Detecção de Hiperônimos com BERT e Padrões de Hearst

Gabriel Escobar Paes

Detecção de Hiperônimos com BERT e Padrões de Hearst

Gabriel Escobar Paes

Orientador: *Prof. Dr. Eraldo Luís Rezende Fernandes*

Dissertação entregue à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul - FACOM-UFMS como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UFMS - Campo Grande
Abril/2021

Abstract

Hypernym relation (also known as is-a relation) is a relevant semantic relation between words that is useful to tasks like coreference resolution, relation extraction, textual entailment, among others. A hypernym is a generic word, while a hyponym is a specific word. For example, city is a hypernym of rome, and dog is a hyponym of animal. In this work, we propose an unsupervised algorithm for hypernym detection that combines Hearst patterns with the BERT language model. Hearst patterns are linguistic patterns such as banana is a kind of fruit, which indicates that fruit is hypernym of banana. An important limitation of such methods is its sparsity, a common problem for pattern-based methods. The BERT language model is a contextual representation model trained to predict masked words within an input sequence. We combine this aspect of BERT with Hearst patterns to create a novel algorithm for hypernym detection which achieves the state-of-the-art performance on 7 out of 13 evaluated datasets. Among these datasets, there are three new datasets in Portuguese, which were developed during this work and are the first for this language. We compare our method to the DIVE algorithm, an extension of the well-known word2vec algorithm. DIVE retained the best results for most of the datasets in English. Our method outperforms DIVE by 3 points on average for the thirteen considered datasets.

Keywords: Hypernym Detection, BERT, Hearst Patterns, Deep Language Model.

Resumo

A relação de hiperônimo é uma importante relação semântica entre palavras que é útil para resolver problemas como resolução de correferência, extração de relações, *textual entailment*, dentre outros. Um hiperônimo é uma palavra de sentido mais genérico, enquanto um hipônimo é uma palavra de sentido mais específico. Por exemplo, cidade é hiperônimo de roma, e cachorro é hipônimo de animal. Neste trabalho, propomos um algoritmo não supervisionado para a tarefa de detecção de hiperônimo que combina os chamados padrões de Hearst com o modelo de linguagem BERT. Padrões de Hearst são padrões linguísticos como banana é um tipo de fruta, o qual é um indício que fruta é um hiperônimo de banana. Uma limitação deste tipo de abordagem é o problema de escassez (*sparsity*), comum a métodos baseados em padrões linguísticos. O modelo de linguagem BERT é um modelo profundo de representação contextual que é treinado para predizer palavras mascaradas na sequência de entrada. Nós combinamos esta característica do BERT com padrões de Hearst para derivar um algoritmo de detecção de hiperônimo que obtém os melhores resultados da literatura em 7 de 13 datasets considerados. Dentre estes datasets, estão os três primeiros datasets em português e que foram desenvolvidos neste trabalho. Nós comparamos nosso método com o algoritmo DIVE, uma extensão do conhecido algoritmo word2vec que detinha os melhores resultados na maioria dos datasets em inglês para detecção de hiperônimo. Nosso método alcança um desempenho 3 pontos acima do DIVE na média dos treze datasets considerados.

Palavras-chave: Detecção de Hiperônimo, BERT, Padrões de Hearst, Modelo Profundo de Linguagem.

Sumário

Sumário iv
Lista de Figuras vi
Lista de Tabelas viii

1 Introdução **1**
1.1 Objetivos 5
1.2 Estrutura do Documento 6

2 Trabalhos Relacionados **7**
2.1 Detecção Supervisionada de Hiperônimos 7
2.2 Descoberta de Hiperônimos 8

3 Referencial Teórico **11**
3.1 Detecção de Hiperônimos 11
3.1.1 Métrica de Avaliação 12
3.2 Padrões de Hearst 15
3.3 BERT 16
3.4 BERT pode Falar 21
3.5 Bases de Semântica Lexical 24
3.6 DIVE 24
3.7 Datasets em Inglês 28

4 BHearst **32**
4.1 BERT e Padrões de Hearst 32
4.1.1 Constante de Normalização 33
4.1.2 Combinando Múltiplos Padrões 37
4.2 Datasets em Português 38
4.3 DIVE em PyTorch e Word2vec 42
4.3.1 Corpus para Treinamento Não Supervisionado 44

5 Resultados Experimentais **46**
5.1 Ambiente Experimental 46
5.2 Análise dos Datasets DEV 48
5.3 Combinando Múltiplos Padrões 49
5.4 Comparação com Estado da Arte 52
5.5 Constante de Normalização 54

