdo, e não apenas a forma, ainda é um desafio para área de Processamento de Língua Natural (PLN) (AGGARWAL, 2018; LAURIOLA; LAVELLI; AIOLLI, 2022).

No início dos anos 70, modelos de espaços vetoriais (do Inglês, VSM [1]) com origem na área de recuperação de informação foram pioneiros na representação de texto em vetores numéricos que pudessem ser usados para o processamento de informação para além da forma. Por exemplo, sua utilização para agrupamento de documentos relacionados por assunto trouxe resultados notáveis para época (TURNEY; PANTEL, 2010).

A partir de 2013, com o trabalho de Mikolov *et al.* (2013a) e Pennington, Socher e Manning (2014), representações obtidas por métodos que mapeiam a distribuição de palavras no texto em vetores densos de valores reais, conhecidas como *embeddings*, ganharam notoriedade. Utilizadas em diversas tarefas, que incluem análise de sentimento, reconhecimento de entidades nomeadas, e detecção de inferência, *embeddings* passaram a ser peça chave no uso de algoritmos de aprendizado de máquina aplicados a PLN. Mais recentemente, modelos de aprendizado profundo que geram representações ajustáveis alcançaram destaque pelo seu alto desempenho

[1] Vector space models

inédito em tarefas conhecidas na área (RADFORD *et al.*, 2018; DEVLIN *et al.*, 2019).

Existem diversos modelos para gerar representações e uma forma tradicionalmente empregada na área de PLN para testá-los é através da tarefa de similaridade semântica. Nesta tarefa, dados manualmente anotados quanto a similaridade de significado são comparados com o resultados obtidos com o uso de *embeddings*. A comparação pode ser feita diretamente testando-se características geométricas de *embeddings* como vetores, em uma abordagem conhecida como avaliação intrínseca, ou indiretamente através dos resultados de sua utilização como entrada em algoritmos preditivos para execução de uma tarefa de PLN, em uma abordagem conhecida como avaliação extrínseca (SCHNABEL *et al.*, 2015).

O uso da tarefa de similaridade semântica para testar *embeddings* se mistura com a própria evolução destas representações, aparecendo em diversos trabalhos ao longo dos anos (TURNEY; PANTEL, 2010; KIELA; HILL; CLARK, 2015; ANTONIAK; MIMNO, 2018; WANG; REIMERS; GUREVYCH, 2021; CHOI *et al.*, 2021). Além de usada para avaliação de modelos de *embeddings*, a identificação de similaridade semântica também tem função prática como tarefa alvo, sendo empregada no agrupamento de avaliações de produtos semelhantes, rastreamento de comentários redundantes em plataformas onlines e categorização de perguntas que possam receber a mesma resposta feita por clientes em sites de empresas. No papel de tarefa alvo de PLN (*downstream task*) no nível sentencial ficou conhecida como identificação de Similaridade Semântica Textual (do Inglês - STS) (ELAVARASI; AKILANDESWARI; MENAGA, 2014).

Apesar de modelos que utilizam *embeddings* serem referenciados como capazes de captar relações de significado, essa relação não é evidente. Resultados de avaliações intrínsecas de *embeddings* de palavras indicam que os vetores podem relacionar-se de diferentes formas, ora por assunto (“mouse” e “teclado”), ora por proximidade de uso (“leite” e “condensado”), ou não apresentarem relação de sentido aparente, o que deixa a dúvida se modelos que geram estas representações refletem algum tipo de relação de sentido de forma sistemática. Resultados de avaliações intrínsecas e extrínsecas também indicam que há diferenças significativas nos resultados obtidos a partir de diferentes *datasets*, bem como de diferentes modelos de geração de vetores (ANTONIAK; MIMNO, 2018; SINOARA; ROSSI; REZENDE, 2016; SCHNABEL *et al.*, 2015; FIALHO; COHEUR; QUARESMA, 2020).

Uma dificuldade encontrada na avaliação de representações com relação à captura de similaridade semântica está na subjetividade do próprio conceito do que é similaridade (FARUQUI *et al.*, 2016). No âmbito computacional a similaridade é obtida por um cálculo matemático de uma medida escolhida, como a distância angular entre vetores, no caso de avaliações intrínsecas, ou avaliada por medidas de desempenho na tarefa, no caso de avaliações extrínsecas. Por outro lado, do ponto de vista da interpretação humana, que origina a anotação de *datasets*, esta questão enfrenta dificuldades de formalização. Na prática a similaridade é avaliada para casos específicos com critérios determinados na atividade de anotação do dados.

Outra questão no campo conceitual está no próprio uso da palavra *semântica*, a qual também remete à captura de significado tal como entendido sob o ponto de vista humano e coletivo e, portanto, subjetivo. A ideia de que resultados computacionais indiquem captura de semântica por parte de algoritmos tem sido questionada em PLN (BENDER; KOLLER, 2020) em uma discussão que não é nova e tem ramificações em outras áreas de conhecimento, como na Filosofia (SEARLE, 1980) e na Linguística (FIRTH, 1957).

Independente da interpretação dada aos resultados recentes na área, sob o olhar computacional a tarefa de STS se traduz no desenvolvimento e avaliação de algoritmos que tentam mapear pares de texto em valores ou classes de similaridade que sejam o mais próximo possível daqueles atribuídos por humanos. Neste âmbito, a área da Computação tem demonstrado resultados expressivos com impacto na indústria e com o envolvimento de grandes empresas no desenvolvimento de bibliotecas e algoritmos, como Google (ABADI *et al.*, 2016; DEVLIN *et al.*, 2019), Facebook (JOHNSON; DOUZE; JÉGOU, 2019; PASZKE *et al.*, 2019; LIU *et al.*, 2019) e OpenAI (RADFORD *et al.*, 2018).

A observação de que um mesmo treinamento de representações poderia ser aproveitado em diversas tarefas, ou seja, que há vantagens nos resultados do uso de modelos já treinados, culminou também em pré-treinamento de representações baseadas em quantidades cada vez maiores de texto. Adicionalmente, a percepção de que uma mesma palavra pode ter diferentes sentidos dependendo do seu uso, e que representações fixas para cada uma não comporta essas mudanças deram origem a métodos para gerar representações contextuais (QIU *et al.*, 2020).

Nos últimos anos o estabelecimento de modelos baseados em Transformers (VASWANI *et al.*, 2017) introduziu uma variedade de modelos pré-treinados que tem sido utilizado de forma bem-sucedida em diferentes tarefas, incluindo STS, como o BERT (DEVLIN *et al.*, 2019), RoBERTa (LIU *et al.*, 2019) e Distilbert (SANH *et al.*, 2019). Inicialmente os modelos eram focados no Inglês, um cenário que vem gradualmente mudando para o Português com o treinamento de modelos multilíngues e específicos para a língua, como o ptBERT (BERT-imbal) (SOUZA *et al.*, 2019), BERT para o Português.

Representações *embeddings* são largamente utilizadas como entrada para outros sistemas em PLN, servindo como recurso inicial. Podem ser utilizadas como *features* fixas na entrada de um algoritmo preditivo ou ter o modelo que a gera acoplado de forma interativa, permitindo que as representações sejam atualizadas para um fim específico em uma arquitetura inter-relacionada (estratégia conhecida como *fine-tuning*). Seja qual for o sistema utilizado para detecção de similaridade, quanto do desempenho na identificação de similaridade é referente ao pré-treinamento de *embeddings* e quanto proveniente da especialização dos parâmetros do algoritmo para o *dataset* da tarefa não fica bem definido. Este uso cada vez mais associado de representações como parte do algoritmo final utilizado em uma tarefa torna ainda mais desafiador o estudo do papel do pré-treinamento de representações no processamento de língua e o conhecimento de suas limitações, uma vez que tarefa alvo, *embeddings* pré-treinadas e

algoritmo preditivo geram os resultados como um todo.

Neste trabalho investigamos o papel das representações *embeddings* nos resultados da tarefa de detecção de similaridade sentencial com o uso de algoritmos supervisionados de aprendizado de máquina. Na parte teórica, fazemos uma breve discussão sobre o conceito de similaridade sob o ponto de vista humano e sob o ponto de vista da tarefa computacional. Na parte prática, o trabalho está dividido em duas grandes investigações. Na primeira, investigamos o impacto das representações nos resultados em comparação com outros parâmetros dos algoritmos utilizados. Na segunda parte focamos na tarefa de similaridade para o Português. Além disso, dada a variedade de modelos para geração de representações textuais usando *embeddings*, também testamos a combinação de representações para melhorar o desempenho na tarefa como alternativa a modelos com alta demanda computacional.

1.2 Objetivo e Questões de Pesquisa

O objetivo geral deste trabalho é investigar o papel de modelos de representações *embeddings* na tarefa de similaridade semântica sentencial, com maior foco para língua portuguesa. Para tanto, o trabalho foi dividido em duas frentes:

I) Na primeira, colocamos a prova se diferentes modelos de representações teriam impacto nos resultados na tarefa de STS em relação a outros parâmetros utilizados nos algoritmos de aprendizado (Capítulo 5), procurando responder as seguintes questões específicas:

   – O que impacta mais o resultado de classificação: o modelo de representação *embedding* escolhido ou os hiperparâmetros do classificador MLP utilizado para estimar a similaridade entre sentenças?

   – Qual a diferença de desempenho obtido entre (i) o uso de *embeddings* pré-treinadas como atributos de um classificador STS e (ii) o ajuste de *embeddings* de forma integrada na tarefa de STS (e.g. *fine-tuning*)?

II) Na segunda, focamos na tarefa aplicada ao Português, propondo uma arquitetura para combinar modelos de representações. Utilizamos modelos de representações sentenciais baseada em arquitetura de redes siamesas e avaliamos o ganho de desempenho em relação ao uso de modelos sozinhos (Capítulo 6), considerando a seguinte questão de pesquisa:

   – A combinação de modelos de *embeddings* pré-treinados pode ser utilizada para melhorar o desempenho na tarefa de detecção de similaridade sentencial no Português em relação à utilização de modelos sozinhos?

1.3 Principais contribuições e resultados

Avaliação do Impacto da escolha do modelo de *embeddings* para a tarefa de STS: Existem diferentes técnicas para gerar representações. O uso de modelos de *embeddings* na detecção de similaridade sentencial como entrada de algoritmos preditivos supervisionados gera resultados que não discriminam o papel das representações em relação aos hiperparâmetros dos algoritmos, ficando a questão se de fato modelos de representações distintos tem impacto no resultado ou o ajuste de hiperparâmetros compensaria esta distinção. Neste trabalho testamos modelos fixos, um contextual e um enriquecido para múltiplas configurações de hiperparâmetros. Nossos resultados indicam que o modelo de representação escolhido é o fator mais impactante na faixa de predição em relação a todos os outros parâmetros testados.

Comparação entre representação fixa *vs* ajustada: O BERT é um dos principais modelos de representação pré-treinada para similaridade sentencial, sendo originalmente pré-treinado em *corpora* com mais de 3 bilhões de palavras. Comparamos o resultado da utilização de suas representações fixas em relação ao ajuste fino para a tarefa de STS. Ajustado para a tarefa (*fine-tuned)* o BERT apresentou o melhor resultado no *dataset* de perguntas QQB em relação aos modelos testados, contudo, como representação fixa, as *embeddings* do BERT desempenharam pior do que as representações geradas diretamente no QQP por um modelo mais simples, o Skip-gram do Word2Vec. O domínio do texto, portanto, foi mais relevante do que complexidade do modelo e tamanho do *corpus* de pré-treino, evidenciando a importância da atualização das representações no contexto específico trabalhado.

Combinação de representação para STS no Português: Propusemos uma arquitetura simples para combinação de representações pré-treinadas. A avaliação dos resultados corrobora a ideia de que diferentes modelos de *embeddings* codificam características distintas do texto. Adicionalmente os resultados mostraram que a combinação de representações através de uma arquitetura simples pode melhorar os resultados em relação ao uso de modelos de representação individualmente. Deixamos nosso modelo de combinação de *embeddings* disponível como ferramenta que pode ser importada e testada como desejado pelo usuário com diferentes modelos pré-treinados do SentenceTransformers.

Generalização dos resultados para além do *dataset* original em STS no Português: Avaliamos a capacidade de generalização dos modelos para além do *dataset* original, comparando o melhor modelo de combinação de *embedding* e o ptBERT fora do domínio de treinamento. Para tanto, treinamos no *dataset* ASSIN1 e testamos no *dataset* ASSIN2, bem como o contrário. Embora o ptBERT seja o estado da arte para os *datasets* separadamente, os resultados apontaram que a diferença de desempenho entre este e o modelo de

combinação de *embeddings* diminuiu quando aplicados fora do *dataset* original, indicando haver superespecialização do ptBERT para o *dataset*.

Avaliação da predição para além das medidas típicas: Complementarmente aos resultados dados pelo cálculo de MSE e correlações de *Pearson* e *Spearman*, estendemos o modelo e explicabilidade LIME (RIBEIRO; SINGH; GUESTRIN, 2016) para receber modelos preditivos cujo *input* são dois textos, possibilitando assim seu uso para modelos de similaridade sentencial. Utilizamos o modelos em algumas instâncias para avaliar o impacto de *tokens* na predição dos valores de similaridade. A comparação entre modelos nas instâncias revelou, entre outras coisas, que de forma geral a polaridade dos *tokens* individualmente nas predições não explicam as diferenças de resultados entre modelos.

1.4 Organização do texto

Capítulo 1: Apresentamos brevemente o contexto, os objetivos e as principais contribuições do trabalho.
Capítulo 2: Abordamos a tarefa de similaridade sentencial e sua relação com representações *embeddings* e as principais medidas utilizadas na prática para cálculo de similaridade.
Capítulo 3: Abordamos o problema conceitual de similaridade e sua relação com outros conceitos subjetivos da interpretação de texto. Adicionalmente explicamos brevemente alguns dos conceitos de linguística que fazemos uso ao longo do trabalho.
Capítulo 4: Neste capítulo o foco é nas representações *embedding*. Apresentando um breve histórico seguido de uma visão geral de modelos fixos e contextuais de representação.
Capítulo 5: Avaliamos o impacto de modelos de *embeddings* nos resultados de similaridade semântica sentencial em relação aos hiperparâmetros do algoritmo preditivo.
Capítulo 6: Avaliamos modelos de *embeddings* sentenciais e os potenciais ganhos da combinação de modelos em relação ao uso de modelos sozinhos na tarefa de similaridade semântica sentencial no Português.
Capítulo 7: Concluímos o trabalho indicando as considerações gerais, principais conclusões e trabalhos futuros.

CAPÍTULO

2

TAREFA DE SIMILARIDADE SEMÂNTICA SENTENCIAL

Neste capítulo abordamos a tarefa de similaridade semântica sentencial e sua relação com as representações *embeddings*.

2.1 Visão geral da tarefa

A tarefa de reconhecimento de similaridade semântica vem sendo explorada desde os anos 70, com aplicação em diversas tarefas, como sistemas de tradução automática, sumarização de texto e sistemas de perguntas e respostas (TURNEY; PANTEL, 2010).

Como tarefa de PLN, o reconhecimento de similaridade consiste em, dadas duas palavras, sentenças ou documentos, avaliar o grau de similaridade entre elas (CER *et al.*, 2017), podendo ser modelada de forma binária (dois textos são ou não similares) em classes de semelhança pré-definidas, ou de forma contínua (a partir de uma escala pré definida são atribuídos valores tanto maiores quanto mais similares forem considerados os textos).

Sob o ponto de vista computacional o grau de similaridade entre dois textos é dado por um cálculo, uma medida objetiva como uma fórmula matemática pré-estabelecida ou pelo resultado da predição de um algoritmo de aprendizado de máquina. Por outro lado, sob um ponto de vista humano, similaridade não é um conceito universal e dois textos podem ser ou não similares dependendo do critério que se escolhe para avaliá-los, como abordaremos com mais detalhes no Capítulo 3.

No âmbito da tarefa computacional, o ponto de vista humano fica a critério da anotação dos dados. A cada par de texto são atribuídos valores que representam o entendimento do anotador sobre sua similaridade a partir de *guidelines* pré-estabelecidos. É o senso humano dado pela anotação dos dados que norteia a tarefa. O objetivo da tarefa de detecção de similaridade é

aproximar os resultados computacionais da interpretação humana representada pela anotação.

A seguir mostramos alguns exemplos de anotação de similaridade em sentenças, também chamada similaridade semântica textual (STS). Na Tabela 1 as sentenças foram retiradas do SICK-BR (REAL *et al.*, 2018), *corpus* anotado para similaridade e inferência. A similaridade foi manualmente anotada variando em uma escala de 1 até 5, sendo consideradas mais similares pelos anotadores quanto maior o valor. Na Tabela 2, traduzimos do Inglês alguns exemplos do *Quora Question Pair Benchmark* (QQP), no qual perguntas são avaliadas para similaridade de forma binária, sendo 1 para equivalentes e 0 para distintas.

Tabela 1 – Exemplos de anotação de similaridade no *corpus* SICK-BR

par	sentença	texto	similaridade
1	A B	Um ciclista solitário está pulando no ar Um ciclista está pulando no ar, sozinho	5,0
2	A B	Dois grupos de pessoas estão jogando futebol Dois times estão competindo em uma partida de futebol	4,7
3	A B	Um homem está pulando em uma piscina vazia Um ciclista está pulando no ar, sozinho	1,6

Fonte: Tabela de autoria própria com dados extraídos do *dataset*

Tabela 2 – Exemplos de anotação de similaridade do QQP

par	questão	texto	similaridade
1	A B	Como perco peso rapidamente? Qual a melhor forma de perder peso rapidamente?	1
2	A B	Como você se torna um controlador de tráfico aéreo? Como o tráfico aéreo é controlado?	0
3	A B	Qual é o QI médio de um ser humano? Qual é o QI de um ser humano regular?	1
4	A B	Por que alguém quereria um panda vermelho como pet? Pode-se ter um panda vermelho como pet?	0

Fonte: Tabela e tradução própria a partir de dados do *dataset*

2.2 Similaridade e representações

O uso de representações vetoriais em tarefas de similaridade semântica inicialmente encontrou motivação na ideia de que termos semanticamente relacionados ocorreriam acompanhados dos mesmos termos, característica que em teoria tornaria possível codificar relações de sentido através da distribuição do texto (TURNEY; PANTEL, 2010). Segundo esta ideia, as palavras *cachorro* e *cão* por exemplo seriam identificadas como semanticamente próximas pelo contexto no texto (aqui entendido como as palavras com as quais coocorrem) em que aparecem:

(1) O *cachorro* comeu a ração.
O *cão* comeu a ração.

(2) O *cachorro* latiu a noite inteira.
O *cão* latiu a noite inteira.

(3) Meu amigo tem um *cachorro* farejador.
Meu amigo tem um *cão* farejador.

A ideia é que este contexto seria capturado não por um único exemplo, mas trazido pela observação sistemática e extensiva de uso em diversas sentenças. Assim, embora na sentença (1) o termo *cachorro* pudesse ser substituído por *gato*, a consideração dos diversos usos mitigaria esta questão, como em (2) em que seria mais difícil *gato* vir associado ao verbo *latir*. Esse processo seria iterado para um número gigante de sentenças e através da coocorrência de cada palavra com outras seriam determinadas palavras próximas semanticamente.

A hipótese de correlação entre a distribuição das palavras e seu significado ficou conhecida como *hipótese distribucional* e é frequentemente relacionada aos linguistas Zelling Harris, em particular ao trabalho *Distributional Structures* (1954). Neste trabalho Harris (1954) questiona a aleatoriedade da língua, propondo que esta é dotada de estrutura, sendo passível de ser analisada pela descoberta de padrões e ocorrências. Contudo, sua proposta está no contexto do estudo linguístico, como oposição a ideia de não ser possível fazer um estudo sistemático da língua. Quanto a relação com semântica, o próprio autor indica que a distinção entre estrutura distribucional e significado não é clara, não sendo atribuição de significado proveniente unicamente de propriedades da língua, mas uma característica geral da atividade humana (HARRIS, 1954). Neste sentido, contexto de ocorrência é uma pista mas não determina ou equivale a semântica da palavra.

No campo filosófico a questão que fica é até que ponto é possível acessar de forma sistemática relações de significado utilizando somente a distribuição dos elementos contidos no texto. No campo prático, até que ponto representações baseadas em distribuição do texto servem para a realização automatizada de tarefas de processamento de língua, independente se carregam ou não relações de significado de forma sistêmica.

2.3 Datasets de STS para o Português

Como recurso de *corpus* anotados no Português para similaridade semântica textual, destacamos os *datasets* ASSIN1 e ASSIN2. Utilizamos estes *datasets* na avaliação de combinação de representações no Capítulo 6.

2.3.1 Dataset ASSIN1

Utilizado na *sharetask* ocorrida em paralelo com a conferência PROPOR2016 [1], o ASSIN1 (FONSECA, 2018) é um *dataset* constituído por 10.000 pares de sentenças em Português anotadas para duas tarefas, similaridade e inferência e previamente dividido em treino (6.000), validação (1.000) e teste (3.000). As sentenças foram selecionadas a partir de textos jornalísticos, contendo duas variantes da língua, a variantes brasileira (PB) e variante europeia (PE). A anotação para a tarefa de inferência foi feita para três classes: implicação, paráfrase e neutro. Já os valores anotados para similaridade semântica variam de 1 a 5, segundo diretrizes de anotação em Fonseca *et al.* (2016). Quanto mais alto o valor maior a similaridade, sendo 1 para sentenças com sentidos completamente diferentes em que não é claro se tratarem do mesmo fato e 5 para sentenças com sentido considerados quase iguais. Alguns exemplos são mostrados na Tabela 4 A distribuição de sentenças por faixa de similaridade para as duas variantes da língua pode ser observada na Tabela 3:

Tabela 3 – Estatísticas de similaridade do ASSIN

Similaridade	**PB**	**PE**	**Total**
4,0 – 5,00	1.074	1.336	2.410
3,0 – 3,75	1.591	1.281	2.872
2,0 – 2,75	1.986	1.828	3.814
1,0 – 1,75	349	555	904

Fonte: (FONSECA, 2018)

Tabela 4 – Exemplos pares de sentenças do ASSIN1

Par de sentença	Rótulo
A faixa etária que mais foi presa é a de 18 a 24 anos. O relatório aponta que 13 Estados tiveram crescimento acima da marca nacional.	1,0
Já no dia seguinte, esse número subiu para 63.455. O contribuinte tem até o dia 30 de abril para fazer a declaração.	1,0
Lançamento nos cinemas brasileiros será no dia 21 de abril de 2016. O filme tem lançamento marcado no Brasil para 21 de abril de 2016.	5,0
O valor foi enviado ao exterior por meio de empresas offshores localizadas no Uruguai e na Suíça. O dinheiro tinha sido enviado ao exterior através de empresas offshores no Uruguai e Suíça.	5,0

Hartmann *et al.* (2017) utiliza o *corpus* ASSIN1 para avaliar extrinsecamente modelos de *embeddings* na tarefa de similaridade. Na sua avaliação faz uso de quatro modelos, a saber GloVe, Word2Vec, Wang2Vec e fastText, utilizando cinco dimensões vetoriais (50, 100, 300, 600, 1000). Os resultados foram comparados considerando a correlação de Pearson (ρ) e o

[1] http://propor2016.di.fc.ul.pt/

erro quadrático médio (MSE) [2]. O melhor resultado na tarefa para o Português europeu e para o Português brasileiro foi obtido respectivamente pelo CBOW e pelo Skip-gram do modelo Word2Vec com dimensão 1000. Os vetores alcançaram os valores de ρ e MSE de 0,6 e 0,49 para o Português brasileiro e 0,55 e 0,86 para o Português europeu. Silva *et al.* (2019) investigou o uso de representações vetoriais obtidas por métodos que não fixam um único sentido para cada palavra aplicados ao Português (brasileiro e europeu) para resolver o problema da ambiguidade lexical. Mais especificamente, o algoritmo Multi-Sense Skip-Gram (MSSG) (NEELAKANTAN *et al.*, 2014) e o Sense2Vec (TRASK; MICHALAK; LIU, 2015). Na sua investigação, avalia os vetores intrinsecamente na tarefa de similaridade sintática e semântica e extrinsecamente tarefas de similaridade semântica e desambiguação lexical de sentidos. A Figura 1 mostra seus resultados no ASSIN1 para a tarefa de similaridade semântica. O modelo Sense2Vec apresentou o melhor desempenho entre os modelos testados com 0,57 de correlação de Pearson e 0,51 de MSE.