6 Conclusão e Trabalhos Futuros **56**
6.1 Conclusão 56
6.2 Trabalhos Futuros 57

Referências **64**

A Resultados Adicionais **65**
A.1 Estatísticas dos Datasets de Teste 65

Lista de Figuras

1.1 Representação em árvore de um hiperônimo com seus hipônimos. 1

2.1 Representação em árvore dos synsets 10

3.1 Exemplo de dataset para detecção de hiperônimo. 13
3.2 Exemplo de três pares preditos como positivos por um modelo fictício. Como apenas um deles é realmente positivo, a precisão é igual a $1/3$. 13
3.3 Lista de predição contendo os pares do dataset da Figura 3.1 ordenados de acordo com a pontuação dada por um modelo fictício. 14
3.4 Valores de precisão@k para os valores k tal que o k-ésimo par, na lista de predição, é positivo. 15
3.5 Ilustração da representação da entrada no BERT. 18
3.6 Representação das camadas do BERT. 19
3.7 Camada de classificação do MLM para um token mascarado. . . . 20

4.1 Lista de predição para cada padrão. 37
4.2 Exemplos de um par de synsets com uma relação de hiperônimo. 40
4.3 Exemplo do treinamento do word2vec para uma sentença. 43
4.4 Exemplo de cálculo da pontuação $P^{\text{DIVE}}(\text{tigre}, \text{animal})$. 44

5.1 Estatísticas de pares no dataset DEV em inglês: (a) distribuição de pares por comprimento; (b) porcentagem de pares positivos por comprimento. 49
5.2 Estatísticas de pares no dataset DEV em português: (a) distribuição de pares por comprimento; (b) porcentagem de pares positivos por comprimento. 49
5.3 Desempenho de cada padrão no dataset DEV em inglês. 50
5.4 Desempenho de cada padrão no dataset DEV em português. . . . 50

A.1 No eixo X temos os comprimentos de par presentes em cada dataset em inglês e no eixo Y temos o número de pares para cada comprimento. 66
A.2 Quantidade de pares para cada comprimento nos datasets de teste em português. 67
A.3 Porcentagem de pares positivos para cada comprimento de par nos datasets em inglês. 68
A.4 Porcentagem de pares positivos para cada comprimento, usando todos os pares dos datasets. 69

Lista de Tabelas

1.1 Lista de pares candidatos de hipônimo e hiperônimo. Cada par pode ser positivo ou negativo. 2

2.1 Exemplo de dataset para classificação binária para a relação de hiperônimo entre pares de palavras (x, y), onde x é o hipônimo e y é o hiperônimo. 7
2.2 Exemplo de um par de instâncias. 8
2.3 Exemplo de deslocamentos entre pares de vetores. 8
2.4 Hipônimos e lista de hiperônimos correspondentes. 9
2.5 Synsets e seus termos. 10

3.1 Padrões usados em inglês. 16
3.2 Padrões usados em português. 17
3.3 Exemplo de uma frase na qual cada palavra é mascarada para gerar uma sequência de entrada para o BERT. $X_{\setminus t}$ representa uma sequência X com seu t-ésimo token mascarado. 22
3.4 Exemplos de contextos de poodle e seu hiperônimo cachorro. . . 25
3.5 Número de pares em cada em dataset antes e depois do pré-processamento. 29
3.6 Distribuição entre pares positivos e negativos em cada dataset em inglês. 31

4.1 Exemplo de dataset contendo oito pares com suas respectivas classes. 33
4.2 Exemplo de sequências mascaradas usando o padrão x é um tipo de y para o par (tigre, anserimal). A palavra tigre é tokenizada em dois sub-tokens (ti e -gre); enquanto a palavra animal em três sub-tokens (a, -ni e -mal). 34
4.3 Exemplos de pares e seus comprimentos. 35
4.4 Combinando padrões usando o ranking de cada par. 38

4.5 Relações presentes no ConceptNet e consideradas para este trabalho. 39
4.6 Relações presentes no ConceptNet e não consideradas para este trabalho. 40
4.7 Número de pares após o pré-processamento. 40
4.8 Balanceamento e total de pares. 41
4.9 Balanceamento e total de pares. 41
4.10 Pares de palavras formadas usando o par de synsets da Figura 4.2. 41
4.11 Número de pares positivos e negativos dos datasets de validação e teste gerados a partir da base OntoPT. 42
4.12 Dados do corpus criado. 45

5.1 Informações de hardware do servidor. 46
5.2 Hiperparâmetros utilizados no treinamento do DIVE para o idioma português. 48
5.3 Hiperparâmetros utilizados no treinamento do word2vec para o idioma português. 48
5.4 Desempenho usando as estratégias Average Rank (AR) e Min Rank (MR) para combinar até os quatro melhores padrões nos datasets de teste em inglês. Os valores k em AR(k) e MR(k) indicam quantos padrões são combinados, sempre em ordem de desempenho no DEV. 51
5.5 Desempenho usando as estratégias Average Rank (AR) e Min Rank (MR) para combinar até os quatro melhores padrões nos datasets de teste em português. Os valores k em AR(k) e MR(k) indicam quantos padrões são combinados, sempre em ordem de desempenho no DEV. 51
5.6 Desempenho do método BHearst comparado aos métodos DIVE e word2vec (W2V) nos datasets de teste em inglês. 53
5.7 Desempenho do método BHearst comparado aos métodos DIVE e word2vec (W2V) nos datasets de teste em português. 53
5.8 Impacto do balanceamento de pares por comprimento para o cálculo da constante de normalização $\log Z_L$ (coluna *Desbalanceado*) e o impacto da própria constante como um todo (coluna *Sem* $\log Z_L$) no desempenho do método BHearst nos datasets em inglês. 55
5.9 Impacto da constante de normalização $\log Z_L$ no desempenho do método BHearst nos datasets em português. 55

CAPÍTULO

1

Introdução

Nas últimas duas décadas, a comunidade de Processamento de Linguagem Natural (PLN) tem investido no desenvolvimento de métodos automáticos para reconhecimento da relação de hiperônimo entre palavras (Shwartz et al., 2017). *Hiperônimia*, *Relação Hipônimo-Hiperônimo*, ou simplesmente *Relação de Hiperônimo* é uma relação semântica entre palavras que representa uma ideia de generalização, sendo uma das principais relações semânticas em diversas ontologias linguísticas (Yu et al., 2015). Mais especificamente, hiperônimo é uma palavra de sentido mais genérico em relação à outra; por exemplo, animal é hiperônimo de tigre. Enquanto a definição de hipônimo corresponde à relação inversa, ou seja, uma palavra de sentido mais específico em relação à outra; por exemplo, tigre é hipônimo de animal. Outros exemplos desta relação são ilustrados na Figura 1.1. A relação de hiperônimo é relevante para

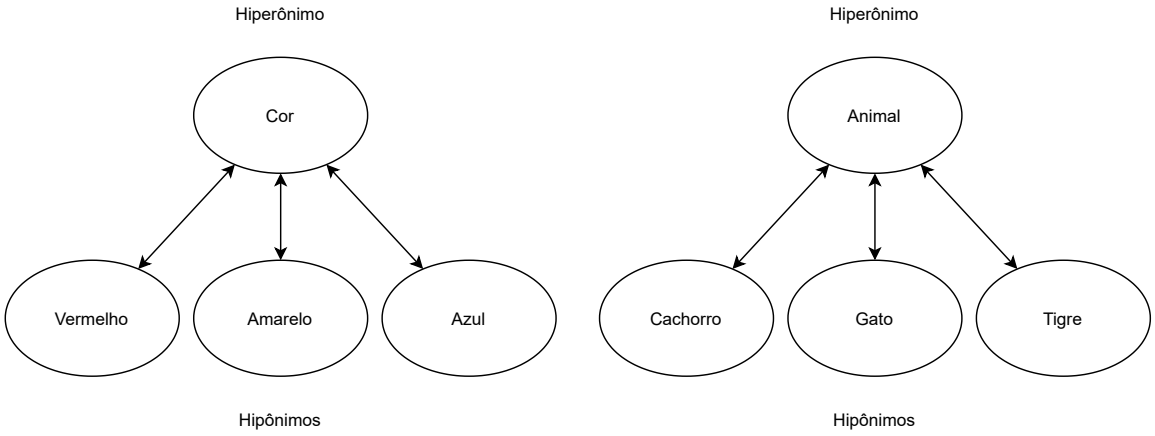


Figura 1.1: Representação em árvore de um hiperônimo com seus hipônimos.

muitas tarefas na área de PLN como, por exemplo, *textual entailment* (Sam-

mons et al., 2011), resolução de correferência (Ponzetto and Strube, 2006), extração de relação (Demeester et al., 2016) e sistemas de perguntas e respostas (Huang et al., 2008).