Figura 1 – Resultados de modelos de vetores na tarefa de similaridade semântica

Embedding	**Tipo**	**Dim.**	**PT-BR**		**PT-EU**	
			ρ (↑)	**MSE** (↓)	ρ (↑)	**MSE** (↓)
Word2Vec	word	300	0,55	0,53	0,55	0,84
GloVe	word	300	0,46	0,60	0,47	0,93
FastText	word	300	0,53	0,55	0,51	0,89
MSSG	sense	300	0,39	0,65	0,35	1,04
Sense2Vec	sense	300	**0,57**	**0,51**	0,53	0,87

Valores para o coeficiente de Pearson (ρ) e MSE obtidos na tarefa de similaridade semântica no *corpus* ASSIN. A setas indicam se menor (↓) ou se maior (↑) é melhor. Fonte: (SILVA *et al.*, 2019)

2.3.2 Dataset ASSIN2

O ASSIN2 (REAL; FONSECA; OLIVEIRA, 2020) foi apresentado em 2019, no *Symposium in Information and Human Language Technology (STIL)*. O *dataset* é constituído de 9.948 pares de sentenças, dividido em treino (6.500), validação (500) e teste (2.448), manualmente anotado para e inferência e similaridade por um time de anotadores. O conteúdo foi baseado no SICK-BR (REAL *et al.*, 2018), uma tradução e adaptação para o Português do SICK (MARELLI *et al.*, 2014), corpus originalmente em Inglês usado no SemEval 2014. A anotação de inferência conta com dois rótulos, implicação e não implicação (*entailment* e *non-entailment*) enquanto a anotação de similaridade, assim como no ASSIN, são valores que variam de 1 a 5, dados pela média dos scores atribuídos pelos anotadores com base nas suas avaliações de similaridade de sentido. Alguns exemplos de pares de sentenças são mostrados na Tabela 5.

[2] MSE do Inglês *Mean Squared Error*

Tabela 5 – Exemplos pares de sentenças do ASSIN2

Par de sentença	Rótulo
Um piano está sendo tocado por uma pessoa Uma pessoa está fatiando uma batata	1,0
Um bebê está engatinhando feliz Um gato está andando no chão de tacos	1,0
Um cara está agachado no arbusto e tirando uma fotografia Um cara está agachado no arbusto e está tirando uma foto	5,0
Uma senhora está cortando brócolis Brócolis estão sendo cortados por uma senhora	5,0

2.4 Medidas de similaridade

Tarefas de similaridade na área de PLN envolvem a comparação da percepção humana, no caso, dada pela anotação dos dados segundo um critério previamente definido, e um valor obtido numericamente por um método escolhido de similaridade. As medidas de similaridade podem ser baseadas diretamente no texto, comparando-se por exemplo cadeias de caracteres, ou baseada em vetores que representam o texto. Para avaliações intrínsecas de *embeddings*, a comparação é feita entre dados anotados e o cálculo geralmente da similaridade de cossenos. A seguir apresentamos algumas medidas de similaridade utilizadas na área de PLN.

- **Similaridade de cossenos:** Dados dois vetores não nulos $\vec{w}_1$ e $\vec{w}_2$, a similaridade de cossenos é dada pelo cosseno do ângulo formado entre eles, definido como:

$$sim(\vec{w}_1, \vec{w}_2) = \frac{\vec{w}_1 \cdot \vec{w}_2}{\|\vec{w}_1\|\|\vec{w}_2\|}$$

  Uma observação importante no uso de similaridade de cossenos é que esta não é uma distância no sentido matemático, pois embora exista simetria com relação aos termos (o cosseno do ângulo entre dois vetores $\vec{u}$ e $\vec{v}$ é igual ao cosseno entre $\vec{v}$ e $\vec{u}$), esta similaridade não possui duas outras propriedades relevantes para ser uma métrica: não obedece a desigualdade triangular e apresenta valores negativos (cosseno varia entre -1 e 1).

  Para realizar comparações em que estas propriedades sejam mantidas, pode ser utilizada a similaridade angular entre os vetores, na qual os valores são dados em uma escala de 0 a 1 e obtida a partir da similaridade de cossenos pelas expressões:

$$\theta(\vec{w}_1, \vec{w}_2) = \frac{\cos^{-1}(sim(\vec{w}_1, \vec{w}_2))}{\pi}$$

$$sim\_ang(\vec{w}_1, \vec{w}_2) = 1 - \theta(\vec{w}_1, \vec{w}_2)$$

- **Distância de Manhattan:** O cálculo desta distância também é feito baseado em vetores. Sejam dois vetores $p = (p_1, p_2, p_3, ..., p_n)$ e $q = (q_1, q_2, q_3, ..., q_n)$, a distância de Manhattan entre eles é dada por:

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- **Similaridade de Jaccard:** Dados dois conjuntos A e B, o índice de Jaccard é dado pela cardinalidade da intersecção entre eles dividida pela cardinalidade da união, como mostrado mais abaixo. A medida depende de como os conjuntos são definidos. Para o processamento de similaridade de suas sentenças, por exemplo, os conjuntos podem ser dados pelo vocabulário presente em cada uma.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- **Distância de Levenshtein:** Dadas duas sequências de caracteres, a distância entre elas é medida pelo número mínimo de operações (inserção, substituição e deleção) necessárias para converter uma sequência na outra. A proporção entre a distância e o comprimento da cadeia é considerada a similaridade entre elas (ELAVARASI; AKILANDESWARI; MENAGA, 2014).

2.5 Métricas de avaliação de resultado de predição

A avaliação de sistemas de predição de similaridade semântica textual é feita por métricas tipicamente usadas em problemas de aprendizado supervisionado. Seguem abaixo algumas destas métricas.

2.5.1 Classificação

- **Acurácia :** A acurácia é dada pela proporção do número de acertos na predição em relação a todas as predições realizadas e varia de 0 a 1. Geralmente é utilizada em conjunto com outras medidas, uma vez que a proporção do número de acertos não é por si só um indicativo do desempenho em todos os casos. No caso de dados muito desbalanceados para uma classe, por exemplo, se o sistema sempre 'chutar' a classe com mais instâncias, a acurácia será equivalente a quantidade da classe predominante. No caso, para ilustrar, se 90% dos dados forem desta classe, a acurácia será 90% sem que de fato o sistema seja capaz de discriminar as classes existentes.

- **Precisão :** A precisão indica quanto de uma classe o sistema acertou em relação a todas as predições que o sistema fez para aquela classe. Seja tp as predições em que o sistema acertou para a classe (verdadeiros positivos) e fp as predições que o sistema errou

classificando como sendo da classe (falsos positivos) a precisão é dada por:

$$precisao = \frac{tp}{tp + fp}$$

- **Revocação :** A revocação é dada pela proporção em que o sistema acertou a predição de uma determinada classe em relação a todos os dados que de fato são desta classe. Seja tp as predições em que o sistema acertou para a classe (verdadeiros positivos) e fn as predições que o sistema errou em não classificar sendo desta classe (falsos negativos) a revoação é dada por:

$$revocacao = \frac{tp}{tp + fn}$$

- **Medida F1:** A medida F1, ou F1-score, considera tanto a precisão quanto a revocação do sistema. É calculada pela média harmônica entre ambas:

$$F1 = 2 \cdot \frac{revocacao \cdot precisao}{revocacao + precisao}$$

2.5.2 Regressão

- **Correlação de *Pearson* :** O coeficiente de correlação de *Pearson* indica o grau de correlação linear entre duas variáveis. Varia entre -1 e 1, sendo quanto mais próximo de 0 o valor, positivamente ou negativamente, menos linearmente correlacionadas são as variáveis. Seja $cov(X,Y)$ a covariância entre as duas variáveis X e Y, e σ_X e σ_Y o desvio padrão de cada uma respectivamente, o coeficiente de *Pearson* $\rho_{X,Y}$ entre X e Y é calculado da seguinte forma:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

- **Correlação de Spearman :** A correlação de *Spearman* mede a correlação entre duas variáveis considerando quanto a relação entre elas pode ser descrita por uma função monotônica. É aplicada aos postos das observações das variáveis e varia de -1 a 1. Seja d_i a diferença entre os postos da observação i, o coeficiente é dado por:

$$r = 1 - \frac{6 \sum_{i=1}^{n} {d_i}^2}{n(n^2 - 1)}$$

- **Erro quadrático médio :** O erro quadrático médio é calculado pela soma da diferença entre a predição de cada instância e a média dos dados ao quadrado dividido pela quantidade de instâncias, como indicado a seguir:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

CAPÍTULO

3

FUNDAMENTAÇÃO TEÓRICA LINGUÍSTICA

Um desafio para o desenvolvimento de recursos para a tarefa de similaridade semântica sentencial é lidar com a questão conceitual envolvida na interpretação do que seria o significado e o que seria similaridade. Neste capítulo abordamos o problema conceitual de similaridade, expondo a relação com o conceito de significado e a dificuldade envolvida na sua formulação. Indicamos dois fatores relevantes na estruturação do problema: a determinação de um critério de similaridade e a representação dos dados.

Ao longo do trabalho e na análise dos resultados fazemos uso de alguns termos como "gramática", "sintáxe", "norma culta" e "gênero textual", os quais possuem sentidos específicos que podem não ser familiares ao leitor. Apresentamos também uma breve base teórica no sentido de esclarecer estes pontos.

3.1 A natureza do *significado*

O conteúdo gerado em língua natural é feito por humanos para humanos e por isso não nasce com a preocupação de ser estruturado de forma que um computador possa processar. Língua natural, como Inglês e Português, assim chamada como forma de distinção entre outras línguas criadas, como C e Python utilizados na área de programação, não é adquirida a partir do estudo de regras bem definidas e estabelecidas; uma criança não recebe uma norma para estudar antes de falar. A compilação formal de regras, como as encontradas em gramáticas normativas ou descritivas, tem origem na observação da língua já em curso.

Essa condição relativa à língua suscita inúmeros questionamentos no que se refere a sua aquisição por seus falantes e a como se dá a construção do significado das expressões por eles utilizadas. Questões como se somos previamente programados para a faculdade da linguagem (CHOMSKY, 2006), se o conhecimento linguístico é armazenado de forma distribuída ou local no cérebro (KEMMERER, 2014), se a língua pode ser interpretada como uma nomenclatura ou

tem relação direta com a organização do pensamento (SAUSSURE, 2008), foram exploradas em diferentes áreas de conhecimento. Um dos consensos entre diversas linhas de pesquisa é que o ser humano aprende pela experiência, e sua experiência não se limita ao som ou à escrita, tendo como participante todos seus sentidos físicos, além da constante interação com outros humanos. Embora um computador não necessariamente precise realizar uma tarefa pelos mesmos mecanismos que um ser humano realiza, a percepção de que se tenta modelar computacionalmente algo que é adquirido pelos diferentes sentidos pode dar uma dimensão da dificuldade envolvida.

No campo do processamento de língua natural essa discussão é particularmente relevante se considerado que diversas tarefas que se deseja automatizar, quando realizadas por seres humanos, dependem de interpretação subjetiva. É esta interpretação subjetiva que se tenta mimetizar em tarefas que envolvem interpretação de texto, e que torna um desafio transpor o nível da forma do que está escrito para resultados condizentes com um nível do significado do que está escrito.

3.1.1 Significado e semântica

O termo *semântica* quando associado a língua natural pode ser entendido sob pelo menos dois pontos de vista. A forma mais corriqueira, usada na comunicação em geral, remete a uma noção de interpretação, equivalente a significado ou sentido de algo dito ou escrito, ainda que sejam subjetivos. “Esta não é a *semântica* do que ele disse”. A outra forma é como área de estudo contemplado dentro do campo da Linguística. Semântica, como área de conhecimento, é estabelecida como o “estudo do significado das expressões das línguas naturais” (CHIERCHIA, 2003). Para ambas interpretações, o termo *semântica* está atrelado ao conceito de significado.

O uso da palavra *significado* traz por si só um impasse. Por um lado, falar que duas coisas são similares e que há uma demanda em se captar níveis semânticos de texto para a automatização de tarefas de PLN é, em alguma medida, fazer referência a algum conceito de significado. O que seria captar níveis semânticos se não captar significado do que é expressado através da língua? Por outro, fica a espinhosa pergunta do que seria *significado*.

Tentar estabelecer um conceito para *significado* é tema discutido em várias linhas de estudo, da Filosofia à Neurociência. Não nos aprofundaremos nesta questão, mas gostaríamos de ressaltar uma visão compartilhada em estudos semânticos de que *significado* é uma relação entre uma expressão e algo não-linguístico (NETO, 2003). Este algo não-linguístico pode ser uma imagem mental, um objeto no mundo, um sentimento, uma situação de uso, entre outras coisas consideradas por diferentes vertentes da Semântica.

Entre suas vertentes estão três abordagens para a questão deste estudo: uma abordagem referencial, um abordagem pragmática e uma abordagem mentalista. Comentamos anteriormente que o significado no contexto de diversos estudos semânticos é estabelecido por uma relação de

expressões com algo não-linguístico. Uma diferença importante entre as diversas abordagens está exatamente em qual algo "não-linguístico" focam suas relações.

Segundo uma tradição referencial, expressões denotam algo no mundo, um referente (CANÇADO, 2012) (p.75). A palavra *gato*, por exemplo, tem como referente indivíduos de uma classe, a classe de gatos. Já um nome próprio, se refere a um indivíduo em particular. A referência contrapõem-se ao sentido, que segundo Frege (1948), precursor desta distinção, seria o modo como ela é apresentada. De acordo com sua visão, uma mesma referência pode ter múltiplos sentidos, diversas formas de ser exposta. O planeta Vênus, por exemplo, pode ser apresentado como sendo a *estrela da manhã* ou a *estrela da tarde*, dois sentidos com um único referente. No âmbito das sentenças, referências associam-se ao seu valor de verdade, ou seja, dado um contexto, uma sentença como "O João é brasileiro" pode ter valor verdadeiro ou falso dependendo da veracidade ou falsidade dessa sentença no mundo (CANÇADO, 2012). Neste âmbito, há um interesse em se estudar quais as condições que tornam uma afirmação verdadeira, suas condições de verdade. Assim, para "O João é brasileiro" ter valor verdadeiro, pressupõe-se que existe um João determinado, que existe um conjunto de brasileiros e este João pertence a este conjunto.

Uma abordagem pragmática relaciona as expressões da língua ao seu uso em situações comunicativas concretas. Utilizando o exemplo em Cançado (2012) (p. 25) , tomemos a sentença "A porta está aberta". Imagine esta sentença falada por um professor dentro de uma classe de aula para um aluno que está do lado de fora olhando para dentro. Isso pode ser interpretado como um convite para que o aluno entre. A mesma sentença proferida pelo professor a um aluno que atrapalha sua aula pode ser uma repreensão ou expulsão do aluno do local. Em ambas as situações o significado não é a relação com o estado da porta no mundo, estar aberta ou fechada, ou com seu valor de verdade, se a porta está de fato aberta, mas com a intenção que atribuímos ao ato da fala. Esta interpretação de intenção advém do conhecimento de mundo que possuímos aplicado a uma situação específica.

Uma abordagem representacional associa expressões linguísticas com construtos mentais que de alguma maneira estão no falante (CANÇADO; AMARAL, 2017). Uma das dificuldades envolvidas neste tratamento é o acesso a estes conceitos, uma vez que o funcionamento daquilo que chamamos de mente não é completamente conhecido. Ainda assim, trabalhos que consideram o papel de representações mentais na construção do conceito de significado tiveram grande impacto nas observações linguísticas. Neste âmbito, destacam-se, entre outros, nomes como Charles J. Fillmore, Ray Jackendoff e Ronald Langacker e George Lakoff.

3.2 Problema conceitual de similaridade

Sob a ótica do indivíduo, a avaliação se duas coisas são similares é algo realizado naturalmente quando se necessita agrupar ou organizar objetos. Tversky (1977) conceitua

similaridade como "(...) uma organização de princípios pelo qual indivíduos classificam objetos, formam conceitos e fazem generalizações.". Mas seriam estes princípios uma propriedade inerente das coisas? Imagine que pedimos para que alguém agrupe as palavras abaixo por similaridade:

$$(a)cachorro, (b)avestruz, (c)coruja, (d)baleia, (e)morcego, (f)tubarão, (g)pinguim$$

Diversas formas de se realizar um agrupamento seria possível considerando-se diferentes características, como nos exemplos abaixo:

(1) Mamíferos *vs* aves *vs* peixes
A = {*cachorro*, *morcego*, *baleia*}
B = {*avestruz*, *pinguim*, *coruja*}
C = {*tubarão*}

(2) Voa ou não voa
A = {*cachorro*, *baleia*, *avestruz*, *pinguim*, *tubarão*}
B = {*coruja*, *morcego*}

(3) Ambiente aquático, terrestre ou misto
A = {*tubarão*, *baleia*},
B = {*cachorro*, *morcego*, *avestruz*, *coruja*}
C = {*pinguim*}

As diversas possibilidades sugerem que aquilo que consideramos por similaridade não é universal e intrínseco ao mundo como o vemos. Coisas são similares sob um ponto de vista, um critério, que determina o que se tem interesse que seja observado. Um indicativo disso é que mudando-se o padrão a ser observado, ocorre mudança no resultado, como exemplificado comparando-se (1) (2) e (3), em que para cada critério utilizado, obteve-se um agrupamento diferente.

Uma questão que pode surgir é se não seria a similaridade das palavras determinada por algum sentido usual que atribuímos e elas. Porém, imagine que em (d) ao invés de *baleia* a palavra fosse *gato*. Ainda que *gato* e *baleia* não sejam considerados como tendo o mesmo sentido usual, o resultado em (1) e (2) seria mantido, com *gato* junto com outros mamíferos, porém em (3), ocorreria alteração nos grupos, ficando *gato* com os animais terrestres. Verificamos, portanto, que o que quer que diferencie o sentido de *gato* e de *baleia*, isso não é relevante para separá-los sob a ótica de qualquer critério (em (1) e (2) se mantiveram no mesmo grupo). O critério seleciona um conjunto de atributos relevantes a serem observados.

Mas será que basta escolher um critério pelo qual se consideraria as coisas similares para termos o problema de similaridade estruturado? Observemos outro exemplo. Imagine que

escolhemos o critério de agrupar as palavras abaixo pela letra inicial, deixando juntas palavras com a mesma letra. Mas para isso temos três opções distintas de representações das palavras: (4), (5) e (6). Na primeira os dados são as próprias palavras, na segunda, ao invés de palavras tal como escrita é fornecido sua típica classe lexical, e na terceira é fornecido apenas a primeira sílaba, como mostrado abaixo.

$$(a)panela, (b)prato, (c)colher, (d)concha, (d)faca, (e)frigideira$$

(4) Representação: completa
(a) *panela*, (b) *prato*, (c) *colher*, (d) *concha*, (e) *faca*, (f) *frigideira*

(5) Representação: classe lexical
(a) substantivo, (b) substantivo, (c) substantivo, (d) substantivo, (e) substantivo (f) substantivo

(6) Representação: primeira sílaba
(a) pa, (b) pra, (c) co, (d) con, (e) fa (f) fri

Neste caso, embora o critério seja o mesmo, "agrupar pela primeira letra", a representação em (5) não possibilita que se "enxergue" a distinção de letra inicial para que o critério seja aplicado. Note-se que não é apenas porque a palavra não foi fornecida, uma vez que em (6), mesmo sem se conhecer a palavra completa, seria possível alcançar o mesmo resultado do que utilizando as palavras em (4). Um problema semelhante encontraríamos se pedíssemos a alguém que só enxerga preto e branco para agrupar objetos coloridos por cor. Portanto, se o critério "olha" alguns atributos sob o qual definirá um agrupamento por similaridade, igualmente relevante são quais atributos estão disponíveis para serem observados.

Na área de PLN, a transposição do texto em vetores numéricos é que faz um recorte que determina estes atributos. Diversas representações realizam esta transposição de forma automatizada, sem que necessariamente se observe a tarefa objetivo. O resultado pode ser a obtenção de representações que, assim como em (b), deixam de fora características relevantes para aquilo que se deseja observar. Da mesma forma, a ausência de critério bem estabelecido pode levar à confusão do que se deseja identificar como similaridade.

A identificação de similaridade fica, portanto, como um problema estruturado em função de dois fatores: um critério escolhido e atributos disponíveis. Na prática, o critério pode ser definido por inúmeros tipos de relações, dependendo do que se pretende. No plano semântico, por exemplo, podem ser consideradas relações de sinonímia, mesmo campo semântico, mesmo referente, paráfrases, ou ainda serem consideradas características como mesma utilidade no mundo ou sentenças que abordam um mesmo assunto. A definição de critérios pode ser difícil dependendo do que se deseja avaliar.

Para a tarefa computacional os atributos são materializados pela representação numérica do texto e podem não ser objetivos ou transparentes do ponto de vista da intuição ou da interpretação. Existem múltiplos modelos que mapeiam o texto em vetores, os quais abordamos no Capítulo 4.

3.3 Alguns conceitos da Linguística

Nesta seção abordamos alguns conceitos provenientes do campo de estudo da Linguística.