Na literatura, existem diferentes tarefas relacionadas à relação de hiperônimo. As duas tarefas mais populares são denominadas descoberta de hiperônimo e detecção de hiperônimo. Na descoberta de hiperônimo, o objetivo é descobrir o hiperônimo de uma palavra específica dado um vasto conjunto de documentos. Por outro lado, a tarefa de *detecção de hiperônimo* consiste em um problema de classificação binária que pode ser abordado de maneira supervisionada ou não-supervisionada.

Neste trabalho, é considerado a tarefa de detecção de hiperônimo por meio de métodos não-supervisionados. Uma entrada desta tarefa é composta por um par de palavras e a saída consiste em predizer se o par possui uma relação de hiperônimo ou não (positivo ou negativo). Na Tabela 1.1, são ilustrados alguns exemplos juntamente com suas classes para esta tarefa.

Hipônimo	**Hiperônimo**	**Classe**
amigo	discurso	Negativo
urânio	elemento	Postivo
profissão	osso	Negativo
vaga	espaço	Positivo

Tabela 1.1: Lista de pares candidatos de hipônimo e hiperônimo. Cada par pode ser positivo ou negativo.

A tarefa de detecção de hiperônimo pode ser resolvida através de abordagens supervisionadas ou não-supervisionadas. As abordagens supervisionadas tendem a superar as não-supervisionadas na maioria dos datasets. Entretanto, Levy et al. (2015b) mostraram que os melhores métodos supervisionados sofrem de um grave problema denominado *memorização léxica* (*lexical memorization*). Este problema ocorre quando, durante o treinamento supervisionado, são utilizados dados anotados em que uma palavra é muitas vezes categorizadas como positiva. Com isso, a relação semântica dessa palavra não é aprendida e apenas um padrão é gravado. Eles mostraram que este problema está presente em vários datasets utilizados em trabalhos anteriores e que o bom desempenho alcançado pelos métodos supervisionados são altamente relacionados a este tipo de viés nos dados. Portanto, neste trabalho, é considerado abordagens não-supervisionadas para detecção de hiperônimos.

Dentre as abordagens não-supervisionadas para detecção de hiperônimo, destacam-se dois grupos de métodos: um baseado em *padrões de linguagem* e outro em *vetores de palavras* (*word embeddings*). Os métodos baseados em padrões de linguagem (Hearst, 1992) fazem uso de padrões como, por

exemplo, x ou outro y. Por exemplo, o par (x=macaco, y=animal) se encaixa neste padrão, pois o padrão macaco ou outro animal é facilmente encontrado em textos. Um par de palavras (x, y) que frequentemente ocorre neste padrão indica que, provavelmente, a palavra y é um hiperônimo da palavra x. Um problema dessa abordagem baseada em padrões é que ela falha quando as palavras não ocorrem exatamente como é ditado pelo padrão. Por exemplo, o padrão anterior x ou outro y não se encaixa na sentença macaco ou algum outro animal, mesmo que esse padrão e essa sentença sejam parecidos.

Já as abordagens baseadas em vetores de palavras (Chang et al., 2018) utilizam modelos preditivos de distribuição de palavras e seus contextos para o aprendizado não supervisionado de vetores $\mathbf{v}_w \in \mathbb{R}^D$ para cada palavra w, onde D é a dimensão do espaço vetorial considerado. Estes métodos predizem a relação de hiperônimo entre um par de palavras (x, y) em função de $\mathbf{v}_x$ e $\mathbf{v}_y$. O método DIVE (*Distributional Inclusion Vector Embedding*), por exemplo, foi proposto por Chang et al. (2018) para a tarefa de detecção não-supervisionada de hiperônimo. Este método estende o popular modelo *word2vec* (Mikolov et al., 2013) para aprender vetores de palavras específicos para a tarefa de detecção de hiperônimos a partir de um grande corpus. Os vetores de palavras são aprendidos de tal maneira que representem a relação de hiperônimo entre pares de palavras por meio da *hipótese de inclusão distributiva* (*distributional inclusion hypothesis* ou, simplesmente, DIH). Resumidamente, a DIH dita que dado um par de palavras (x, y) e um grande corpus, os contextos onde o hipônimo x aparece são um subconjunto dos contextos onde o hiperônimo y é encontrado, pois um hiperônimo é uma generalização dos seus hipônimos. Uma das principais vantagens dos métodos baseados em vetores de palavras é não sofrer do problema de escassez que assola os métodos baseados em padrões.