3.3.1 Palavra, sentença e texto

Como observação inicial chamamos a atenção para os termos *palavra*, *sentença* e *texto*. Esses termos podem ser interpretados de diversas formas. *Palavra*, por exemplo, pode fazer referência a um *token* específico ou a uma classe de *tokens*, chamadas de *types*. O texto " a filha do meu amigo veio brincar com a minha filha" contém 11 *tokens* e 9 *types*, no entanto usualmente um falante de Português pode ora falar que há 11 palavras na sentença e ora que a palavra *filha* aparece duas vezes. Neste trabalho, como usualmente ocorre no Português, faremos um uso não rigoroso destes três termo e algumas vezes usaremos *termo* como sinônimo de *palavra*.

Já o termo *sentença* é de difícil definição precisa. Pode ser referenciado em gramáticas normativas como enunciado de sentido completo ou trechos linguísticos gramaticalmente vinculados. Para o escopo deste trabalho, aplicamos este termo como uma sequência de *tokens* com ou sem ponto final. O termo *texto* usaremos para qualquer tamanho de sequência de *tokens*, não necessariamente sendo uma redação, ou um documento.

3.3.2 Estrutura versus sentido

Podemos analisar um texto sob o ponto de vista do sentido ou sob o ponto de vista de uma estrutura utilizada para sua formação. Chomsky (1957) faz essa distinção quando apresenta que um texto pode ser gramaticalmente bem construído e ser aparentemente desprovido de sentido: "A noção *gramatical* não pode ser confundida com as noções *dotado de sentido* ou *significativo* em qualquer sentido semântico." (p.19). Para ilustrar esta diferença apresentamos o exemplo abaixo:

(1) A menina foi ao banco tirar dinheiro.
(2) A flor espacial comeu vegetais na sala da cadeira.

Ambas as sentenças são reconhecidas como gramaticalmente aceitas no Português, no entanto, a dificuldade de interpretação do que significa cada uma pode variar bastante. Acreditamos que, sem um contexto específico que torne a segunda coerente, é mais difícil atribuir sentido a ela do que à primeira.

A noção de gramaticalidade neste contexto diz respeito ao reconhecimento de quais sequências são aceitas e quais não são como parte de uma determinada língua. Como exemplo de diferença entre sentenças gramaticais e não gramaticais apresentamos as sentenças abaixo:

(1) *Menina a banco ao tirar dinheiro.
(2) *Banco menina a foi ao dinheiro tirar.
(3) A menina foi ao banco tirar dinheiro.

A sentença (1) e (2) fogem do padrão do Português, enquanto a sentença (3) é reconhecida como pertencente a língua.

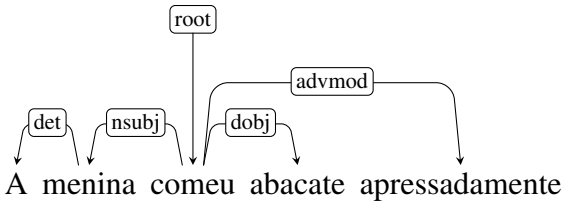
O termo "gramática" se refere ao conjunto de regras e padrões aceitos de forma geral naturalmente internalizadas no aprendizado da língua. Do ponto de vista do estudo de uma língua, não há um certo ou errado *a priori* que deva ser seguido, mas o que é aceito ou não de forma comum pelos falantes. Diferente é a gramática ensinada nas escolas, que tem função de norma, por isso normativa, ou também chamada gramática da norma culta. O uso da língua segundo a norma é conhecido como *norma culta* ou *variante culta* da língua.

A identificação de padrões e regras que descrevem o que é aceito ou não por falantes de uma determinada língua quanto à gramaticalidade é estudada pela Sintaxe. Mais formalmente, "Sintaxe é o estudo dos princípios e dos processos por meio dos quais as sentenças são construídas em línguas particulares" (CHOMSKY, 1957) (p.15).

3.3.3 Gramática de dependência

Para analisar as estrutura das sentenças neste trabalho, escolhemos trabalhar com um modelo de gramática de dependência aplicado à computação. Neste formalismo, constituintes e regras de estruturas frasais não são centrais, como conhecido em alguns outros tipos de análise sintática. No lugar, a estrutura sintática de uma sentença é descrita em termos da associação gramatical entre *tokens* presentes nela. Os argumentos das relações consistem de um *token pai* (*head*) e um *token dependente*, cuja relação entre ambos podem receber marcações de acordo com a relação gramatical do *dependente* em relação ao *pai*, como exemplificado na Figura 2.

Figura 2 – Exemplo de estrutura da gramática de dependência



Jurafsky *et al.* (2014) chama esse tipo de estrutura, nas quais *tags* gramaticais entre *tokens* são provenientes de um inventário fixo e pré-determinado, de relações de *tipos de estrutura de*

dependência [1]. O conjunto de *tags* usadas para marcar estas relações não é necessariamente único, havendo diferentes taxonomias possíveis para determiná-las. Algumas variações destas derivam de diferenças existentes entre as línguas, outras, da história do projeto em que foram desenvolvidas.

A ideia de desenvolver um modelo de dependência que pudesse ser utilizado em diversas línguas, com um conjunto padrão de *tags* e o desenvolvimento de um *treebank* multilínguas anotado deu origem ao projeto *Universal Dependencies* (UD)[2] (NIVRE *et al.*, 2016). O projeto UD tem como proposta prover um inventário de relações de dependência linguisticamente motivadas, úteis computacionalmente e que possa ser utilizado em diversas línguas (NIVRE *et al.*, 2016). Sua origem está na combinação do modelo de dependência sintática de Stanford (MARNEFFE; MACCARTNEY; MANNING, 2006), desenvolvido inicialmente para auxiliar sistemas de reconhecimento de acarretamento textual, e da formulação das *tags* universais do Google (PETROV; DAS; MCDONALD, 2012). O resultado foi um *treebank* para 6 línguas em 2013, 11 em 2014 e que atualmente conta com 90 línguas, incluindo o Português.

3.3.4 Gêneros e tipos textuais

Gêneros textuais são de difícil definição (MARCUSCHI *et al.*, 2002). Quando se diz que um texto tem um gênero, como jornalístico, carta, bula de remédio, receita culinária, entre tantos outros, o que está implícito é que há um conjunto de características que possibilitam o agrupamento de certas formas de comunicação. Estas características são definidas por noções como estilo, propriedades funcionais, conteúdo e composição característica e situam-se funcionalmente nas culturas em que são desenvolvidas (MARCUSCHI *et al.*, 2002), variando também conforme a época e a necessidade de cada população.

A expressão *tipo textual* refere-se à construção teórica definida pela natureza linguística da composição (aspectos lexicais, sintáticos, tempos verbais, relações lógicas), dividida em cinco categorias: narração, argumentação, exposição, descrição e injunção (MARCUSCHI *et al.*, 2002). Neste sentido, associa-se mais a forma em que o texto é escrito do que ao assunto ou objetivo social do texto, podendo um mesmo documento ser misto. Um texto do gênero jornalístico, por exemplo, pode ora fazer uso de narrativa, ora de descrição, entre outras possibilidades.

- Narração: Textos que apresentam sequência de eventos no tempo, geralmente fazendo uso de verbos no passado.
- Argumentação: Textos cuja forma indica argumentação em relação a algum ponto, havendo tentativa de convencimento do leitor.
- Exposição: Textos que descrevem uma situação. Exemplo: ‘Os produtos mais vendidos atualmente são os livros, seguidos dos cadernos.’

[1] Typed dependency structure
[2] https://universaldependencies.org/

- Descrição: Textos que relatam percepções sensoriais.
- Injunção: Textos injuntivos são textos instrutivos, na qual a forma de expressão indica conjunto de passos a serem seguidos. Geralmente fazem uso de verbos no imperativo ou infinitivo. Exemplo: ‘Adicione duas gemas à mistura.’, ‘Pressionar o botão até aparecer uma luz vermelha.’

CAPÍTULO

4

EMBEDDINGS

Neste capítulo tratamos de representações *embeddings* e sua utilização na tarefa de similaridade. Inicialmente apresentamos um contexto histórico, em seguida a descrição de alguns modelos, e por fim um método que usamos para treinar *embeddings* enriquecidas.

4.1 Contexto histórico

Representações vetoriais conhecidas como *embeddings* ficaram populares nos últimos anos. Desde de 2013, impulsionadas com o surgimento de dois métodos hoje bastante conhecidos na área de PLN, Word2Vec (MIKOLOV *et al.*, 2013a) e GloVe (PENNINGTON; SOCHER; MANNING, 2014), as técnicas utilizadas para gerar essas representações e sua relação com tarefas da área sofreram diversas alterações, as quais podem ser divididas em três fases.

4.1.1 Primeira fase: Vamos contar!

A ideia de utilizar vetores para representar textos tem origem nos modelos de espaços vetoriais (do Inglês, VSM) utilizados na área de Recuperação da Informação [1] no início da década de 1970 (TURNEY; PANTEL, 2010).

Inicialmente, vetores para representar documentos eram obtidos através da contagem de termos no texto. Por este método, dado um conjunto de documentos de texto, um documento era representado por um vetor cujos valores eram a quantidade dos termos que constavam nele.

Analogamente, um termo era representado por um vetor cujos valores eram a quantidade de vezes que este aparecia em cada documento, em matrizes conhecidas como termo-documento, ou pela quantidade de vezes que aparecia com outro termo, em matrizes conhecidas como termo-termo.

[1] Área conhecida como *Information Retrieval*

Tabela 6 – Exemplo de matriz documento-termo e termo-termo

	doc 1	doc 2
cão	1	0
gato	0	3
mamífero	1	1
...	...	...
animal	1	1
humano	2	0

	cão	gato	mamífero	...	animal	humano
cão	0	0	0	0	0	0
gato	0	1	0	0	0	0
mamífero	1	1	0	0	0	0
...	...	...	...	...	...	...
animal	0	1	0	0		0
humano	2	0	0	0	0	0

Fonte: Autoria própria

A partir destas matrizes eram calculados, por exemplo, semelhança entre documentos e termos. Visando melhorar os resultados obtidos, algumas modificações foram propostas. Considerações como a de que um termo muito frequente nos diversos documentos, como um artigo, caracteriza menos o conteúdo de um documento do que outros mais raros utilizados especificamente nele, como um nome próprio ou um substantivo raro, motivaram o uso de uma proporção do termo em relação a sua frequência nos diversos documentos, conhecida como tf–idf, do Inglês *term frequency–inverse document frequency*. Outra modificação foi a incorporação da contagem de outros atributos, como classes lexicais (substantivos, adjetivos, etc.) ou a contagem de *n* termos que ocorrem sequencialmente, os *n*-gramas.

O sucesso de modelos de VSM na área de recuperação de informação inspirou pesquisadores a ampliarem o uso dos métodos para outras tarefas em PLN (TURNEY; PANTEL, 2010). Contudo, o uso destes métodos baseados em contagem apresentava duas desvantagens. A primeira era a esparsidade das representações geradas. Uma vez que cada dimensão no vetor representava um atributo, como um termo do vocabulário ou um *n*-grama do conjunto de documentos, e grande parte desses geralmente não aparecem em todos os documentos, as representações geradas eram repletas de valores nulos. A segunda era a grande dimensão destes vetores, que crescia de acordo com o tamanho do vocabulário e atributos adicionados, gerando representações computacionalmente pesadas de serem processadas. Para estas questões, eram aplicados métodos de redução de dimensionalidade e seleção de atributos independentes, como *Singular Value Decomposition* (SVD) e *Latent Semantic Analysis* (LSA), na tentativa de reduzir o tamanho das representações e torná-las mais adequadas ao processamento, sem que com isso se perdessem atributos relevantes.

4.1.2 Segunda fase: Vamos prever!

Em 2003, Bengio *et al.* propõem um modelo em que representações vetoriais de palavras são diretamente aprendidas pela atualização de pesos decorrente do treino de uma rede neural. Sua proposta contrapõe os outros modelos de representações vetoriais difundidos, nos quais matrizes de co-ocorrência de termos em documentos eram usadas para gerar vetores representativos de palavras.

A ideia que apresenta é que a distribuição de probabilidade da sequência de palavras no texto pudesse ser representadas de modo compacto, em vetores gerados de forma automatizada, aprendidos diretamente pelo modelo neural. Os autores destacam particularmente o ganho em relação à dimensionalidade dos vetores resultantes, propondo que, ao invés de caracterizar similaridade com variáveis randômicas ou determinísticas, fossem usados vetores com valores contínuos e reais para representar cada palavra.

Embora não utilizem o termo *embedding* especificamente, a ideia que expõem, a de fazer uso do mecanismo de aprendizado de pesos proveniente de redes neurais para incorporar informações de distribuição de palavras no texto como forma de gerar vetores densos, seria largamente associada ao termo anos depois. Seu trabalho também antecipa algumas das questões que viriam a ser bastante exploradas com a popularização do uso de *embeddings* em PLN. Entre elas, a limitação de utilizar uma única representação para cada palavra, podendo cada uma estar associada a múltiplos sentidos, e o uso de redes neurais recorrentes para capturar informações de contextos mais distantes.

Dez anos mais tarde, em 2013, Mikolov *et al.* propõe o Word2Vec, um método de gerar representações vetoriais de palavras utilizando princípios análogos e que ganharia notoriedade na área de PLN. No Word2Vec, a partir de uma janela de palavras de tamanho pré-fixado e de uma palavra *target* central nesta janela, vetores são automaticamente gerados através da atualização de pesos decorrente da fase de treinamento do algoritmo. O método possui duas variações, uma na qual o objetivo é predizer a palavra *target* a partir das palavras na janela (CBOW), outra, oposta, na qual o objetivo é predizer as palavras da janela dada a palavra *target* (Skip-Gram). Ambas utilizam tanto como *input*, como como alvo, o próprio texto.

A previsão, portanto, não é feita em uma tarefa típica, como *parsing*, *part of speech tagging* ou análise de sentimento, por exemplo, mas do texto em si mesmo.

Outro método conhecido surgiu na mesma época, o GloVe (PENNINGTON; SOCHER; MANNING, 2014). Este método combina características de técnicas que tinham como base matrizes de contagem com o treinamento de vetores, treinando os elementos não nulos de matrizes de coocorrência termo-termo. Embora considere que exista pontos em comuns entre modelos baseados em matrizes de contagem e modelos do tipo do Word2Vec, Pennington, Socher e Manning (2014) aponta que os primeiros carregam a vantagem de utilizarem a estatística presente no *corpus* como um todo, uma vez que utilizam matrizes com informações de todos os documentos simultaneamente, conceito que incorpora em seu modelo e que lhe confere o nome GloVe, proveniente de "global vector". Além desta consideração, para Pennington, Socher e Manning (2014) Word2Vec é um método mais opaco em relação às estatísticas de ocorrências de *corpus* que carregam.

A principal diferença que distingue estes métodos dos anteriores utilizados na primeira fase está no uso de algoritmos que realizam treinamento para se obter os vetores densos. Baroni, Dinu e Kruszewski (2014) resumem essa distinção separando os métodos em duas classes,

modelos baseados em contagem (*count-based*) e modelos preditivos baseados em treinamento (*training-based*). Embora *embeddings* não seja um termo utilizado para denominar somente representações *training-based*, estes modelos ficaram conhecidos como modelos de *embedding*.

Modelos de *embedding* ganharam fama de capturarem relações semânticas do texto em virtude de trabalhos como o do próprio Mikolov *et al.* (2013a), que cita a possibilidade de se realizar operações algébricas com os vetores mantendo-se um espelhamento da relação de significado das palavras. Como exemplo cita que em seu trabalho o vetor(*king*) - vetor(*man*) + vetor(*woman*) teve como resultado um vetor próximo ao vetor(*queen*). A avaliação deste tipo de relação com vetores representativos de palavras ficou conhecida como tarefa de analogia. Outro fator relevante para a associação entre *embeddings* e relações semânticas veio da percepção de que havia certa associação de significado de pares de palavras com o cálculo do cosseno entre os pares de *embeddings*,conhecido como similaridade de cossenos.

A ideia de comparar relações vetoriais com relações de significado já estava presente nos modelos de VSM – Turney e Pantel (2010) apontam que a inovação destes modelos estava em utilizar a frequência do *corpus* de texto como uma pista para descobrir informações semânticas – e ganha força com as *embeddings*. Consequentemente questões como *se* e *quais* relações semânticas estariam de fato refletidas nos vetores gerados por modelos de *embeddings* passam a motivar uma série de testes comparativos entre suas características como vetores e a percepção de similaridade que humanos têm sobre as palavras (ANTONIAK; MIMNO, 2018; SCHNABEL *et al.*, 2015). Paralelamente, o uso de *embeddings* como entrada de algoritmos de aprendizado de máquina na realização de tarefas variadas, motivaram o teste destes modelos não por suas características vetoriais, mas como parte de tarefas de classificação, avaliando seu desempenho sob a luz dos resultados obtidos nestas tarefas.

Em 2015, Schnabel *et al.* categorizaram as avaliações em duas classes distintas, avaliações intrínsecas e avaliações extrínsecas. Avaliações intrínsecas avaliam *embeddings* de palavras observando diretamente a correlação entre relações geométricas entre os vetores e relações semânticas, geralmente obtidas com o uso de um inventário de termos anotados. Avaliações extrínsecas medem o desempenho de *embeddings* utilizadas como vetores de atributos em algoritmos supervisionados de aprendizado de máquina utilizados em tarefas alvo de PLN, frequentemente referidas como *dowstream tasks* (BAKAROV, 2018).

Entre as limitações dos modelos iniciais de *embeddings* de palavras, duas ganharam destaque (QIU *et al.*, 2020). A primeira é que as representações geradas eram estáticas, ou seja, uma vez treinado, o modelo associa cada palavra a um vetor numérico único, o que limita distinções decorrentes de contexto. Uma palavra como *manga*, por exemplo, teria uma única representação, independente se esta aparecesse na sentença “Comi uma *manga* hoje.” ou “A *manga* da minha camisa rasgou.”. A segunda é quanto à existência de representação para elementos nunca vistos no treinamento. Dentro do paradigma de *embeddings* estáticas, uma palavra nova não possui representação no modelo, ficando a pergunta de como lidar com este

caso. Ignorar? Assumir uma *embedding* padrão? Para mitigar estas limitações foram propostos modelos de representações não restritos ao nível da palavra, tal como representações associadas a sentenças e textos, como Doc2Vec (LE; MIKOLOV, 2014) e representações de segmentos ou sub-palavras, como FastText (BOJANOWSKI *et al.*, 2017).

4.1.3 Terceira fase: Vamos transferir!

O avanço na capacidade de processamento e armazenamento computacional abriu caminho para o treinamento de modelos de representações em quantidades cada vez maiores de dados e o emprego de redes neurais com arquiteturas cada vez mais profundas, favorecendo o aprendizado de propriedades distribucionais da língua em larga escala. Adicionalmente, a percepção de que há ganho na reutilização e no retreino de modelos vem como um incentivo ao pré-treinamento de modelos em *corpus* extensos como etapa preliminar para algoritmos antes da especialização em tarefas específicas (QIU *et al.*, 2020). O uso de modelos pré-treinados como ponto de partida permite que se economize o tempo de treinamento de *embeddings*. Além disso, uma nova geração de modelos incorpora a ideia de aprender *embeddings* contextualizadas, como ELMo (PETERS *et al.*, 2018), OpenAI GPT (RADFORD *et al.*, 2018) e BERT (DEVLIN *et al.*, 2019).

A proposta de diversos modelos atuais é que treinamento de *embeddings* e tarefa alvo não sejam necessariamente etapas isoladas. No caso do BERT, por exemplo, sistema que quando publicado alcançou o estado da arte em diversos *datasets*, como de identificação de similaridade semântica textual e identificação de paráfrase, o treino de representações inicialmente ocorre diretamente no texto em tarefas que não requerem rotulação de dados. A partir dela, o modelo prevê dois usos possíveis, utilização das representações geradas ou posterior ajuste do sistema para uma tarefa alvo.

4.2 Modelos de *embeddings*

Embedding de palavras são vetores obtidos por um mapeamento:

$$V \rightarrow \mathbb{R}^D : w \rightarrow \vec{w}$$

no qual uma palavra w de um vocabulário V é mapeada em um vetor de valores reais $\vec{w}$ em um espaço de dimensão D (SCHNABEL *et al.*, 2015).

Na prática, a dimensão D é dada como um parâmetro do modelo utilizado para realizar o mapeamento. Embora este mapeamento seja bastante difundido para o nível da palavra, não se restringe a ele. Também podem ser geradas representações *embeddings* de segmentos de tamanho arbitrários, sejam maiores, como sentenças, ou menores, como caracteres. Nestes casos

o vocabulário V é generalizado para o conjunto destes segmentos e w representa cada elemento deste conjunto para o qual se obtém um vetor $\vec{w}$.

A forma como este mapeamento é realizado depende do algoritmo utilizado. Alguns algoritmos tem como *output* diretamente representações *embeddings*, que podem ser utilizadas como entrada em outro algoritmo na realização de tarefas de PLN. Outros tratam a geração de *embeddings* como uma etapa preliminar, um pré-treinamento, que é depois ajustado para uma tarefa específica no mesmo algoritmo. No primeiro caso os vetores são fixos, ou seja, uma vez gerados, cada dimensão do vetor, que representa um atributo, não é alterada. A seguir apresentamos alguns dos principais métodos de geração de *embeddings* fixas, que podem ser baseados apenas no texto ou serem enriquecidas com informações adicionais.

4.2.1 *Modelos de* embeddings *baseadas no texto*

Como modelos de *embeddings* baseadas no texto consideramos modelos que não utilizam informações extras anotadas, como por exemplo anotação de *part of speech* ou anotações sintáticas, na geração dos vetores. Modelos que incorporam estas informações, entram em modelos de *embeddings* enriquecidas. A seguir, apresentamos resumidamente alguns dos principais métodos de *embeddings* baseadas no texto.

Word2Vec (MIKOLOV *et al.*, 2013a) inclui duas técnicas de geração de vetores, Skip-gram e CBOW. Ambos utilizam da predição supervisionada de palavras em palavras, através de uma arquitetura inspirada em redes neurais com uma única camada. No método de geração Skip-gram o objetivo é predizer o contexto dada uma palavra. Contexto, neste cenário, é o conjunto de palavras que aparecem em uma janela de texto de tamanho pré-determinado adjacente (antes e depois) a palavra de entrada. No CBOW o objetivo é prever a palavra, dado o contexto. Word2Vec serviu de inspiração para outros modelos que a partir de adaptações desenvolveram outras formas de gerar *embeddings*.

GloVe (PENNINGTON; SOCHER; MANNING, 2014), gera vetores de palavras a partir da fatorização explícita da matriz dada pelo logaritmo de contagens de palavras no *corpus* com relação aos pares de contexto de palavras.

FastText (BOJANOWSKI *et al.*, 2017) utiliza cadeias de caracteres no treinamento de *embeddings* de palavras, considerando n-gramas de caracteres que a compõem. O treino segue os métodos Word2Vec (CBOW e Skip-gram). A representação de palavras utilizando caracteres é proposta para capturar aspectos de morfologia e contribuir com a representação de palavras que estejam fora do vocabulário de treinamento.

Wang2Vec (LING *et al.*, 2015) é baseado em uma modificação do método Skip-gram do Word2Vec para que seja considerada a ordem das palavras na janela de contexto utilizada no treinamento dos vetores. A motivação deste método é que a incorporação da sequência em que as palavras aparecem seja relevante para capturar aspectos sintáticos da língua.

Doc2vec (LE; MIKOLOV, 2014) gera representações densas que podem ser aprendidas para um tamanho arbitrário de texto, variando de frases a múltiplos parágrafos. O treino é realizado como nos modelos CBOW e Skip-gram, porém considerando para cada texto um id, que entra como *input* no treino. Desta forma, a predição de palavra-contexto e contexto-palavra é feita com a indicação extra de diferenciação de texto dado pelo id.

4.2.2 Modelos de embeddings enriquecidas

Consideramos *embeddings* enriquecidas representações geradas a partir de modelos que utilizam informações anotadas para serem geradas.

Sence2Vec (TRASK; MICHALAK; LIU, 2015) utiliza o texto anotado com *part-of-speach* para desambiguar representações de palavras. A partir do rótulo lexical anotado de cada palavra, inicialmente o modelo contabiliza os diversos usos atribuídos a elas. Assim, se uma ocorrência da palavra *fala* estiver anotada como substantivo e outra como verbo por exemplo, o modelo distingue dois usos e gera um "senso" (*sense*) para cada um. As representações são obtidas pelo treino utilizando o Word2Vec (CBOW, Skip-gram ou Structured Skip-gram) na qual se aprende representações para cada senso. Ao invés do modelo tentar prever cada *token* dados os *tokens* no redor, são preditos os sensos dados os sensos no contexto.

Levy e Golgberg (LEVY; GOLDBERG, 2014) Os autores utilizam anotações sintáticas no seu modelo de representação para alterar o contexto de predição. Diferentemente de modelos como Word2Vec, no qual o contexto de predição é dado por *tokens* adjacentes à palavra *target*, o contexto utilizado por Levy e Goldberg (2014) são *tokens* relacionadas por relações sintáticas. Os autores generalizaram o modelo de Skip-gram para aceitar outros contextos, substituindo a *bag of words* do contexto linear de *tokens* adjacentes pelo contexto dado pela anotação de um parser.

4.2.3 BERT

BERT (DEVLIN *et al.*, 2019), acrônimo de *Bidirectional Encoder Representations from Transformers*, é um modelo de representação de língua que utiliza uma arquitetura com camadas de Encoders Transformers bidirecionais, baseada na implementação descrita em Vaswani *et al.* (2017), com mecanismo de atenção para pré-treinar representações.

Uma vez pré-treinado, o modelo pode ser utilizado como fonte representações fixas para outro sistema, em uma abordagem conhecida como *feature based*, ou retreinado especificamente para uma tarefa alvo desejada, técnica conhecida como *fine-tuning*.

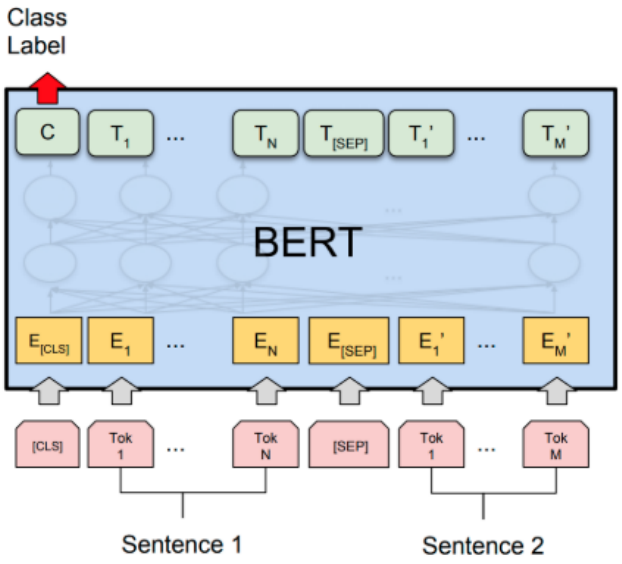
O pré-treinamento não requer corpus anotado e é feito para duas tarefas, predizer *tokens* escondidos do texto de entrada, do Inglês *Masked Language Model* (MLM), e predizer a próxima sentença do texto, do Inglês *Next Sentence Prediction* (NSP):

- *Masked Language Model* (MLM): O modelo randomicamente "mascara" alguns dos *tokens* do texto de entrada que passam a ser o alvo da tarefa de predição. O objetivo é acertar o vocabulário original mascarado.
- *Next Sentence Prediction* (NSP): Dado um par de sentenças, a tarefa é predizer se são consecutivas. O modelo utiliza 50% das vezes pares de sentenças que aparecem em seguida nos documentos do *corpus* de treinamento, e as 50% restantes seleciona a segunda sentença randomicamente no *corpus*.

As camadas do modelos são completamente conectadas e para cada *token* de entrada são computados três vetores, denominados *key*, *value* e *query* (VASWANI *et al.*, 2017), utilizados para criar representações com pesos distintos para cada um dos *tokens*. A representação *embedding* do BERT é contextualizada, o vetor é gerado a partir do contexto de ocorrência.

No *fine-tuning* o modelo é inicializado com os parâmetros pré-treinados e uma última camada é adicionada para a classificação final. A partir desta configuração o treino é feito com os dados da tarefa alvo, *input* e rótulos, e os parâmetros reajustados. Na Figura 3 é representado o *fine-tuning* do BERT para tarefas com pares de sentenças como *input*. Na primeira etapa o modelo "tokeniza" as sentenças (primeira fileira da figura). Os *tokens* são próprios do modelo, não correspondendo necessariamente a divisão de palavras usuais. Dois *tokens* especiais, [CLS] e [SEP], são utilizados para marcar o *input* e separar sentenças respectivamente.

Figura 3 – Representação do *fine-tuning* do BERT



Fonte: (DEVLIN *et al.*, 2019)

Na Figura 4 são apresentados os resultados do BERT nos *datasets* do GLUE Benchmark (WANG *et al.*, 2018) em relação ao resultado de outros sistemas neste mesmo *benchmark*. Os dados mostram que o BERT alcançou 86,5 na correlação de Spearman no *Semantic Textual Similarity Benchmark* (STS) e 72,1 F1-score no *Quora Question Pairs* QQP, dois *datasets* anotados para tarefa de similaridade, além de alcançar o estado da arte em 2018 para outros *datasets* focados em outras tarefas.

Figura 4 – Resultados do BERT no GLUE Benchmark

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

O número em cada tarefa denota o número de exemplos no treino (k = mil). Para o QQP e MRPC são reportados o F1-scores, para o STS-B a correlação de Spearman, e para outras tarefas é reportada a acurácia. Fonte: (DEVLIN *et al.*, 2019)

Outros modelos baseados na mesma arquitetura foram propostos. A Tabela 7 ilustra as diferenças entre o BERT e alguns destes quanto ao *corpus* utilizado e a tarefa escolhidos para o pré-treino.

Tabela 7 – Comparação entre modelos: características de pré-treino

modelo	tarefa	arquitetura	*corpora*
BERT	MLM, NSP	Transformer based	BooksCorpus (800M words) English Wikipedia (2,500M words).
DistilBERT	MLM	Transformer based	BooksCorpus (800M words) English Wikipedia (2,500M words)
RoBERTa	MLM	Transformer based	BooksCorpus (800M words) English Wikipedia (2,500M words) CC-News (63M news articles) OpenWebText + Stories
MPNet	MLM+PLM	Transformer based	BooksCorpus (800M words) English Wikipedia (2,500M words) CC-News (63M news articles) OpenWebText + Stories

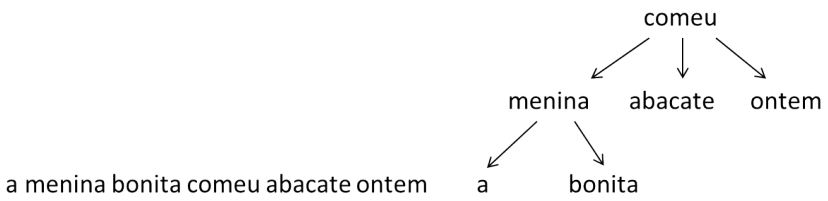
MLM: masked language modeling, NSP: Next sentence prediction, PLM: permuted language modeling

4.3 Grafos e *embeddings*

Diversas relações linguísticas podem ser representadas em forma de grafos. Um exemplo no escopo deste trabalho são as árvores de dependência sintática na qual palavras são conectadas de acordo com a relação gramatical que possuem. Tais relações podem não ser captadas por métodos de pré-treinamento de *embeddings* como Word2Vec, que atuam sobre o texto considerando as palavras linearmente, na ordem em que aparecem.

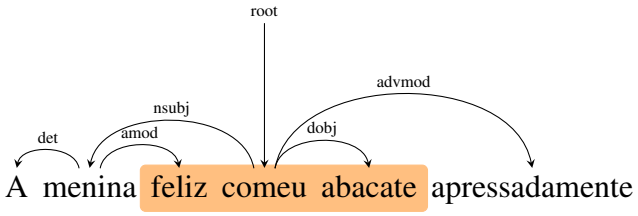
Esta diferença é ilustrada nos dois exemplos a seguir. No primeiro, na Figura 5, destacamos a distinção entre ordem linear e a relação dada pela gramática de dependência na árvore sintática.

Figura 5 – Ordem linear *versus* grafo de dependência



Fonte: Autoria própria.

No segundo exemplo abaixo, contrastamos uma janela na ordem linear do texto (em laranja) e a árvore representada pelas setas. Observamos que “apressadamente” é uma forma de “comer” e “menina” quem faz o ato de comer, embora ambas as palavras não estejam juntas ao verbo.



Alguns algoritmos de predição atuam diretamente em grafos, utilizando por exemplo pesos de arestas no cálculo de probabilidades para classificar nós cuja classe é desconhecida no grafo. Outros, como redes neurais, recebem como entrada vetores numéricos em que cada dimensão representa um atributo da instância a ser classificada, os quais utilizam para a predição.

O encontro destas duas abordagens requer uma forma de mapear nós de uma representação em grafo em vetores densos que possam ser processados por estes algoritmos. Matematicamente, seja um grafo $G = (V, E)$, em que V são os nós (vértices) e E as arestas, o que se deseja é uma função que mapeie $f : V \rightarrow \mathbb{R}^d$, na qual d é o número de dimensões (atributos) que o vetor de cada vértice terá. O método escolhido para realizar este mapeamento é variável. A seguir apresentamos aquele que utilizamos neste trabalho.

4.3.1 Node2Vec

Node2Vec (GROVER; LESKOVEC, 2016) é um algoritmo semi-supervisionado para o aprendizado de atributos em grafos no qual os nós são mapeados em um espaço dimensional denso de atributos. Para cada nó $u \in V$ é definida uma noção de vizinhança $N_S(u) \subset V$ do nó u gerada por uma estratégia de amostragem S. A função objetivo do algoritmo é maximizar a o logaritmo da probabilidade de observar a vizinhança $N_S(u) \subset V$ para um nó u dado a representação vetorial $f(u)$ de u:

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u))$$

O cálculo da probabilidade condicional é feito assumindo-se que as probabilidades são independentes, isto é, que a probabilidade de se observar um determinado nó na vizinhança é independente da de se observar qualquer outro nó na vizinhança. Também é assumido pelo modelo que há simetria entre os nós no espaço de atributos, um nó fonte e um nó de sua vizinhança possuem o mesmo efeito um no outro. Pelo processo de otimização ser computacional, assim como no caso de Skip-gram, é feita uma aproximação usando *negative sampling* (MIKOLOV *et al.*, 2013b).

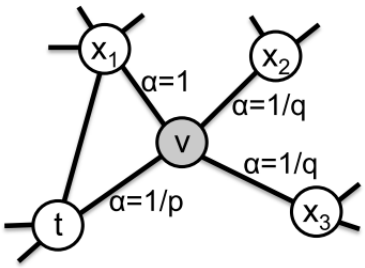
A estratégia de amostragem da vizinhança S é feita considerando-se um tamanho definido k máximo de nós. Para um dado nó u, é realizada a amostragem de múltiplos conjuntos de vizinhança obtidos através de um procedimento de passeios aleatórios (*random walks*) enviesados

que exploram os nós vizinhos tanto em profundidade quanto em largura. O passeio é feito considerando-se um viés de busca α calculado a partir de dois parâmetros: p e q.

O parâmetro de retorno p controla a probabilidade de imediatamente se revisitar um nó no passeio. Quanto mais alto, menor a tendência de revisitar um nó. O parâmetro in-out q permite que a busca diferencie entre *inward* e *outward* nós. A Figura 6 ilustra o procedimento de passeio em que a transição do nó t para o nó v acabou de ocorrer. Neste ponto o algoritmo deve avaliar a próxima transição com base nos valores de viés de busca. Os parâmetros p e q são indicados de acordo com essa transição.

A partir das caminhas são obtidas as vizinhanças para maximização da função objetivo.

Figura 6 – Ilustração dos parâmetros na caminhada do Node2Vec



Fonte: (GROVER; LESKOVEC, 2016)

CAPÍTULO

5

AVALIAÇÃO DE MODELOS DE REPRESENTAÇÃO

Neste capítulo avaliamos modelos de representação através dois experimentos.

No **Experimento 1**, examinamos o impacto de alterar o modelo de *embedding* nos resultados finais da classificação de similaridade sentencial em relação a variação de outros parâmetros. Também avaliamos o resultado obtido com o BERT comparando o uso de suas *embeddings* como *feature based* em relação à realização de *fine-tuning*. Nossos resultados mostram que a escolha do modelo de *embedding* foi o fator mais impactante nos valores de similaridade preditos em relação a outros parâmetros, como número de épocas e tamanho do *batch* usados no algoritmo, evidenciando a importância que modelos de representação tem nos resultados finais.

No **Experimento 2**, realizamos uma avaliação intrínseca comparando valores de similaridade para pares de *embeddings* de sinônimos e antônimos no Português. O resultado indica que, embora os pares de sinônimos tenham valores de similaridade mais altos do que pares de palavras aleatórias, o mesmo ocorre para pares de antônimos, não sendo possível distingui-los através da observação apenas da similaridade de cossenos entre eles.

5.1 Experimento 1

Neste experimento, nossa motivação foi responder as seguintes questões:

- O que impacta mais o resultado de classificação: o modelo de representação *embedding* escolhido ou os hiperparâmetros do classificador MLP utilizado para estimar a similaridade entre sentenças?
- Qual a diferença de desempenho obtido entre (i) o uso de *embeddings* pré-treinadas como atributos de um classificador STS e (ii) o ajuste de *embeddings* de forma integrada na

tarefa de STS (e.g. *fine-tuning*)?

Investigamos quatro modelos de representações *embeddings*. Dois modelos considerados de uma primeira geração de representações, Word2Vec (MIKOLOV *et al.*, 2013a) e Doc2Vec (LE; MIKOLOV, 2014), um modelo de representação contextual, *embeddings* extraídas do BERT (DEVLIN *et al.*, 2019), e um modelo que desenvolvemos a partir do treinamento no grafo de dependência sintática das sentenças. Treinamos dois classificadores, um algoritmo Multi Layer Perceptron (MLP) para identificação de similaridade textual aplicada especificamente ao *Quora Question BenchMark* (QQB) [1] e o próprio BERT com uma camada de classificação adicional, no modo *fine-tuning*, para efeito de comparação.

Para cada modelo e suas variações testamos múltiplas arranjos de MLP, alterando a dimensão da camada escondida, tamanho de *batch* e número de épocas, totalizando 273 configurações. A escolha do algoritmo não foi feita orientada para superar os melhores resultados conhecidos neste *dataset*, mas visando a comparação entre diferentes modelos de *embeddings* de forma controlada em relação aos resultados e em relação a alteração dos hiperparâmetros utilizados na classificação. Os resultados indicam que o impacto do modelo de *embedding* utilizado é maior do que aquele causado pela variação de qualquer outro parâmetro, como a dimensão dos vetores e hiperparâmetros do MLP na classificação final. Observamos também que as *embeddings* extraídas a partir do BERT não obtiveram os melhores resultados entre todos os modelos quando usadas no MLP, embora o resultado do *fine-tuning* do BERT seja significativamente melhor.

5.1.1 Material e Métodos

5.1.1.1 Dataset

Para este trabalho utilizamos o *Quora Question BenchMark* (QQB), um *dataset* composto por 403.564 pares de perguntas em Inglês anotadas para equivalência semântica, sendo 363.177 treino e 40.371 teste. A anotação é binária, tendo 1 como indicativo de que o par é equivalente e 0 para não equivalente do ponto de vista da resposta que podem receber. Exemplos de pares anotados podem ser vistos na Tabela 8.

Os dados foram disponibilizado pelo próprio site *Quora*, que consiste em uma plataforma para viabilizar troca de conhecimento através de perguntas e respostas. O usuário pode publicar perguntas que ficam abertas para que outros usuários respondam. Uma das dificuldades enfrentadas pelo site é a identificação de perguntas equivalentes para mitigar a ineficiência de se ter questões duplicadas, o que motiva a busca por melhores resultados na tarefa de similaridade.

[1] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Tabela 8 – Exemplos de pares de sentenças do QQP

questão 1	questão 2	classe
How do I lose weight fast?	*What is the best way to reduce weight fast?*	1
How do you become an air traffic controller?	*How is air traffic controlled?*	0
What is the average IQ of a human?	*What is the iq of the average person?*	1
Why would anyone want a Red Panda as a pet?	*Can you keep a Red Panda as a pet?*	0

Fonte: Tabela elaborada pelo autor com dados do QQP.

5.1.1.2 Pré-processamento

Os dados do *dataset* original são desbalanceados, tendo mais amostras de treino para a classe 0. Em vista disso, optamos por balancear os dados realizando uma subamostragem dos dados de treino, retirando a diferença da classe excedente de forma aleatória. O resultado foi um *dataset* com 268.282 pares de sentenças de treino, sendo 134.141 da classe 1 (equivalentes) e 134.141 da classe 0 (diferentes). Mantivemos os dados de teste com os 40.371 pares originais.

Padronizamos o texto, deixando todas as sentenças em caixa baixa e o ponto de interrogação no final de cada pergunta separado da última palavra. As sentenças pré-processadas foram utilizadas como *input* nos modelos de geração de vetores descritos com mais detalhes a seguir.

5.1.1.3 Modelos de representações embeddings

Para verificar o impacto que a mudança na representação de texto tem no resultado final de tarefas de similaridade semântica textual, escolhemos quatro modelos de representação *embeddings*, Skip-gram do Word2Vec, Doc2Vec, *embeddings* extraídas do BERT especificamente para o *dataset* e um modelo enriquecido com informações sintáticas que desenvolvemos para este trabalho.

Para as *embeddings* do BERT utilizamos o modelo pré-treinado *base-uncased*, somando os valores do vetor de cada *token* da última camada do pré-treinamento para gerar uma representação sentencial.

Para as representações enriquecidas, geramos as *embeddings* sintáticas a partir do grafo de dependência sintática de cada sentença. O trabalho foi feito em quatro etapas: na primeira fizemos uso do parser de dependência do spaCy [2] para anotar as relações sintáticas. Na segunda, geramos um grafo considerando as palavras como nós e as relações sintáticas como arestas. Na terceira, para preparar as representações vetoriais de cada nó, fizemos uso da implementação do método Node2Vec[3] para realizar *random walks* (passeios aleatórios) de tamanho pré-fixado no grafo a partir de cada nó. Desta forma, obtivemos um conjunto de dados com as palavras sintaticamente adjacentes, ao invés de linearmente adjacentes. Por fim, utilizando estes dados,

[2] https://spacy.io/
[3] https://pypi.org/project/node2vec/

treinamos as *embeddings* com o modelo Skip-gram do Word2Vec[4] e obtivemos as representações sentenciais a partir da soma dos vetores das palavras.

5.1.1.4 Modelo classificação

As representações de sentenças serviram como entrada para dois modelos de classificador: Multi Layer Perceptron (MLP), implementado através da biblioteca do TensorFlow [5], e o modelo BERT, pela implementação do Hugging Face [6]. Os testes foram realizados para diferentes arquiteturas de MLP, permitindo a comparação da influência dos quatro modelos de representações em conjunto com os parâmetros do classificador. A classificação de similaridade realizada utilizando o BERT foi feita para o pré-treinamento *base-uncased* fazendo uso do *fine-tuning* do modelo.

5.1.1.5 Configuração dos experimentos

Selecionamos para o experimento quatro tipos de *embeddings*, duas fixas tradicionalmente utilizadas na área de PLN, uma contextual proveniente do pré-treinamento do BERT e uma gerada a partir de informações sintáticas de nossa autoria.

Para as *embeddings* fixas, usamos os modelos Skip-gram e Doc2Vec para três tamanhos diferentes de vetores (300, 500 e 768) e dois tamanhos de janelas de contexto (2 e 3) para cada modelo, obtendo-se assim 12 representações *embeddings* dos dados. Para as contextuais, as *embeddings* do BERT foram extraídas da última camada modelo pré-treinado, que tem dimensão padrão *base* 768. As representações enriquecidas foram geradas em três dimensões (300, 500, 768). Para as duas primeiras com janela de contexto de 3 e para a última janela de contexto de 2 e 3, totalizando 4 representações *embeddings* dos dados.

Para permitir a comparação dos resultados entre diferentes experimentos, o *dataset* apresenta uma segmentação padrão dos dados entre treinamento e teste previamente fornecida. Nos experimentos realizados utilizamos esta mesma divisão, fazendo uso de 5% do treino como validação.

Os dados foram treinados no classificador MLP com um única camada escondida. Testamos 16 configurações, variando a dimensão da camada escondida (50 e 80), o número de épocas (20, 40, 80 e 100) e o tamanho do *batch* (200 e 400) e no BERT, totalizando no final 273 experimentos. Para cada experimento calculamos a acurácia, precisão, revocação e F1-score nos dados de teste.

[4] https://radimrehurek.com/gensim/models/word2vec.html
[5] https://www.tensorflow.org/
[6] https://huggingface.co/

5.1.2 Resultados

A seguir apresentamos os resultado do experimento realizados para os dois classificadores, MLP e BERT no modo *fine-tuning*. A Tabela 9 mostra os melhores resultados para cada modelo de *embedding* e classificadores considerando os valores de F1-score. Os modelos de *embedding* estão nomeados pelas duas primeiras letras do modelo, com DV para Doc2Vec, SG para Skip-gram, BE para BERT *embedding* e NV para *embeddings* sintáticas, seguidas pelo tamanho do vetor e da janela de contexto utilizada.