Apesar da vantagem citada acima, Roller et al. (2018) apresentam um estudo indicando que métodos baseados em padrões de Hearst (padrões linguísticos que frequentemente indicam a presença da relação de hiperônimo entre duas palavras) podem ser competitivos e até superiores aos métodos baseados em vetores de palavras. Desta forma, neste trabalho, é proposto um novo método não-supervisionado para detecção de hiperônimos denominado *BHearst* que combina o modelo *contextual* de linguagem BERT (Devlin et al., 2018) e padrões de Hearst. É importante salientar que o método DIVE, e outros métodos de vetores de palavras, não são considerados contextuais. Apesar destes vetores serem aprendidos a partir dos contextos onde uma palavra ocorre, no momento de utilizá-los em um contexto específico (uma frase, por exemplo), o vetor de uma palavra não depende deste contexto. Já o modelo BERT é uma representação contextual, no sentido de que o vetor de uma palavra é

calculado com base no contexto onde ela está inserida. Esta característica é essencial na proposta deste trabalho.

Bidirectional Encoder Representations from Transformers (BERT) é um modelo de representação contextual de linguagem (Devlin et al., 2018) que tem alcançado resultados estado-da-arte para diversas tarefas de PLN. Este modelo é treinado de maneira não-supervisionada utilizando um grande corpus. Este treinamento é baseado em uma abordagem que *mascara* (*masked language model* ou, simplesmente, MLM) algumas palavras em uma sequência de palavras do corpus. Isto é, o modelo tem acesso a todas as palavras da sequência de entrada exceto às palavras mascaradas. Desta forma, o treinamento consiste em predizer quais são as palavras mascaradas na sequência original. Para isto, a representação de cada palavra é calculada por um mecanismo de atenção que pondera a representação de todas as outras palavras da sequência. O MLM é similar ao popular exercício em aulas de línguas conhecido como "preencha as lacunas", no qual o estudante precisa preencher palavras que foram retiradas de uma frase. Este modelo tem uma relação interessante com os padrões de Hearst. Por exemplo, na frase morango é um tipo de y, podemos mascarar a palavra y e o modelo BERT calculará a probabilidade de qualquer palavra aparecer na posição de y. Desta forma, é possível comparar a probabilidade de diferentes palavras e concluir, por exemplo, que as palavras fruta e alimento são mais prováveis do que veículo, por exemplo. A ideia básica do método proposto neste trabalho é utilizar esta característica para detectar hiperônimos. Existem alguns desafios teóricos e práticos nesta ideia. Neste trabalho, foram desenvolvidas soluções para estes desafios, propondo um método para detecção de hiperônimo baseado nesta ideia.

Apesar dos métodos não-supervisionados dispensarem o uso de dados anotados durante o treinamento, é importante usar dados anotados para avaliar a qualidade dos modelos treinados. Por isto, neste trabalho, ainda se faz necessário o uso de datasets de relações de hiperônimo. A construção destes datasets é geralmente baseada em bases semânticas existentes, dado que a maioria destas bases incluem a relação de hiperônimo. A base semântica mais conhecida é provavelmente a *WordNet* (Miller, 1995), uma base de palavras de língua inglesa contendo verbos, substantivos e advérbios, agrupados em conjuntos denominados *synsets*. As palavras em um mesmo *synset* são sinônimas; e estes *synsets* são interligados por diferentes relações, sendo uma delas a relação de hiperônimo. Existem outras bases semelhantes, como *MultiWiBi* (Flati et al., 2016), *BabelNet* (Navigli and Ponzetto, 2012) (que engloba a *WordNet*) e *ConceptNet* (Speer et al., 2017). Uma base semântica de língua portuguesa é a *Onto.PT* (Gonçalo Oliveira and Gomes, 2014), que integra conhecimento extraído de várias fontes. Atualmente esta base contém mais de

190 mil *synsets* conectados por mais de 173 mil relações de diferentes tipos, incluindo a relação de hiperônimo. Outra base semântica que contém palavras em português é a *ConceptNet* (Speer et al., 2017). Esta base se diferencia das outras pois, ao invés de relações entre *synsets*, suas relações são diretamente entre palavras.

O *BHearst* foi avaliado experimentalmente em dez datasets na língua inglesa. Para o português, foram desenvolvidos neste trabalho três datasets de detecção de hiperônimo seguindo abordagens semelhantes às usadas na criação dos datasets em inglês. Entre os datasets em inglês o método proposto neste trabalho conseguiu a melhor pontuação em cinco de um total de dez datasets. Já nos três datasets em português esse método conseguiu a melhor pontuação em dois deles.