Tabela 9 – Resultados por modelo de *embedding* no MLP e BERT

configuração	épocas	batch	acurácia	precisão	revocação	F1-score	tempo (horas)
SG_300_3_MLP(80)	100	400	0.784	0.695	0.734	**0.714**	< 0.15
BE_768_MLP(80)	100	400	0.749	0.683	0.592	0.634	< 0.15
DV_768_2_MLP(80)	40	200	0.700	0.630	0.449	0.524	< 0.15
NV_300_3_MLP(80)	80	200	0.679	0.566	0.549	0.557	< 0.15
BERT_fine-tuning	2	50	0.883	0.874	0.877	**0.875**	8.46

São apresentados os melhores resultados para F1-score de cada modelo de *embedding* e o resultado do BERT-*fine-tuning*. O nome de cada experimento é dado pelo modelo de *embedding*, tamanho do vetor, tamanho da janela utilizada e dimensão da camada escondida do algoritmo MLP. Última coluna é o tempo de treinamento sob as mesmas condições.

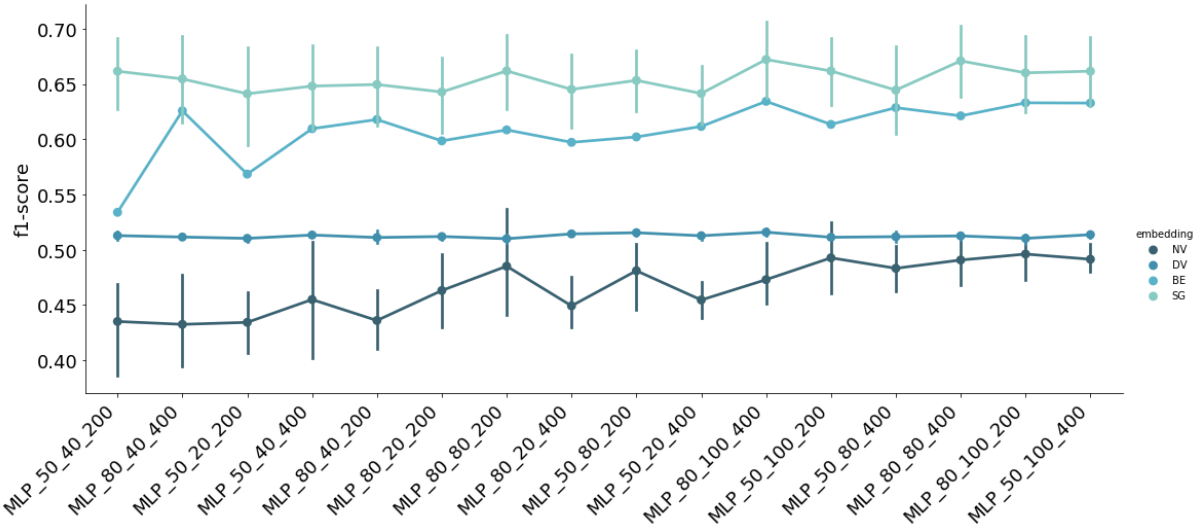
5.1.2.1 Dimensão vetorial

A análise do tamanho dos vetores gerados e a janela de contexto utilizada nos experimentos mostra que os melhores resultados para o modelos DV foram obtidos com os vetores de maior dimensão (768). Já no modelo SG e NV o melhor desempenho foi obtido para vetores de dimensão 300. No entanto, no geral, a influência da alteração de tamanho e janela de contexto nos resultados de cada modelo foi pequena, havendo pouca diferença nos valores finais. O impacto maior ocorreu na comparação entre modelos: observamos um claro desempenho melhor do SG, que alcançou 0.78 de acurácia e 0.71 de F1-score, seguido das *embeddings* do BERT.

5.1.2.2 Hiperparâmetros do classificador MLP

Uma segunda análise foi realizada quanto a relação dos modelos e os hiperparâmetros do classificador MLP. A Figura 7 mostra os resultados de todas as configurações do classificador para cada modelo de *embedding* testado. Embora alterações no número de neurônios da camada escondida, tamanho de *batch* e número de épocas do classificador tenha impactado os resultados, com destaque para um aumento de acurácia conforme aumentou-se o número de épocas, a variabilidade dos parâmetros não teve tanta influência quanto a mudança do modelo de *embedding*. Cada modelo ficou em uma faixa de valores para F1-score, como mostrado pela separação entre as linhas de cada modelo do gráfico da figura.

Figura 7 – Valores de F1-score por modelo de *embedding* para cada configuração de MLP

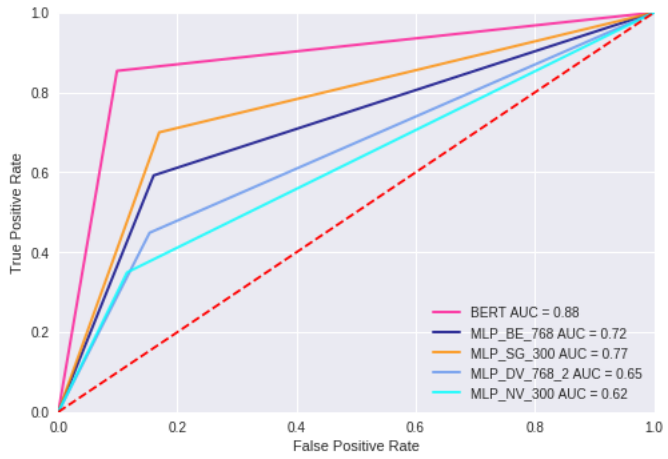


No eixo horizontal cada configuração é indicada pelos parâmetros dimensão da camada escondida, número de épocas e tamanho do *batch*, respectivamente. Fonte: Autoria própria.

5.1.2.3 *Predição e comparação entre classificadores*

Observamos, como esperado de acordo com a literatura, um desempenho superior na utilização do BERT no modo *fine-tuning* no qual uma camada extra foi adicionada e o modelos inteiro atualizado. O uso do BERT com duas épocas e *batch* tamanho 50 resultou em 0.875 de F1-score, enquanto com o MLP para uma camada o melhor resultado foi de 0.714 para o modelo SG_300_3. Quanto ao tempo, o treinamento do BERT consumiu mais de 8 horas, fazendo uso de GPU, enquanto o treinamento do MLP durou menos de 8 minutos sob as mesmas condições.

Figura 8 – Curva ROC



Receiver Operating Characteristic Curve (curva ROC) do classificador BERT e dos quatro melhores resultados de cada modelo de *embedding* para o classificador MLP. Fonte: Autoria própria.

A comparação entre o resultado dos melhores classificadores observando-se a curva ROC da Figura 8, indica que o classificador BERT discrimina melhor entre as classes (AUC = 0.88)

seguido pelo classificador MLP combinado com as *embeddings* SG_300_3 (AUC=0.77).

As *embeddings* do BERT quando utilizadas no classificador MLP tiveram desempenho pior do que quando utilizadas no BERT, alcançando segundo lugar entre os modelos testados, com F1-score de 0.634. Este resultado sugere que o processo de *fine-tuning* do BERT tem impacto expressivo no ganho de desempenho observado.

5.1.3 Conclusão e trabalhos futuros

Analisamos os resultados em relação a três aspectos: os parâmetros dos modelos de *embeddings*, parâmetros do classificador e o impacto da mudança de modelo de representação na classificação final. Os resultados mostraram que, embora alterações nos parâmetros janela de contexto e tamanho do vetor dos modelos geradores de *embedding* tenham influenciado os valores de acurácia e F1-score, assim como também os parâmetros da MPL, a maior mudança na faixa de valores de F1 relaciona-se ao modelo de *embedding* selecionado. O destaque foi para o modelo Skip-gram que apresentou o melhor desempenho, seguido das *embeddings* extraídas do BERT. A escolha do modelo de representação teve, portanto, maior impacto no resultado final do que a alteração de qualquer um dos outros hiperparâmetros.

Em uma segunda análise comparamos os modelos e o ganho obtido pelo uso do BERT ajustado para a tarefa em relação ao uso apenas de suas *embeddings* pré-treinadas no classificador MLP. O BERT teve desempenho consideravelmente melhor, com F1-score de 0.875 e AUC da curva ROC de 0.88, em relação ao melhor modelo no MLP, o Skip-gram com F1 de 0.714 e AUC da curva ROC de 0.77. Contudo, observamos que o resultado de suas *embeddings* como entrada no modelo MLP ficou com 0,63 de F1-score, o que indica, portanto, que o *fine-tuning* tem grande impacto no resultado superior obtido para a tarefa.

Outra aspecto notável é que as *embeddings* do BERT, embora mais recentes e dentro de um paradigma de geração de *embeddings* contextuais, não foram superiores às *embedding* fixas do modelo Skip-gram do Word2Vec na tarefa, embora ainda tenham alcançado melhor desempenho do que outras. Isso levanta a questão sobre se o maior ganho do BERT não está na alta especialização para o *dataset* na tarefa, realizado pelo ajuste dos parâmetros durante o *fine-tuning* ao invés de no pré-treinamento de *embeddings*.

Quanto as *embeddings* enriquecidas, apresentaram o pior resultado, contudo nesta análise testamos para uma única configuração de passeios aleatórios no grafo. Como trabalho futuro planejamos a extensão do teste para outros modelos de *embeddings* e para o Português. Também vemos como relevante investigar a questão do ganho de modelos de *embedding* em relação ao *fine-tuning* do BERT expandindo o experimento para outros *datasets*.

5.2 Experimento 2

A noção de que palavras semanticamente semelhantes aparecem no mesmo contexto de palavras no texto, conhecida como hipótese distribucional, motivou o uso de representações vetoriais que consideram coocorrência de palavras como forma de capturar níveis de significado da língua escrita, entre elas *embeddings*. Segundo esta ideia, relações semânticas poderiam ser medidas por proximidade de *embeddings* no espaço vetorial.

Neste segundo experimento, testamos *embeddings* de forma intrínseca para duas relações, sinônimos e antônimos. Nossa motivação foi verificar duas hipóteses: A primeira, se de fato palavras consideradas sinônimas sob um ponto de vista humano teriam seus respectivos vetores mais próximos do que palavras não sinônimas. A segunda, se o valor a similaridade de sinônimos e antônimos seria passivel de distinção.

Por um lado, pela hipótese distribucional apresentada, por terem relações semânticas opostas, os pares vetores de sinônimos deveriam ser distintos dos de antônimos. Por outro, intuitivamente é observável que palavras sinônimas e antônimas podem ocupar o mesmo espaço entre outras palavras (ex: ele é *feio*, ele é *bonito*). Alguns estudos para uma seleção específica de palavras foram feitos nesse sentido, como o realizado por Thalenberg (2016). Aqui procuramos avaliar todos os sinônimos e antônimos presentes nos *corpus* selecionado.

Uma medida comum de ser utilizada como proximidade de *embeddings*, também chamada de similaridade, é relativa ao ângulo formado entre os pares de vetores. Neste trabalho, seguindo a literatura, consideramos como proximidade a distância angular entre *embeddings*.

No contexto deste experimento usaremos *proximidade* e *similaridade* de vetores como termos equivalentes, fazendo referência a medida calculada da distância angular formada entre eles. Também assumiremos a existência de uma percepção coletiva de sinônimos e antônimos de palavras, dada pela anotação manual do dicionário utilizado.

5.2.1 Material e Método

A configuração do experimento pode ser vista na Tabela 10. O processo foi realizado em três passos: geração de vetores de palavras (*word embeddings*), seleção dos pares de sinônimos e comparação da similaridade entre os vetores dos pares.

5.2.1.1 Bosque

Os vetores de palavras foram gerados pelo método Word2Vec para o *corpus* Bosque[7] na versão disponibilizada com anotação de dependência universal (RADEMAKER *et al.*, 2017). O Bosque é parte do *treebank* Floresta Sintática e contém sentenças em Português na variante brasileira e portuguesa.

[7] Disponível em: https://github.com/UniversalDependencies/UD_Portuguese-Bosque

Tabela 10 – Configuração do experimento 2

	Etapa	Configuração
1	*Relação observada*	Sinônimos e Antônimos
2	*Modelo de embedding*	Word2Vec
3	*Medida de Similaridade*	Distância angular
4	*Corpus*	Bosque
5	*Inventário (baseline)*	Dicionário de Sinônimos

Fonte: Autoria própria.

Tabela 11 – Dados da versão do Bosque utilizada

sentenças	9.368
tokens	227.653

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

5.2.1.2 Dicionário de sinônimos

A realização da avaliação foi feita considerando os dados de dois dicionários, um de sinônimos, um de antônimos. Selecionamos para este fim o dicionário de sinônimos online e o dicionário de antônimos online [8], bastante consultados para o português. Ambos são dicionários disponíveis gratuitamente, criados e mantidos pela empresa 7Graus. A equipe responsável é composta por Flávia Neves, lexicógrafa e professora de Português, e Débora Ribeiro, linguista e lexicógrafa, responsáveis desde 2012 pela gestão e produção do conteúdo linguístico disponibilizado nos dicionários [9].

5.2.1.3 Pré-processamento

Os vetores de palavras foram gerados pelo método Word2Vec para o *corpus* Bosque. O *corpus* passou por um pré-processamento no qual, além da mudança para caixa baixa, foram retirados números e caracteres especiais. Em seguida, cada *token*[10] foi lematizado utilizando o lematizador do spaCy[11]. Uma amostra do resultado do lematizador foi analisada, indicando alguns erros, que foram corrigidos, como por exemplo, o lema do artigo "uma" constar como "umr". Como a análise foi realizada em uma amostra e não em todos os lemas, outros erros podem estar presentes. Outra questão percebida é que diversos substantivos que possuem a mesma forma flexionada de um verbo foram lematizado para o lema do verbo (ex: a *fala* indo para *falar*). Embora nem todas as palavras que possuem o mesmo lema tenham de fato o mesmo sentido, optou-se pela lematização para juntar palavras possivelmente com sentidos muito próximos. Para

[8] Disponíveis em: sinônimos.com.br e em antonimos.com.br
[9] Dados disponibilizados no site dos dicionários.
[10] Foram consideradas *tokens* as sequências de caracteres separadas por espaço.
[11] spaCy: https://pypi.org/project/spacy/

efeito deste experimento, não era interessante que palavras como “é” e “são” fossem consideradas distintas. O *corpus* resultante foi usado para a obtenção das *word embeddings* e dos pares de sinônimos.

5.2.1.4 Seleção dos pares de sinônimos e antônimos

Foram selecionados todos os pares de palavras sinônimas presentes no vocabulário do *corpus*. Para esse fim, foi consultado o dicionário de sinônimos online. A seleção dos pares foi feita da seguinte forma: cada palavra p_i do vocabulário P do *corpus* foi consultada no dicionário. Aquelas presentes no dicionário tiveram todos os seus sinônimos consultados de volta no vocabulário P do *corpus*. Para cada sinônimo p_j que também fosse uma palavra no *corpus*, foram gerados pares (p_i, p_j). Processo análogo foi feito para gerar os pares de antônimos. No total foram gerados 28.656 pares de sinônimos e 20.423 de antônimos. O resultado foi um *dataset* de pares de sinônimos e antônimos em que todas as palavras estavam presentes no *corpus*. Características do *corpus* após o processamento e do *corpus* de sinônimos podem ser vistas na Tabela 12.

Tabela 12 – Características do *dataset*

Característica	Quantidade
Vocabulário lematizado	15.520
Vocabulário no dicionário	7.304
Pares de sinônimos	28.656
Pares de antônimos	20.423

Fonte: Autoria própria.

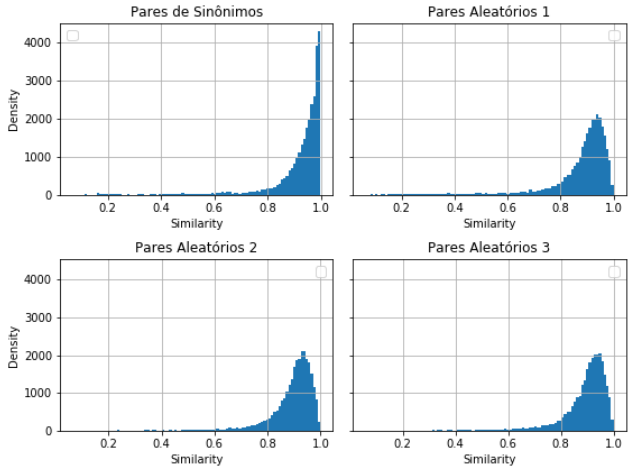
5.2.1.5 Comparação da similaridade entre os vetores dos pares

Para cada par de palavras sinônimas, foi medida a similaridade dos respectivos vetores das palavras do par. Como medida de similaridade foi usada a distância angular entre os vetores. Desta forma, obtivemos um valor representativo de similaridade entre 0 e 1 para cada par, sendo mais próximo de 1 quanto mais similares o par de vetores.

5.2.2 Resultados

A densidade de pares por faixa de similaridade para os pares sinônimos foi comparada com a densidade de similaridade para pares não necessariamente sinônimos de vetores de palavras do *corpus*. Para tanto, foram gerados pares de palavras de forma randômica na mesma quantidade dos pares de sinônimas e suas respectivas similaridades computadas. O histograma dos pares de sinônimos e de mais três variações de pares aleatórios de palavras estão apresentados na Figura 9.

Figura 9 – Comparação da distribuição de similaridade para sinônimos

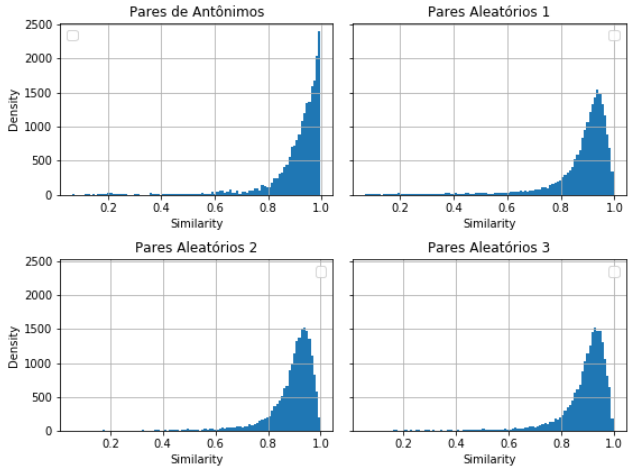


Comparação da distribuição de pares *sinônimos* por faixa de similaridade. O gráfico 1 apresenta a distribuição dos pares sinônimos, os gráficos 2, 3 e 4, apresentam distribuição de pares selecionados aleatoriamente. Fonte: Autoria própria.

Observamos que, para pares de sinônimos, a densidade de similaridade é maior na faixa próxima de 1 com destaque ao pico ocorrendo muito próximo a este valor. Para os pares aleatórios a densidade também se encontra em uma faixa alta de similaridade, estando majoritariamente entre 0.8 e 1.0, mas com pico próximo ao meio deste intervalo. Os valores de similaridades calculados indicaram que os pares de vetores sinônimos apresentaram similaridade geral mais alta.

A mesma análise foi feita para os pares de antônimos. A Figura 10 apresenta os histogramas para estes pares.

Figura 10 – Comparação da distribuição de similaridade para antônimos



Comparação da distribuição de pares *antônimos* por faixa de similaridade. O gráfico 1 apresenta a distribuição dos pares antônimos, os gráficos 2, 3 e 4, apresentam distribuição de pares selecionados aleatoriamente. Fonte: Autoria própria.

Observamos que, da mesma forma apresentada pelos pares de sinônimos, os antônimos também apresentam distribuição concentrada em valores altos de similaridade com pico próximo de 1.

5.2.3 Conclusão e trabalhos futuros

Os pares de *embeddings* apresentaram de forma geral valores altos de similaridade (concentrados na faixa 0.8 - 1.0). Os pares de sinônimos especificamente apresentaram valores de similaridade mais altos, concentrados mais próximos de 1 do que pares de palavras aleatórias. Contudo, tal efeito também foi observado para pares de palavras antônimas, confirmando indicações prévias de que a similaridade vetorial de *embeddings* não reflete a distinção semântica entre sinônimos-antônimos. Como trabalhos futuros sugerimos a investigação destas relações sem a lematização do texto na fase de pré-processamento, e o teste para um *corpus* mais extenso a fim de se verificar se o tamanho seria um fator importante para que sinônimos e antônimos passassem a ser diferenciados pelo contexto de palavras no qual ocorrem.

CAPÍTULO

6

COMBINAÇÃO DE REPRESENTAÇÕES

Neste capítulo exploramos o potencial da combinação de representações sentenciais na tarefa de detecção de similaridade aplicado ao Português. Propomos um modelo de meta-embedding, no qual modelos de *embedding* pré-treinados são combinados de forma supervisionada com foco na tarefa, e avaliamos os resultados dos modelos considerando correlação entre predições, capacidade de generalização e explicabilidade de instâncias. Nossos resultados indicam, entre outras coisas, que uma arquitetura leve de combinação pode superar o uso de modelos sozinhos, e que a predição dos modelos não é explicada pelos tokens das sentenças individualmente.

6.1 Combinação de representações

Modelos de embeddings distintos apresentam desempenho diferente quando testados para tarefa de identificação de similaridade (XU *et al.*, 2018). Uma suposição bastante presente é que isso seja decorrente do fato de que diferentes modelos codificam características distintas do texto (LIU *et al.*, 2021; KIELA; WANG; CHO, 2018), as quais podem ser menos ou mais relevantes dependendo da tarefa e do domínio do corpus trabalhado. Sob está hipótese, propomos e testamos um sistema supervisionado de combinação de modelo de embeddings pré-treinadas comparando o desempenho com os modelos usados sozinhos para similaridade sentencial. Procuramos responder a seguinte pergunta:

- A combinação de modelos de embeddings pré-treinados pode ser utilizada para melhorar o desempenho na tarefa de detecção de similaridade sentencial no Português em relação à utilização de modelos sozinhos?

Para responder esta questão, exploramos a combinação de diversos modelos e contrastamos os resultados com os obtidos pelo uso dos modelos sozinhos. Escolhemos uma arquitetura

leve para realizar a combinação, baseada em transformações lineares, escalável para múltiplos modelos de *embedding* de forma que o foco seja na ponderação das características dos modelos. Especificamente, utilizamos modelos pré-treinados por encoders sentenciais com uma arquitetura baseada em redes siamesas.

Nossos resultados mostram que combinar modelos pré-treinados de embeddings em uma arquitetura leve pode superar os resultados do uso de modelos sozinhos, corroborando a hipótese de que diferente modelos pré-treinados codificam diferentes características do texto.

6.1.1 Motivação

Treinamento não parte do zero: O uso de modelos pré-treinados apresenta vantagens práticas que são incorporadas no modelo de meta-embedding: Não é preciso ter acesso ao corpus fonte usado no treinamento do modelo, o qual muitas vezes não está disponível. Além disso, treinar modelos competitivos do zero é computacionalmente caro.

Seleção de múltiplos domínios: O uso de diferentes modelos de embeddings permite que o sistema atribua pesos de forma automática conforme relevância dos atributos das múltiplas representações para a tarefa ao invés de depender da representação de um único modelo, a qual pode ser sub-ótima. Também possibilita que sejam aproveitadas características provenientes de múltiplos domínios de texto.

Sistema escalável: Não há limite na arquitetura do modelos para o número de modelos de representações que podem ser conectadas no sistema, o que permite que novos modelos possam ser incorporados para o enriquecimento do aprendizado do modelo de meta-embedding.

Modelo mais leve: Fazer fine-tuning de sistemas pré-treinados como o BERT requer a atualização de um grande número de parâmetros (pelo menos 110M para um fine-tuning total do BERT (DEVLIN *et al.*, 2019)). A abordagem que propomos de meta-embedding utiliza uma arquitetura muito mais leve para aprender características específicas da tarefa.

6.2 Conceitos e Métodos

Apresentamos brevemente alguns conceitos e métodos que fizemos uso no experimento.

6.2.1 Encoders

A passagem do texto para uma representação numérica pode ser realizada de múltiplas formas, desde métodos que utilizam contagem de frequência de palavras a métodos de aprendizado de máquina. Para efeito deste trabalho utilizaremos a denominação *encoder* para métodos que mapeiam uma sequência de texto em vetores densos de números reais, independente do algoritmo utilizado. Formalmente, dado uma sequência de texto t, um encoder $\mathscr{F}$ retorna uma

representação vetorial x de dimensão d.

$$\mathscr{F}(t) \rightarrow x, x \in \mathbb{R}^d$$

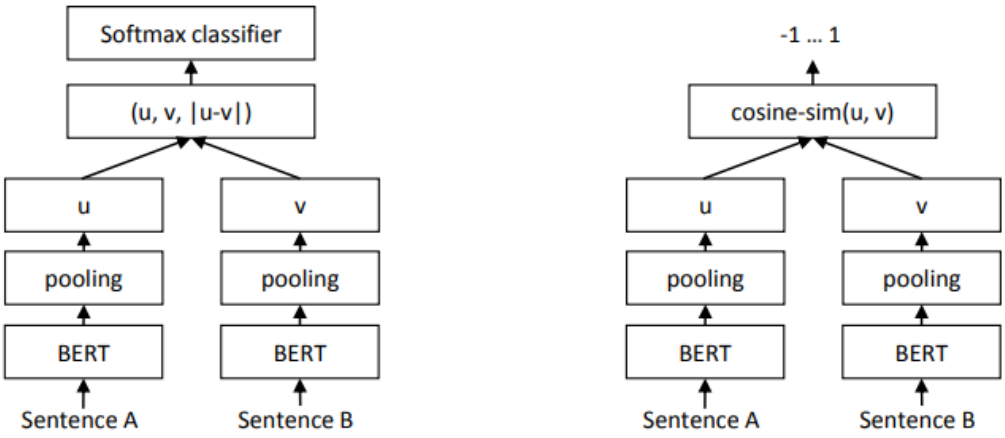
6.2.2 Encoders sentenciais

Para encoders sentenciais a entrada de texto t é uma sentença[1]. Dado um conjunto de sentenças $\mathbb{S}$, um encoder sentencial $\mathscr{F}$ tem por objetivo mapear qualquer sentença $s \in \mathbb{S}$ em um vetor de tamanho fixo d-dimensional. O vetor resultante pode ser gerado tanto por métodos que combinam algebricamente vetores de palavras pré-treinados ou métodos que foram treinados para gerar diretamente representações sentenciais.

6.2.3 Encoders baseados em redes siamesas

Nos últimos anos diversos encoders utilizam redes siamesas em uma arquitetura baseada no SentenceBERT (REIMERS; GUREVYCH, 2019) para derivar representações sentenciais. Nesta arquitetura, cada sentença serve de entrada para um encoder pré-treinado seguido de uma camada de pooling que deriva vetores de tamanho fixo, u e v. O cálculo do pooling pode ser feito de diversas formas, sendo o padrão a média dos vetores. A última camada realiza o cálculo da similaridade de cossenos entre u,v para o caso de tarefas de regressão, ou recebe a concatenação de u,v e $|u-v|$ em um classificador Softmax pra tarefas de classificação, como representado na Figura 11. Embora a arquitetura original utilize o BERT e o RoBERTa como encoders base, a arquitetura é abrangente e na prática outros encoders podem ser utilizados.

Figura 11 – Arquitetura baseada em redes siamesas



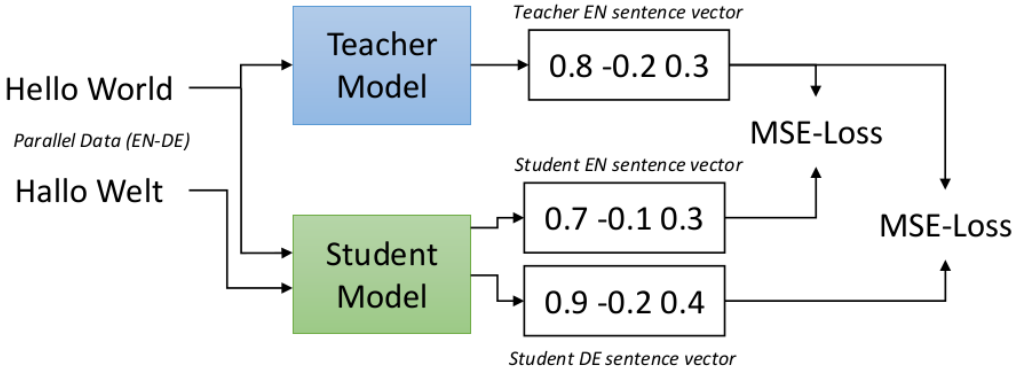
Fonte: (REIMERS; GUREVYCH, 2019)

[1] Observação sobre o conceito de sentença em Base Teórica.

6.2.4 Destilação Multiligue

Os modelos multilingues disponíveis foram gerados por um método de destilação de conhecimento multilíngue (*multilingual knowledge distillation*) (REIMERS; GUREVYCH, 2020). Esta técnica utiliza um modelo monolingual fonte (*teacher*) para gerar modelo multilingual (*student*) treinando um sistema paralelo que mimetiza o modelo original. A ideia é atualizar as representações do modelo *student* em uma determinada língua original aproximando a representação daquela do modelo *teacher* enquanto também atualiza a correspondente representação em uma nova língua. O processo é representado na Figura 12.

Figura 12 – Esquema do processo de destilação de conhecimento multilíngue



Fonte: (REIMERS; GUREVYCH, 2020)

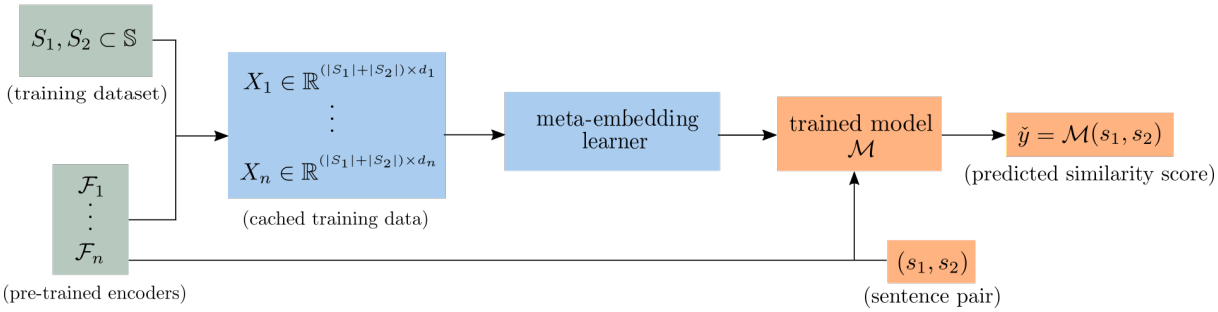
6.2.5 Modelo meta-embedding para similaridade

Dado um conjunto de n encoders pré-treinados no nível sentencial $F = \{\mathscr{F}_1, \mathscr{F}_2, \mathscr{F}_3, ..., \mathscr{F}_n\}$, um conjunto de sentenças $\mathbb{S}$ e um par $p = (s_1, s_2)$, tal que $s_1, s_2 \in \mathbb{S}$, o algoritmo de meta-embedding supervisionado aqui denominado por M tem por objetivo mapear as n representações do par dadas por $((\mathscr{F}_1(s_1), \mathscr{F}_1(s_2)), ..., (\mathscr{F}_n(s_1), \mathscr{F}_n(s_2))$ em um score de similaridade $\check{y}$.

$$M(F, p) \rightarrow \check{y}(p)$$

As etapas do processo são apresentadas na Figura 13, com S_1 e S_2 representando o conjunto de sentenças dos pares de treino e X as representações dos pares para cada encoder.

Figura 13 – Etapas do aprendizado supervisionado de meta-embedding para similaridade



Cores representam entrada, treinamento e predição.
Fonte: Figura de autoria própria inspirada em Poerner, Waltinger e Schütze (2019).

6.2.6 Explicabilidade das predições

Modelos supervisionados de aprendizado de máquina em PLN costumam ser avaliados por métricas que comparam os valores preditos com o o esperado, como F1, acurácia e revocação no caso de modelos de classificação e MSE e correlações no caso de modelos de regressão. No entanto, tais métricas não esclarecem quais características do texto o modelo efetivamente utiliza para realizar uma predição. Com o intuito de inspecionar o impacto de cada token do texto de entrada em relação à predição do modelo, fizemos uma extensão do LIME (RIBEIRO; SINGH; GUESTRIN, 2016) [2], algoritmo de explicabilidade de instâncias, e aplicamos ao nosso modelo.

6.2.6.1 LIME e extensão

LIME, acrônimo de *Local Interpretable Model-agnostic Explanations*, é um método desenvolvido para explicar predições de modelos não intuitivos, como modelos de predição baseados em redes neurais. Dado um modelo preditivo e uma instância de entrada, a proposta do LIME é explicar a predição do modelo para aquela instância usando o impacto no resultado gerado por perturbações na mesma através de uma aproximação linear.

No módulo de texto do sistema, são geradas variações da instância de entrada por um processo em que um ou mais *tokens* são aleatoriamente retirados. Através da distância das variações com a instância original e a predição do modelo para as mesmas são obtidas relações entre os *tokens* e a predição do modelo.

Estendemos o sistema original para receber mais de um texto como entrada e aplicá-lo no caso de regressão (para texto, o LIME original é preparado somente para modelos de classificação com um texto de entrada). Na prática, para o caso no qual trabalhamos com nossa extensão, a instância é um par de sentenças e a predição um valor de similaridade. São geradas variações do par e computados os respectivos valores de similaridade para cada um.

[2] Disponível em:
original: https://github.com/marcotcr/lime
estendido: https://github.com/anasampa/lime

Uma desvantagem do LIME é que o sistema fornece uma avaliação para cada instância individualmente, sendo um desafio a generalização dos resultados para o todo do corpus analisado. Por outro lado, uma vez que o processo independe dos mecanismos utilizados pelo modelo para predizer o resultado, ele é agnóstico ao modelo, podendo ser utilizado para modelos "black box" como forma de avaliar onde o modelo "olha" para produzir a predição.

6.3 Experimento

Mostramos a seguir o experimento propriamente dito, passando pelas configurações, resultados e a avaliação das predições.

6.3.1 Ferramentas e métodos

6.3.1.1 Modelos pré-treinados

Para avaliação de combinação de representações, selecionamos cinco modelos de embeddings pre-treinadas, indicados na Tabela 13. Os modelos foram pré-treinados seguindo a arquitetura de redes siamesas do SentenceBERT e estão disponíveis no Hugging Face[3].

Foram priorizados modelos multilingues, uma vez que nossa aplicação é no Português.

Tabela 13 – Modelos sentenciais pré-treinados

denominação	modelo	dimensão
modelo 1	paraphrase-multilingual-mpnet-base-v2	768
modelo 2	paraphrase-multilingual-MiniLM-L12-v2	384
modelo 3	distiluse-base-multilingual-cased-v1	512
modelo 4	distiluse-base-multilingual-cased-v2	512
modelo 5	all-mpnet-base-v2	768

Fonte: Autoria própria.

6.3.1.2 Arquitetura

Considerando o foco na combinação de atributos por cada modelo de embedding e visando uma arquitetura leve do ponto de vista de demanda computacional, adotamos uma arquitetura direcionadas para a busca de relações lineares entre os modelos como apresentado no esquema da Figura 14.

Inputs: Os encoders são alimentados com os pares de sentenças do domínio, gerando as respectivas representações embeddings. $\mathscr{F}_i(s_a) \rightarrow x^i_{s_a}, a \in \{1,2\}, 1 \leq i \leq n$.

[3] Modelos disponíveis em: https://huggingface.co/sentence-transformers

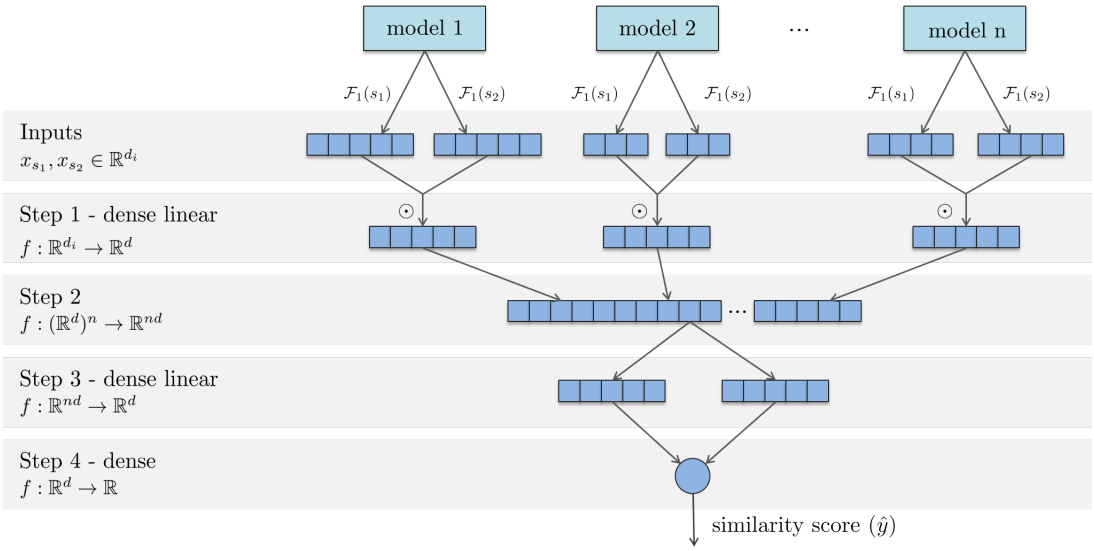
Etapa 1: Para cada modelo o produto de Haddamard $x^i_{s_1} \odot x^i_{s_2}$ é mapeado para um vetor d-dimensional. Diferentes encoders podem gerar representações de diferentes tamanhos. Esta etapa uniformiza o tamanho dos múltiplas representações.

Etapa 2: Os n vetores são concatenados.

Etapa 3: O vetor nd-dimensional serve como entrada à duas transformações paralelas, cada uma resultando em uma saída de tamanho d.

Etapa 4: Camada densa com função de ativação Sigmoid. A saída é um valor real entre 0 e 1 (O valor $\hat{y}$ predito de similaridade entre as sentenças).

Figura 14 – Arquitetura do modelo proposto de meta-embedding



Fonte: Autoria própria.

6.3.1.3 *Métrica de avaliação e baselines*

Seguindo trabalhos anteriores, como métrica de avaliação usamos o Erro Quadrático Médio (MSE), a correlação de *Pearson* e a correlação de *Spearman*.

A correlação de *Person* não é considerada uma boa métrica de avaliação de resultados de detecção de similaridade (REIMERS; BEYER; GUREVYCH, 2016), uma vez que as predições podem ser próximas dos valores reais (anotação) e não linearmente correlacionadas. Contudo, uma vez que largamente utilizada, optamos por mantê-la para permitir comparação entre outros sistemas no mesmo corpus.

Tomamos como baselines o valores de similaridade de cada modelo individualmente calculado a partir da similaridade de cossenos, considerando que nosso objetivo é a avaliação comparativa entre o desempenho dos modelos sozinhos em relação a combinação de modelos.

6.3.1.4 Datasets

Utilizamos dois datasets anotados para similaridade sentencial no Português, o ASSIN1 (FONSECA *et al.*, 2016) para os testes gerais e o ASSIN2 (REAL; FONSECA; OLIVEIRA, 2020) na avaliação de generalização dos modelos, apresentados no Capítulo 4. Ambos foram manualmente anotados com valores que variam de 1 a 5, sendo 1 sentenças com mensagens diferentes e 5 sentenças entendidas como tendo sentidos equivalentes. O ASSIN1 possuí 10.000 pares de sentenças e o ASSIN2, 9.448. Realizamos os testes na mesma partição de teste sugerida pelos organizadores dos datasets, sendo composto por 3000 pares para o primeiro e 2.448 para o segundo.

6.3.2 Abordagens anteriores

Apresentamos na Tabela 14 um resumo de resultados anteriores no ASSIN1.

Tabela 14 – Resultado de outros sistemas - ASSIN1

Sistema		**MSE**	**Pearson**	**Spearman**
Outros sistemas				
(SOUZA *et al.*, 2019)		0.56	0.66	
(FREIRE; PINHEIRO; FEITOSA, 2016)		0.59	0.62	
(HARTMANN, 2016)		0.52	0.68	
(BARBOSA *et al.*, 2016)		0.59	0.63	
(ALVES; RODRIGUES; OLIVEIRA, 2016)		0.57	0.65	
(PINHEIRO *et al.*, 2017)		0.47	0.70	
BERT como *feature base*				
(FIALHO; COHEUR; QUARESMA, 2020)	ptBERT-Large (linear SVM)	0.43	0.76	0.76
(FIALHO; COHEUR; QUARESMA, 2020)	ptBERT-Base (linear SVM)	0.42	0.76	0.76
(FIALHO; COHEUR; QUARESMA, 2020)	mBERT-Base (linear SVM)	0.42	0.77	0.76
(FIALHO; COHEUR; QUARESMA, 2020)	mBERT-Base (rbf SVM)	0.41	0.77	0.76

6.3.3 Abordagem com fine-tuning

A abordagem de realizar *fine-tuning* em encoders pré-treinados tem se demonstrado bem-sucedida em diversas tarefas, incluindo detecção de similaridade. O atual estado da arte para ambos os datasets foi obtido no ptBERT (FIALHO; COHEUR; QUARESMA, 2020; SILVA; CASELI, 2021).

Nesta seção, apresentamos na Tabela 15 os resultados com aplicação de *fine-tuning* no BERT pré-treinad no Português (BERTpt) e no BERT multilingual (mBERT). Adicionalmente realizamos o *fine-tuning* do SentenceTransformer para os cinco modelos pré-treinados que utilizamos, gerando resultados que, até onde sabemos, não foram obtidos para o ASSIN1 antes.

Tabela 15 – Resultados com abordagem *fine-tuning*

System	MSE	Pearson	Spearman
BERT *fine tuning*			
ptBERT-Base	0.36	0.82	0.81
mBERT-Base	0.36	0.80	0.78
SentenceTransformer *fine-tuning*			
model 1	0.02	0.79	0.78
model 2	0.42	0.78	0.77
model 3	0.40	0.78	0.76
model 4	0.41	0.78	0.77
model 5	0.49	0.72	0.71

6.4 Resultados

Testamos o modelo de meta-embedding para diferentes combinações dos seis modelos sentenciais pré-treinados. Os resultados são descritos na Tabela 16.

Tabela 16 – Resultados - ASSIN1

modelo pré-treinado	(PT-PT+PT-BR)			(PT-PT)			(PT-BR)		
embedding model	MSE	Pearson	Spearman	MSE	Pearson	Spearman	MSE	Pearson	Spearman
Baselines									
model 1	0.94	**0.71**	**0.71**	1.23	**0.74**	**0.74**	0.65	**0.73**	**0.72**
model 2	0.85	0.68	0.68	1.10	0.72	0.72	0.60	0.69	0.68
model 3	**0.57**	**0.71**	0.70	**0.74**	**0.74**	**0.74**	**0.40**	0.72	0.70
model 4	0.66	0.69	0.68	0.87	0.72	0.72	0.46	0.70	0.68
model 5	1.68	0.52	0.51	2.04	0.54	0.54	1.32	0.52	0.51
Meta-embedding									
models 1, 2	0.55	0.65	0.64	0.67	0.66	0.65	0.42	0.68	0.66
models 2, 3	0.45	0.73	0.72	0.56	0.76	0.75	0.35	0.74	0.71
models 1, 3	0.45	0.74	**0.73**	0.57	0.77	0.76	**0.33**	**0.76**	**0.73**
models 3, 4	0.57	0.73	0.71	0.74	0.76	0.75	0.40	0.74	0.71
models 1,2,3	**0.43**	**0.75**	**0.73**	**0.54**	**0.78**	**0.77**	**0.33**	**0.76**	**0.73**
models 1,2,3,4,5	0.47	**0.75**	**0.73**	0.58	0.72	0.71	0.38	0.72	0.70

O melhor modelo de meta-embedding (dado pela combinação dos modelos 1+2+3) superou os modelos individuais para todas as partições do dataset em todas as métricas de avaliação. Em particular, em relação ao MSE, mesmo o pior resultado de meta-embedding (a combinação dos modelos 1+2) superou os resultados dos baselines. Adicionalmente, destacamos abaixo algumas observações:

- No geral o melhor modelo de meta-embedding foi conseguido combinando-se os modelos 1,2 e 3. Curiosamente, a combinação dos modelos 1 e 2 teve o pior desempenho antre os modelos de meta-embedding.
- A combinação dos cinco modelos produziu o segundo pior resultado com relação ao MSE. Se consideradas as correlações de *Pearson* e *Spearman*, obteve desempenho competitivo

no todo (PT-PT+PT-BR), mas figura como segundo pior nas partições separadas da língua (PT-PT) e (PT-BR).

- Os modelos 3 e 4 possuem o melhor MSE dos modelos individuais (são os dois valores mais baixos). Contudo, a combinação de ambos, modelo de meta-embedding 3+4, teve o pior MSE entre os modelos de meta-embedding, e ficou como segundo pior com relação a correlação de *Pearson* e *Spearman*.

A partir das observações, destacamos os seguintes pontos gerais:

(A) A combinação de modelos não segue um ganho linear.

(B) Quantidade não é qualidade. Introduzir mais modelos ao sistema não necessariamente traz ganho aos resultados.

(C) A combinação dos melhores modelos individuais não produz necessariamente a melhor combinação.

(D) Uma combinação favorável para um dataset não é automaticamente favorável a outro.

Temos como hipótese que ocorre uma sobreposição das características codificadas nos diversos modelos e, portanto, uma interseção no escopo de influência no aprendizado dos valores de similaridade. Ou seja, mais de um modelo está contribuindo da mesma forma, realizando o mesmo trabalho. Para uma melhor análise, na próxima seção avaliamos as predições dos modelos.

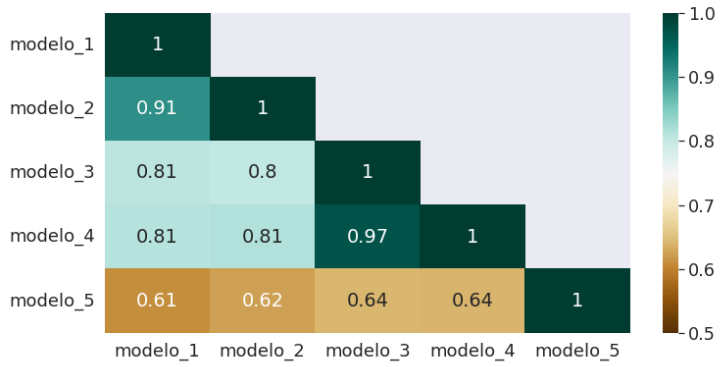
6.5 Avaliação das predições

Nesta seção avaliamos as predições dos modelos com relação a correlação de resultados, capacidade de generalização para outro *dataset* e explicabilidade de predições de instâncias com base nos *tokens* do texto.

6.5.1 Análise de correlação entre predições

Para investigar sobreposição de modelos, comparamos as predições para similaridade dos modelos individualmente (baselines). Para tanto, utilizamos a correlação de *Pearson* para observar as correlação lineares entre as predições. O resultado é apresentado na Figura 15.

Figura 15 – Mapa de correlação das predições de cada modelo



Fonte: Autoria própria.

Os modelos 1 e 2 apresentam alta correlacão (0.91), indicando que ambos predizem valores de similaridades correlacionados. Esta relação está em concordância com os resultados do modelo de meta-embedding que mostraram que combinar 1+2 ajuda pouco na melhora de desempenho comparativamente à combinação de outros modelos.

O mesmo pode ser observado com relação aos modelos 3 e 4, que também tem grande correlação (0.97). Como indicado nos resultados, apesar de terem desempenho no topo entre os baselines, a combinação de ambos não trouxe muito ganho de desempenho, indicando a sobreposição de acertos do modelo.

Dado um par como entrada, investigamos quais tokens impactam positivamente ou negativamente as predições dos modelos e se este impacto difere entre os modelos. Para este propósito, utilizamos nossa extensão do LIME apresentada no seção 6.2.6 para receber pares de texto como entrada e realizar regressão.

6.5.2 Capacidade de Generalização

Tipicamente no trabalho de aprendizado de máquina, a avaliação do modelo é realizada em uma partição do próprio dataset que foi separada das demais. Este procedimento garante que o teste seja feito em dados desconhecidos ao modelo, ou seja, que não foram utilizados no processo de treino e validação. No entanto, por este método, estes dados tem grande chance de terem propriedades semelhantes aos dados de treino. Em um caso real, o esperado é que a predição do modelo ocorra em novos dados de entrada, os quais não necessariamente compartilham das mesmas propriedades. Idealmente o desejado é que o algorítimo treinado seja capaz de generalizar para novos dados de entrada. Nesta seção testamos a capacidade de generalização dos modelos avaliando o desempenho com dados cruzados entre os corpus. Procedemos da seguinte forma, avaliamos os dois melhores modelos treinado no ASSIN1 nos dados de teste do ASSIN2, bem como o oposto. Os resultados em comparação com o BERT-pt e o BERT

multilíngue encontram-se nas Tabela 17 e 18.

Tabela 17 – Generalização ASSIN1 para ASSIN2

Modelo	MSE	Pearson	Spearman
modelo1,2,3	0.51	0.71	0.67
modelo 1,2,3,4,5	0.52	0.72	0.67
BERT com *fine-tuning*			
BERTpt-large	0.48	0.77	0.72
BERTpt-base	0.48	0.71	0.63
BERT-base	0.64	0.66	0.59

Tabela 18 – Generalização ASSIN2 para ASSIN1

modelo	MSE	Pearson	Spearman
modelo 1,2,3	0.82	0.65	0.65
modelo 1,2,3,4,5	1	0.66	0.66
BERT com *fine-tuning*			
BERTpt-large	0.96	0.68	0.7
BERTpt-base	1.28	0.62	0.66
BERT-base	0.8	0.71	0.71

A partir dos resultados observamos que a generalização do ASSIN1 para o ASSIN2 apresenta desempenho melhor do que o contrário. Além disso, o BERTpt generalizou melhor para o primeiro caso (ASSIN1 para ASSIN2), enquanto o BERT multilingual generalizou melhor para o caso contrário. Outro ponto a ser notado é que a diferença entre o desempenho dos modelos de *deep learning* ajustados (*fine-tuning*) e os modelos com arquitetura mais simples de combinação de *embeddings* diminuiu. Embora ambos *datasets* sejam anotados para a mesma tarefa, as características de ambos são distintas. A seguir comparamos o ASSIN1 e o ASSIN2 quanto às suas propriedades.

6.5.2.1 Avaliação da distinção entre os datasets

As sentenças do ASSIN1 foram retiradas de notícias de jornal para duas variantes da língua, Português brasileiro e Português europeu. O gênero de todas é, portanto, jornalístico. Quanto ao tipo textual, se caracteriza por mescla de narração, exposição e eventualmente injunção. Estão presentes diversas entidades nomeadas (nomes de pessoas públicas em geral como esportistas, políticos e artistas além de instituições e lugares).

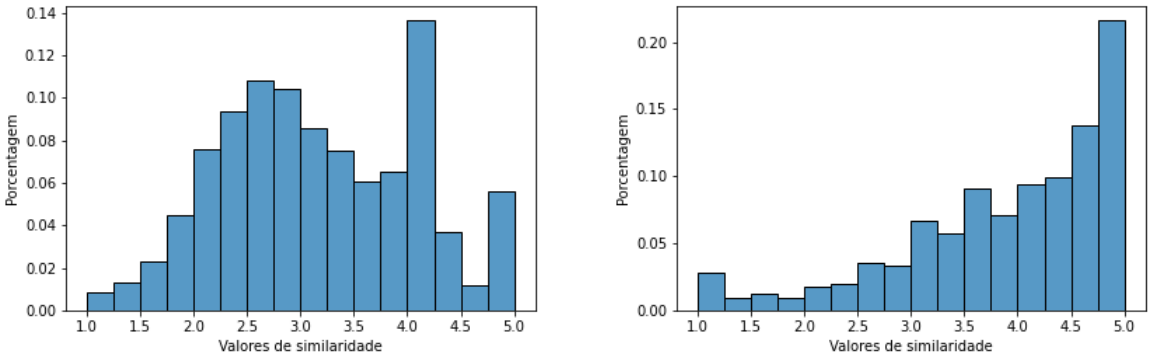
As sentenças do ASSIN2 são provenientes do SICK-BR, corpus traduzido do Inglês para o Português brasileiro. No Inglês foram originadas a partir de descrições de eventos observados em imagens. Descrevem uma situação sem necessariamente haver sequência temporal (as sentenças usam o verbo no presente), tendo um tipo textual de exposição de um evento. Não há entidades nomeadas.

Nos pares do ASSIN1 existem poucas repetições de sentenças, enquanto no ASSIN2 é maior a concentração de sentenças que se repetem em pares distintos. A distribuição dos *datasets* em relação aos valores de similaridade também é distinta, como mostrado na Figura 16.

Quanto a estrutura das sentenças, o ASSIN2 segue um padrão regular, havendo alternância entre voz passiva e ativa e afirmação e negação para sentenças com o mesmo vocabulário. Diferentemente o ASSIN1 é mais complexo do ponto de vista de repetição de padrão.

A partir destas características, levantamos a hipótese que o desempenho inferior da generalização do ASSIN2 para o ASSIN1 em relação ao contrário seja consequência das caracte-

Figura 16 – Distribuição dos pares em relação aos valores de similaridade



ASSIN1 ASSIN2

Fonte: Autoria própria.

rísticas textuais mais diversas e complexas do segundo em relação ao primeiro.

6.5.3 Influencia dos tokens na predição dos modelos

Utilizamos a extensão do LIME aplicado ao modelo de meta-embedding para avaliar o impacto dos *tokens* das sentenças na predição. Na tabela Tabela 19 apresentamos algumas das instâncias analisadas com os respectivas indicações de impacto positivo (laranja) ou negativo (azul) nas predições. O impacto positivo significa que pela analise do LIME os *tokens* causam aumento dos valores de similaridade preditos, enquanto *tokens* de impacto negativo causam a diminuição dos valores. Palavras não marcadas são neutras quanto ao impacto na predição.

Foram considerados três casos para avaliação: *tokens* iguais (*tokens* que são exatamente iguais nas duas sentenças), *tokens* equivalentes (distintos porém com mesmo papel na sentença em termos de sentido) e *tokens* dissociados (distintos e sem conexão de sentido):

***tokens* iguais:** Nas instâncias analisadas é possível perceber a prevalência de impacto positivo em palavras que aparecem nas duas sentenças do par, sendo ainda mais forte essa prevalência para palavras consideradas *content words*.

***tokens* equivalentes** Quanto aos *tokens* equivalentes, notamos que em alguns casos o impacto de ambas é igual e positivo (*construção* - *construir*), em outros, distintos (*apenas* - *somente*, *manifestações* - *casos*) ou sem impacto para alguma delas (*mencionaram* - *responderam*).

***tokens* dissociados:** No geral os *tokens* dissociados aparecem sem impacto ou com impacto negativo.

Tabela 19 – Impacto dos tokens na predição para o ASSIN1 (modelo 1,2,3)

Polaridade nas predições	Similaridade
Tenho orgulho de ter feito parte da construção do PSOL. Ajudei a construir o PSOL, e disso muito me orgulho.	gold = 1.0 predito = 0.66
Foi uma daquelas corridas onde as coisas não dão certo. Foi uma daquelas corridas em que as coisas não estavam acontecendo direito.	gold = 1.0 predito = 0.69
Prometi ao estúdio que entregaria uma última trilogia para terminar a saga. Prometi ao estudo que faria uma última trilogia para finalizar a saga.	gold = 1.0 predito = 0.68
Esta proposta lida com o persistente aumento nos gastos ao longo dos anos. Esta proposta enfrenta o persistente aumento de despesas por anos.	gold = 1.0 predito = 0.73
Apenas 5% mencionaram ficar desconectados a maior parte do tempo. Somente 5% responderam que permanecem desconectados a maior parte do tempo.	gold = 1.0 predito = 0.72
O valor foi enviado ao exterior por meio de empresas offshores localizadas no Uruguai e na Suíça. O dinheiro tinha sido enviado ao exterior através de empresas offshores no Uruguai e Suíça.	gold = 1.0 predito = 0.84
Assim, num Universo infinito, têm de existir outras manifestações de vida. Em um universo infinito, devem existir outros casos de vida.	gold = 1.0 predito = 0.69

Resultados do LIME para instâncias do ASSIN1, com similaridade em escala de 0 a 1. Indicação de cores: Laranja para tokens que afetam positivamente a predição do modelo, azul para tokens que afetam negativamente. Fonte: Autoria própria.

Outra análise que realizamos foi a comparação do impacto dos *tokens* entre os modelos. Na Tabela 20 indicamos o impacto na predição de instâncias para modelos distintos. Observamos que o impacto ocorre de forma semelhante na predição dos modelos, ou seja, os diversos modelos consideram a mesma polaridade para a maioria dos *tokens* das sentenças, com prevalência forte dos mesmos *tokens* positivos. Isto ocorre mesmo em casos cujos valores preditos são distintos entre os modelos, como no modelo 1 e 3 do *par d*, para o qual as predições foram 0.92 e 0.75 respectivamente. Existem portanto outros fatores influenciando nos valores das predições. Uma possível explicação é que as predições não são provenientes exclusivamente dos *tokens* individualmente, mas que os modelos captam combinações.

Tabela 20 – Comparação da polaridade entre de diferentes modelos

par a – gold standard = 1.0
modelo 1 – valor predito = 0.88

O Corinthians vai ficar até dois meses sem Rildo
Rildo desfalcará o Corinthians por um período de até dois meses

modelo 2 – valor predito = 0.85

O Corinthians vai ficar até dois meses sem Rildo
Rildo desfalcará o Corinthians por um período de até dois meses

modelo 3 – valor predito = 0.85

O Corinthians vai ficar até dois meses sem Rildo
Rildo desfalcará o Corinthians por um período de até dois meses

modelo 4 – valor predito = 0.86

O Corinthians vai ficar até dois meses sem Rildo
Rildo desfalcará o Corinthians por um período de até dois meses

modelo 5 – valor predito = 0.85

O Corinthians vai ficar até dois meses sem Rildo
Rildo desfalcará o Corinthians por um período de até dois meses

par b – gold standard = 1.0

modelo 1 – valor predito = 0.90

Ele acabou sendo aplaudido pela maioria da plateia, apesar de vaias terem sido registradas
Ele foi bastante aplaudido pela plateia, mas, também recebeu algumas vaias

modelo 2 – valor predito = 0.91

Ele acabou sendo aplaudido pela maioria da plateia, apesar de vaias terem sido registradas
Ele foi bastante aplaudido pela plateia, mas, também recebeu algumas vaias

modelo 3 – valor predito = 0.71

Ele acabou sendo aplaudido pela maioria da plateia, apesar de vaias terem sido registradas
Ele foi bastante aplaudido pela plateia, mas, também recebeu algumas vaias

modelo 4 – valor predito = 0.75

Ele acabou sendo aplaudido pela maioria da plateia, apesar de vaias terem sido registradas
Ele foi bastante aplaudido pela plateia, mas, também recebeu algumas vaias

modelo 5 – valor predito = 0.80

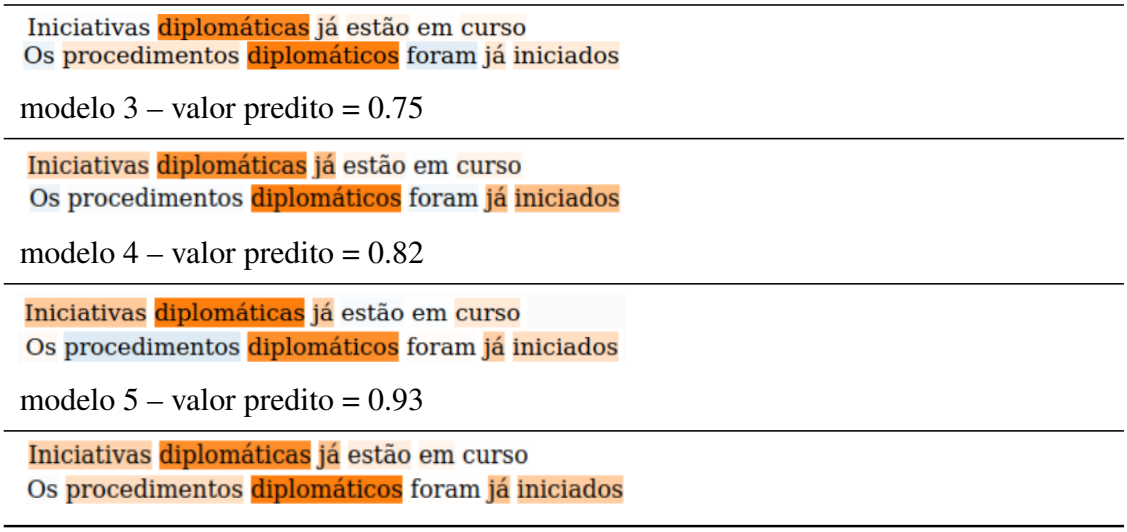
Ele acabou sendo aplaudido pela maioria da plateia, apesar de vaias terem sido registradas
Ele foi bastante aplaudido pela plateia, mas, também recebeu algumas vaias

par d – gold standard = 1.0

modelo 1 – valor predito = 0.92

Iniciativas diplomáticas já estão em curso
Os procedimentos diplomáticos foram já iniciados

modelo 2 – valor predito = 0.90



Resultados do LIME para instâncias do ASSIN1 considerando-se diferentes modelos. Predição em escala de 0 a 1. Indicação de cores: Laranja para tokens que afetam positivamente a predição do modelo, azul para tokens que afetam negativamente. Fonte: Autoria própria.

6.6 Conclusão e trabalhos futuros

Neste experimento testamos o modelo de meta-embedding proposto na tarefa de detecção de similaridade sentencial no Português, comparando o desempenho entre os modelos individualmente e seu uso de forma combinada. Nossos resultados mostram que a combinação de *embeddings* com o modelo de meta-embedding supera o desempenho dos modelos individualmente. Ademais, indicam que a seleção de modelos voltada à melhora de resultados não é trivial, uma vez que associam-se ao escopo de cada modelo. Assumimos isto como evidência indireta de que *embeddings* de fato codificam diferentes informações do texto e que nossa abordagem pode ser utilizada para melhorar os resultados de predição de similaridade.

CAPÍTULO

7

CONSIDERAÇÕES FINAIS E CONCLUSÃO

Neste trabalho avaliamos o uso de modelos de *embeddings* na tarefa de similaridade sentencial. Em uma primeira etapa, testamos os resultados de diferentes modelos de *embeddings* com relação a variação de parâmetros utilizados por um MLP. Contrastamos também o uso de *embeddings* do BERT como *feature based* em relação ao ajuste para a tarefa na abordagem *fine-tuning*. Na segunda parte focamos na tarefa para o Português. Trabalhamos com modelos pré-treinados sentenciais baseados em redes siamesas, comparando diferentes combinações de modelos em relação ao uso de modelos unitários. Avaliamos os fatores envolvidos na escolha direcionada de modelos para serem combinados e a generalização do treinamento para um *dataset* de outro domínio. No geral testamos diferentes modelos de *embedding*, entre eles fixo, contextual, enriquecido e sentencial. Apresentamos neste capítulo as considerações finais e as principais conclusões advindas do trabalho.

7.1 Considerações gerais e conclusões

- **Similaridade, conceito e tarefa computacional:** O conceito do que é similaridade não é universal e tem um papel importante com relação à tarefa de STS, uma vez que duas coisas serem ou não similares não é uma propriedade intrínseca das coisas, mas depende de uma interpretação de atributos, como abordado no Capítulo 3 - Seção 3.2. Nesta seção concluímos que o problema da detecção de similaridade fica estruturado em dois fatores: critérios e atributos. Na tarefa computacional o primeiro está na anotação dos dados e o segundo depende das características presente nas representações. Portanto, como hipótese do ponto de vista teórico, a forma da fazer a passagem do texto para representações numéricas não é secundaria à tarefa, mas determinante, uma vez que os atributos necessários para a similaridade almejada podem não estar representados.

- **Relevância dos modelos de representação** Examinamos o impacto entre diferentes mo-

delos de representação vetorial *embedding* e seus respectivos parâmetros de geração nos resultados na tarefa de identificação de perguntas equivalentes no *Quora Question Bench-Mark* (QQB) em relação aos hiperparâmetros de classificação (Capítulo 5). Utilizamos dois modelos para classificação, Multi-Layer Perceptron (MLP) e BERT acoplado a uma camada adicional no modo *fine-tuning* para a tarefa , e quatro modelos de *embedding*, Skip-gram do Word2Vec, Doc2Vec, BERT *embeddings* e um modelo próprio desenvolvido para o trabalho, enriquecido com informações sintáticas. Variamos os hiperparâmetros em uma combinação que totalizou 273 configurações de experimentos testados. Como conclusão, a mudança do modelo de *embedding* teve maior impacto na faixa de valores dos resultados do que qualquer outro parâmetro alterado, indicando na prática que a forma em que é feita a passagem do texto para representações numéricas tem papel fundamental no desempenho resultante.

- **Combinação de representações:** No Capítulo 6 - Seção 6.4 propomos uma arquitetura de combinação de *embeddings* para tarefa de similaridade sentencial. Testamos cinco modelos de representações pré-treinadas e 6 combinações dos mesmos na tarefa de STS no Português. Os resultados mostraram que a combinação de representações em uma arquitetura simples é uma estratégia eficiente para melhorar os resultados na tarefa de STS no Português em relação a utilização de modelos sozinhos. Esta estratégia ajustou um número de parâmetros durante o treino consideravelmente menor do que o *fine-tuning* de grandes modelos como o ptBERT (para os cinco modelos o sistema de combinação ajustou menos de 10 vezes a quantidade de parâmetros ajustadas no ptBERT), sendo computacionalmente mais leve. Quanto ao desempenho, ficou abaixo do ptBERT no modo *fine-tuning*, contudo há espaço para que outras combinações modelos de *embeddings* sejam explorados no sistema. Levando em conta um ambiente dinâmico, em que possivelmente ocorra retreinamento de tempos em tempos para atualização com novas entradas, demanda computacional e tempo de treinamento podem ser fatores determinantes na escolha do sistema para STS.

- **Generalização:** A prática de averiguar desempenho de algoritmos de aprendizado supervisionado em partições de um mesmo *dataset* indica a capacidade do algoritmo especificamente naquele *dataset*, uma vez que treino e teste compartilham características textuais. Estas características podem não estar presentes em outros textos, e por isso esta prática não garante a generalização do desempenho alcançado para novas entradas externas, propriedade desejada nas aplicação práticas.
  No Capítulo 6 - Subseção 6.5.2 testamos dois modelos para STS no Português, ptBERT e nosso modelo de combinação de *embeddings* para além do *dataset* original. Treinamos no ASSIN1 e testamos no ASSIN2, bem como o contrário. Os resultados indicam que o desempenho ptBERT, estado da arte em STS para os *dataset* ASSIN1, caiu quando aplicado no fora do *dataset* de original, indicando superespecialização do ptBERT para

as propriedades do *dataset* original. Considerando um ambiente aplicado, para mitigar esta questão, um caminho é o retreino constante do sistema com novas entradas. Contudo esta prática cobra seu preço em demanda computacional, uma vez que o *fine-tuning* de grandes sistemas como o BERT envolve a atualização milhões de parâmetros (DEVLIN *et al.*, 2019).

7.2 Ferramentas e recursos

Durante este projeto de mestrado, também foram desenvolvidas e disponibilizadas ferramentas computacionais para combinação de *embeddings*, explicabilidade para a tarefas STS, bem como geração de metadados adicionais para o dataset ASSIN1. A seguir, tais ferramentas e recursos são descritos em mais detalhes:

- **Extensão do LIME:** Para avaliação do nosso modelo quanto ao impacto dos *tokens* na predição fizemos uma extensão do modelo de explicabilidade LIME, permitindo que esse seja utilizado para modelos preditivos cujas entradas são pares de sentenças.
- **Modelo meta-embedding**: Propomos um modelo de combinação de *embedding* que combina diferentes modelos pré-treinados sentenciais. Deixamos o modelo disponível para uso de forma amigável como biblioteca, na qual o usuário pode escolher entre os modelos do SentenceTransformer quais modelos pré-treinados deseja combinar. Na biblioteca também está extensão do LIME adaptada ao modelo de forma que o usuário pode aplicá-la para seu modelo de combinação. Deixamos disponível em: <https://github.com/anasampa/metaembedding>.
- **Anotação adicional do ASSIN1:** Durante o trabalho, com objetivo de analisar possíveis correlações entre as predições dos modelos e atributos textuais do dataset ASSIN1, realizamos a anotação da partição de teste para entidade nomeada, tópico, semelhança sintática no primeiro nível, entre outras medidas descritas em mais detalhes no anexo. O *dataset* com as anotações está disponível em: <https://github.com/anasampa/assinplus>

7.3 Publicações

Os resultados obtidos com o desenvolvimento deste trabalho permitiram a submissão de um trabalho como autora principal em conferência internacional (aceito para publicação), bem como co-autoria e colaboração em outro trabalho publicado em conferência internacional.

– **Rodrigues, A.C.**, Marcacini R. M. *Sentence Similarity Recognition in Portuguese from Multiple Embedding Models.*. (aceito na ICMLA - IEEE International Conference on Machine Learning and Applications, 2022)

– Cabezudo, M.A.S., Inácio, M., **Rodrigues, A.C.**, Casanova, E. and Sousa, R.F.D., March. Natural language inference for portuguese using bert and multilingual information. In International Conference on Computational Processing of the Portuguese Language (pp. 346-356). Springer, Cham, 2020.

7.4 Limitações e trabalhos futuros

Como trabalho futuro, especialmente na combinação de *embeddings* observamos alguns pontos que podem ser explorados ou aprofundados. No experimento de combinação de modelos de *embeddings*, há espaço para explorar outros modelos. Devido à limitações com relação ao tempo, optamos por um aprofundamento na avaliação em detrimento de mais experimentos, focando em uma única arquitetura e com um número limitado de combinações de modelos. Como trabalho futuro prevemos a exploração de novas arquiteturas com mais combinações de modelos pré-treinados.

Outro ponto a ser explorado é um aprofundamento na análise das predições. Realizamos um primeiro passo na avaliação de instâncias do *dataset* sobre um ponto de vista da informação no texto utilizando o LIME estendido, o que acreditamos pode ser enriquecido com seu uso para mais instâncias, inclusive incorporando *metadados* para apoiar a explicação.

Um outro caminho a ser seguido está nas relações entre atributos linguísticos e predições. Uma das abordagens na avaliação de modelos durante o trabalho foi a busca por correlações entre atributos textuais e as predições dos modelos, a qual fizemos para alguns atributos anotados. Os resultados iniciais foram inconclusivos. Contudo, acreditamos que esta abordagem pode ser retomada e aprofundada considerando-se mais atributos do texto e com a inclusão de *datasets* de outros domínios.

Deixamos também como sugestão de investigação futura o estudo de formas de comparar características do texto entre *datasets* de diferentes domínios para determinar quais características são relevantes para generalização de resultados na tarefa de STS.

REFERÊNCIAS

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; KUDLUR, M.; LEVENBERG, J.; MONGA, R.; MOORE, S.; MURRAY, D. G.; STEINER, B.; TUCKER, P.; VASUDEVAN, V.; WARDEN, P.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: A system for large-scale machine learning. In: **12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)**. Berkeley: USENIX Association, 2016. Citado na página 21.

AGGARWAL, C. C. **Machine learning for text**. Nova York: Springer, 2018. v. 848. Citado na página 19.

ALVES, A. O.; RODRIGUES, R.; OLIVEIRA, H. G. Asapp: alinhamento semântico automático de palavras aplicado ao português. **Linguamática**, v. 8, n. 2, p. 43–58, 2016. Citado na página 74.

ANTONIAK, M.; MIMNO, D. Evaluating the stability of embedding-based word similarities. **Transactions of the Association for Computational Linguistics**, MIT Press, v. 6, p. 107–119, 2018. Citado nas páginas 20 e 46.

BAKAROV, A. A survey of word embeddings evaluation methods. **arXiv preprint arXiv:1801.09536**, 2018. Citado na página 46.

BARBOSA, L.; CAVALIN, P.; GUIMARAES, V.; KORMAKSSON, M. Blue man group no assin: Usando representações distribuídas para similaridade semântica e inferência textual. **Linguamática**, v. 8, n. 2, p. 15–22, 2016. Citado na página 74.

BARONI, M.; DINU, G.; KRUSZEWSKI, G. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics**. Baltimore, Maryland: Association for Computational Linguistics, 2014. p. 238–247. Citado na página 45.

BENDER, E. M.; KOLLER, A. Climbing towards NLU: On meaning, form, and understanding in the age of data. In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, 2020. p. 5185–5198. Citado na página 21.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JAUVIN, C. A neural probabilistic language model. **Journal of machine learning research**, v. 3, n. Feb, p. 1137–1155, 2003. Citado na página 44.

BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, Curran Associates, Inc, Red Hook, v. 5, p. 135–146, 2017. ISSN 2307-387X. Citado nas páginas 47 e 48.

CANÇADO, M. **Manual de semântica**. São Paulo: Contexto, 2012. Citado na página 35.

CANÇADO, M.; AMARAL, L. **Introdução à Semântica Lexical: papéis temáticos, aspecto lexical e decomposição de predicados**. Petrópolis: Editora Vozes Limitada, 2017. Citado na página 35.

CER, D.; DIAB, M.; AGIRRE, E.; LOPEZ-GAZPIO, I.; SPECIA, L. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: **Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)**. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 1–14. Citado na página 25.

CHIERCHIA, G. **Semântica, Tradução de Luis Arthur Pagani**. Campinas: Unicamp eduel, 2003. Citado na página 34.

CHOI, H.; KIM, J.; JOE, S.; GWON, Y. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In: **2020 25th International Conference on Pattern Recognition (ICPR)**. Milano: IEEE, 2021. p. 5482–5487. Citado na página 20.

CHOMSKY, N. **Estruturas Sintáticas**. Petrópolis: Editora Vozes, 1957. Citado nas páginas 38 e 39.

____. **Linguagem e mente**. São Paulo: Editora Unesp, 2006. Citado na página 33.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. 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**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Citado nas páginas 20, 21, 47, 49, 51, 56, 68 e 85.

ELAVARASI, S. A.; AKILANDESWARI, J.; MENAGA, K. Ontology based semantic similarity measure using concept weighting. In: CITESEER. **International Journal of Computer Applications (IJCA)**. Coimbatore, India, 2014. Citado nas páginas 20 e 31.

FARUQUI, M.; TSVETKOV, Y.; RASTOGI, P.; DYER, C. Problems with evaluation of word embeddings using word similarity tasks. In: **Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP**. Berlin, Germany: Association for Computational Linguistics, 2016. p. 30–35. Citado na página 20.

FIALHO, P.; COHEUR, L.; QUARESMA, P. Benchmarking natural language inference and semantic textual similarity for portuguese. **Information**, Multidisciplinary Digital Publishing Institute, v. 11, n. 10, p. 484, 2020. Citado nas páginas 20 e 74.

FIRTH, J. R. **A Synopsis of Linguistic Theory, 1930-1955**. Oxford: Oxford, 1957. 4-32 p. Citado na página 21.

FONSECA, E.; SANTOS, L.; CRISCUOLO, M.; ALUISIO, S. Assin: Avaliacao de similaridade semantica e inferencia textual. In: **Computational Processing of the Portuguese Language-12th International Conference**. Tomar: Springer, 2016. p. 13–15. Citado nas páginas 28 e 74.

FONSECA, E. R. **Reconhecimento de implicaçao textual em português**. Tese (Doutorado) — Universidade de São Paulo, 2018. Citado na página 28.

FREGE, G. Sense and reference. **The philosophical review**, JSTOR, v. 57, n. 3, p. 209–230, 1948. Citado na página 35.

FREIRE, J.; PINHEIRO, V.; FEITOSA, D. Flexsts: Um framework para similaridade semântica textual. **Linguamática**, v. 8, n. 2, p. 23–31, 2016. Citado na página 74.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: **Proceedings of the 22nd International conference on Knowledge discovery and data mining (SIGKDD)**. New York: Association for Computing Machinery, 2016. p. 855–864. Citado nas páginas 53 e 54.

HARRIS, Z. S. Distributional structure. **Word**, Taylor & Francis, v. 10, n. 2-3, p. 146–162, 1954. Citado na página 27.

HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; RODRIGUES, J.; ALUISIO, S. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **arXiv preprint arXiv:1708.06025**, 2017. Citado na página 28.

HARTMANN, N. S. Solo queue at assin: Combinando abordagens tradicionais e emergentes. **Linguamática**, v. 8, n. 2, p. 59–64, 2016. Citado na página 74.

JOHNSON, J.; DOUZE, M.; JÉGOU, H. Billion-scale similarity search with GPUs. **IEEE Transactions on Big Data**, IEEE, v. 7, n. 3, p. 535–547, 2019. Citado na página 21.

JURAFSKY, D.; MARTIN, J.; NORVIG, P.; RUSSELL, S. **Speech and Language Processing**. Upper Saddle River: Pearson, 2014. ISBN 9780133252934. Citado na página 39.

KEMMERER, D. **Cognitive neuroscience of language**. Londres: Psychology Press, 2014. Citado na página 33.

KIELA, D.; HILL, F.; CLARK, S. Specializing word embeddings for similarity or relatedness. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 2044–2048. Citado na página 20.

KIELA, D.; WANG, C.; CHO, K. Dynamic meta-embeddings for improved sentence representations. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 1466–1477. Citado na página 67.

LAURIOLA, I.; LAVELLI, A.; AIOLLI, F. An introduction to deep learning in natural language processing: Models, techniques, and tools. **Neurocomputing**, Elsevier, v. 470, p. 443–456, 2022. Citado na página 19.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: **International conference on machine learning**. Massachusetts: JMLR, 2014. p. 1188–1196. Citado nas páginas 47, 49 e 56.

LEVY, O.; GOLDBERG, Y. Dependency-based word embeddings. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Baltimore, Maryland: Association for Computational Linguistics, 2014. p. 302–308. Citado na página 49.

LING, W.; DYER, C.; BLACK, A. W.; TRANCOSO, I. Two/too simple adaptations of word2vec for syntax problems. In: **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Denver,

Colorado: Association for Computational Linguistics, 2015. p. 1299–1304. Citado na página 49.

LIU, Q.; LU, J.; ZHANG, G.; SHEN, T.; ZHANG, Z.; HUANG, H. Domain-specific meta-embedding with latent semantic structures. **Information Sciences**, Elsevier, v. 555, p. 410–423, 2021. Citado na página 67.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019. Citado na página 21.

MARCUSCHI, L. A. *et al.* Gêneros textuais: definição e funcionalidade. **Gêneros textuais e ensino. Rio de Janeiro: Lucerna**, v. 20, 2002. Citado na página 40.

MARELLI, M.; MENINI, S.; BARONI, M.; BENTIVOGLI, L.; BERNARDI, R.; ZAMPARELLI, R. A sick cure for the evaluation of compositional distributional semantic models. In: **Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)**. Reykjavik, Iceland: European Language Resources Association (ELRA), 2014. p. 216–223. Citado na página 29.

MARNEFFE, M.-C. de; MACCARTNEY, B.; MANNING, C. D. Generating typed dependency parses from phrase structure parses. In: **Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)**. Genoa, Italy: European Language Resources Association (ELRA), 2006. Citado na página 40.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013. Citado nas páginas 19, 43, 45, 46, 48 e 56.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. Lake Tahoe: Curran Associates Inc., 2013. p. 3111–3119. Citado na página 53.

NEELAKANTAN, A.; SHANKAR, J.; PASSOS, A.; MCCALLUM, A. Efficient non-parametric estimation of multiple embeddings per word in vector space. In: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1059–1069. Citado na página 29.

NETO, J. B. Semântica de modelos. **MÜLLER, AL, NEGRÃO, EV, FOLTRAN, MJ Semântica formal. São Paulo: Contexto**, 2003. Citado na página 34.

NIVRE, J.; MARNEFFE, M.-C. D.; GINTER, F.; GOLDBERG, Y.; HAJIC, J.; MANNING, C. D.; MCDONALD, R.; PETROV, S.; PYYSALO, S.; SILVEIRA, N. *et al.* Universal dependencies v1: A multilingual treebank collection. In: **Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)**. Portorož, Slovenia: European Language Resources Association (ELRA), 2016. p. 1659–1666. Citado na página 40.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L. *et al.* Pytorch: An imperative style, high-performance deep learning library. **Advances in neural information processing systems**, v. 32, 2019. Citado na página 21.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543. Citado nas páginas 19, 43, 45 e 48.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. In: **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 2227–2237. Citado na página 47.

PETROV, S.; DAS, D.; MCDONALD, R. A universal part-of-speech tagset. In: **Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)**. Istanbul, Turkey: European Language Resources Association (ELRA), 2012. p. 2089–2096. Citado na página 40.

PINHEIRO, A.; FERREIRA, R.; FERREIRA, M. A. D.; ROLIM, V. B.; TENÓRIO, J. V. S. Statistical and semantic features to measure sentence similarity in portuguese. In: IEEE. **2017 Brazilian Conference on Intelligent Systems (BRACIS)**. Uberlândia, 2017. p. 342–347. Citado na página 74.

POERNER, N.; WALTINGER, U.; SCHÜTZE, H. Sentence meta-embeddings for unsupervised semantic textual similarity. **arXiv preprint arXiv:1911.03700**, 2019. Citado na página 71.

QIU, X.; SUN, T.; XU, Y.; SHAO, Y.; DAI, N.; HUANG, X. Pre-trained models for natural language processing: A survey. **Science China Technological Sciences**, Springer, v. 63, n. 10, p. 1872–1897, 2020. Citado nas páginas 21, 46 e 47.

RADEMAKER, A.; CHALUB, F.; REAL, L.; FREITAS, C.; BICK, E.; PAIVA, V. de. Universal dependencies for portuguese. In: **Proceedings of the Fourth International Conference on Dependency Linguistics (Depling)**. Pisa,Italy: Linköping University Electronic Press, 2017. p. 197–206. Citado nas páginas 62 e 63.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. Improving language understanding with unsupervised learning. **Technical report, OpenAI**, 2018. Citado nas páginas 20, 21 e 47.

REAL, L.; FONSECA, E.; OLIVEIRA, H. G. The assin 2 shared task: a quick overview. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. Évora, 2020. p. 406–412. Citado nas páginas 29 e 74.

REAL, L.; RODRIGUES, A.; SILVA, A. V. e; ALBIERO, B.; THALENBERG, B.; GUIDE, B.; SILVA, C.; LIMA, G. de O.; CÂMARA, I. C.; STANOJEVIĆ, M. *et al.* Sick-br: a portuguese corpus for inference. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. Canela, 2018. p. 303–312. Citado nas páginas 26 e 29.

REIMERS, N.; BEYER, P.; GUREVYCH, I. Task-oriented intrinsic evaluation of semantic textual similarity. In: **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers**. Osaka, Japan: The COLING 2016 Organizing Committee, 2016. p. 87–96. Citado na página 73.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 3982–3992. Citado na página 69.

____. Making monolingual sentence embeddings multilingual using knowledge distillation. In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Online: Association for Computational Linguistics, 2020. p. 4512–4525. Citado na página 70.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should I trust you?": Explaining the predictions of any classifier. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016**. Nova York: Association for Computing Machinery, 2016. p. 1135–1144. Citado nas páginas 24 e 71.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019. Citado na página 21.

SAUSSURE, F. D. **Curso de linguística geral**. São Paulo: Editora Cultrix, 2008. Citado na página 34.

SCHNABEL, T.; LABUTOV, I.; MIMNO, D.; JOACHIMS, T. Evaluation methods for unsupervised word embeddings. In: **Proceedings of the 2015 conference on empirical methods in natural language processing**. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 298–307. Citado nas páginas 20, 46 e 47.

SEARLE, J. R. Minds, brains, and programs. **Behavioral and brain sciences**, Cambridge University Press, v. 3, n. 3, p. 417–424, 1980. Citado na página 21.

SILVA, J. R. d. *et al.* Geração de vetores de sentido para o português. Universidade Federal de São Carlos, 2019. Citado na página 29.

SILVA, J. Rodrigues da; CASELI, H. d. M. Sense representations for portuguese: experiments with sense embeddings and deep neural language models. **Language Resources and Evaluation**, Springer, v. 55, n. 4, p. 901–924, 2021. Citado na página 74.

SINOARA, R. A.; ROSSI, R. G.; REZENDE, S. O. Semantic role-based representations in text classification. In: IEEE. **2016 23rd International Conference on Pattern Recognition (ICPR)**. Cancun: Institute of Electrical and Electronics Engineers (IEEE), 2016. p. 2313–2318. Citado na página 20.

SOUZA, J. V. A. de; OLIVEIRA, L. E. S. e; GUMIEL, Y. B.; CARVALHO, D. R.; MORO, C. M. C. Incorporating multiple feature groups to a siamese neural network for semantic textual similarity task in portuguese texts. In: **ASSIN@ STIL**. Salvador: SBC, 2019. p. 59–68. Citado nas páginas 21 e 74.

THALENBERG, B. **Distinguishing antonyms from synonyms in vector space models of semantics**. Tomar, 2016. Citado na página 62.

TRASK, A.; MICHALAK, P.; LIU, J. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. **arXiv preprint arXiv:1511.06388**, 2015. Citado nas páginas 29 e 49.

TURNEY, P. D.; PANTEL, P. From frequency to meaning: Vector space models of semantics. **Journal of artificial intelligence research**, v. 37, p. 141–188, 2010. Citado nas páginas 19, 20, 25, 26, 43, 44 e 46.

TVERSKY, A. Features of similarity. **Psychological review**, American Psychological Association, v. 84, n. 4, p. 327, 1977. Citado na página 35.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: **Advances in neural information processing systems**. Red Hook: Curran Associates Inc., 2017. p. 5998–6008. Citado nas páginas 21, 49 e 50.

WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O.; BOWMAN, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: **Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 353–355. Citado na página 51.

WANG, K.; REIMERS, N.; GUREVYCH, I. TSDAE: Using transformer-based sequential denoising auto-encoderfor unsupervised sentence embedding learning. In: **Findings of the Association for Computational Linguistics: EMNLP 2021**. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. p. 671–688. Citado na página 20.

XU, H.; LIU, B.; SHU, L.; YU, P. S. Lifelong domain word embedding via meta-learning. **arXiv preprint arXiv:1805.09991**, 2018. Citado na página 67.

CAPÍTULO

8

APÊNDICE

Durante o trabalho realizamos experimentos adicionais com o intuito de avaliar se haveria correlação entre atributos do texto e a predições de similaridade do modelos de meta-embedding que propusemos. A ideia era observar se as predições poderiam ser explicadas por um conjunto de atributos interpretáveis observados diretamente no texto. Para tanto, realizamos a anotação de uma parte do *dataset* ASSIN1, considerando características sintáticas, entidade nomeada e tópico abordado nas sentenças. Nossos resultados preliminares indicaram que os atributos com os quais trabalhamos não estão linearmente correlacionados com as predições, e ainda, para cada faixa de valor de similaridade há diferentes pesos na correlação de atributos. Deixamos aqui uma descrição resumida da anotação realizada como informação adicional.

8.1 Anotação de atributos

Com o intuito de analisar a correlação entre as predições de modelos e características do texto, realizamos uma anotação extra da partição de teste do *dataset* ASSIN1. Cada instância do *dataset* é um par de sentenças, portanto, foi necessário caracterizar as sentenças individualmente e posteriormente criar atributos a partir da comparação das sentenças no par. Um exemplo está no comprimento das sentenças. Dado um par composto por duas sentenças, s_1 e s_2, inicialmente foram contados os números de *tokens* de cada uma, caracterizando-as individualmente. Para obter um atributo do par, foi calculado a proporção da diferença entre a contagem de s_1 e s_2 e média das duas. Segue abaixo a descrição dos atributos associados a cada sentença e aqueles que foram calculados para o par.

Atributos das sentenças

1. **comprimento ℓ:** Quantidade de *tokens* da sentença.

2. **entidade nomeada:** Entidades nomeadas presentes nas sentenças, anotadas manualmente. Entre elas constam nomes de esportistas, políticos e instituições.

3. **tópico:** Assunto abordado na sentença. A anotação foi manual para as sentença individualmente, considerando-se apenas a informação contida em cada uma. As sentenças do ASSIN1 são extraídas de mídias jornalísticas, portanto os tópico ficam como subdivisões do gênero jornalístico. Os tópicos identificados foram: esporte, entretenimento, produtos de tecnologia, economia e mercado, política, saúde, notícias mundiais (notícias de países do mundo exceto o Brasil e Portugal), ciência e outras notícias. A cada sentença foi atribuído apenas um tópico.

4. **dependência sintática:** estrutura de dependência sintática de acordo com o modelo de *Universal Dependency* anotada de forma automatizada pelo parser sintático do SciPy.

Atributos dos pares

1. **proporção do comprimento:** Relação do comprimento entre as duas sentenças. Tomamos a diferença de comprimento dividido pelo comprimento médio das sentenças.

$$L = \frac{|\ell_1 - \ell_2|}{(\ell_1 + \ell_2) \cdot 0,5}$$

2. **proporção de entidade nomeada em ambas sentenças:** Proporção de entidades nomeadas presentes nas duas sentenças em relação a soma de entidades do par.

3. **relação entre tópicos:** Valor binário no qual foi atribuído 0 se o tópico de cada sentença do par é distinto e 1 caso o tópico seja o mesmo.

4. **semelhança de dependência sintática:** Mesma estrutura sintática do primeiro nível de dependência a partir da raiz. São observados quantos nós filhos a raiz possuí nas sentenças e feita uma proporção entre aqueles que são iguais em ambas (possuem o mesmo rótulo sintático) e todos do par. Na prática isso significa que se a raiz tiver os mesmos nós filhos sintáticos, o valor será 1. Caso as raízes das sentenças tenham filhos distintos, o valor é menor, chegando a 0 para nenhuma intersecção de filhos sintáticos.


ICMC
USP
SÃO CARLOS