Dentre as contribuições deste trabalho, se destacam a criação de um método não-supervisionado para a detecção de hiperônimos, a criação de datasets de detecção de hiperônimos no idioma português. Outra contribuição deste trabalho consiste na implementação do algoritmo DIVE utilizando o framework de deep learing *PyTorch* (Paszke et al., 2017) e a versão 3.7 da linguagem Python. Originalmente, este algoritmo foi implementado em *Tensorflow* (Abadi et al., 2015) utilizando Python 2.7. Este algoritmo, que originalmente foi aplicado apenas para língua inglesa, também foi aplicado aqui para o português. Para o treinamento não-supervisionado do DIVE, foi utilizado um corpus contendo mais de 240 milhões de palavras criado a partir de artigos da Wikipédia em português.

1.1 Objetivos

O objetivo geral deste trabalho é avançar o estado-da-arte para a tarefa de detecção de hiperônimo, particularmente, para a língua portuguesa. Desta forma, alguns objetivos específicos que se destacam neste trabalho são:

1. Datasets para Detecção de Hiperônimo:
   (a) disponibilizar datasets de detecção de hiperônimos em português.
2. DIVE:
   (a) disponibilizar um corpus para treinamento não-supervisionado do DIVE;
   (b) disponibilizar uma implementação do algoritmo DIVE utilizando PyTorch;
3. BERT e Padrões de Hearst:

(a) avançar a teoria do BERT no sentido de permitir o cálculo da probabilidade de padrões de Hearst usando este modelo de linguagem;

(b) disponibilizar um algoritmo não-supervisionado baseado na combinação BERT e padrões de Hearst;

4. Prover uma comparação empírica dos algoritmos desenvolvidos.

1.2 Estrutura do Documento

O restante do trabalho está organizado da seguinte forma. No Capítulo 2, são discutidos os principais trabalhos relacionados. No Capítulo 3, são apresentados os referenciais teóricos deste trabalho, a saber: formalização da detecção de hiperônimos, padrões da linguagem, algoritmo DIVE, modelo BERT e datasets utilizados. No Capítulo 4, é descrito o trabalho desenvolvido, como o algoritmo proposto pela combinação do BERT com padrões de Hearst (*BHearst*), a criação dos datasets em português, e informações específicas sobre a implementação do algoritmo DIVE utilizando *PyTorch*. No Capítulo 5, é descrito o ambiente experimental deste trabalho, apresentado os resultados alcançados e uma discussão sobre estes resultados. Por fim, no Capítulo 6 é apresentada a conclusão e alguns trabalhos futuros promissores.

CAPÍTULO

2

Trabalhos Relacionados

Neste capítulo serão abordados os trabalhos relacionados, como a detecção supervisionada de hiperônimos e uma outra modelagem para problemas relacionados à essa relação semântica.

2.1 Detecção Supervisionada de Hiperônimos

Em alguns trabalhos, a detecção de hiperônimos tem sido abordada como um problema de classificação binária usando treinamento supervisionado (Baroni and Lenci, 2011). Um dataset de classificação binária para a relação de hiperônimo é exibido na Tabela 2.1, onde pares de palavras (x, y) são candidatos a possuírem esta relação na forma y é hiperônimo de x.

x	y	**Classe**
catapora	doença	POSITIVO
galáxia	via láctea	POSITIVO
ferramenta	doença	NEGATIVO
lápis	computador	NEGATIVO

Tabela 2.1: Exemplo de dataset para classificação binária para a relação de hiperônimo entre pares de palavras (x, y), onde x é o hipônimo e y é o hiperônimo.

Alguns métodos baseados em vetores de palavra (*word embedding*) foram propostos para esta tarefa. Estes métodos lidam com uma representação distribuída de cada palavra observada e são capazes de identificar relações de hiperônimo mesmo que as palavras não coocorram explicitamente no texto.

Os primeiros trabalhos na modelagem de hiperônimos foram não supervisionados. Alguns trabalhos mais recentes, por outro lado, são supervisionados e usam vetores de palavras (Shwartz et al., 2017).

A tarefa de classificação de hiperônimos usando vetores de palavra descrito em (Sanchez and Riedel, 2017) é exemplificada na Tabela 2.2. São fornecidos dois vetores $\mathbf{v}_x$ e $\mathbf{v}_y$ representando as palavras do par candidato (x, y) e uma saída binária (positivo ou negativo) é esperada. Operações entre os vetores também são utilizadas.

$\mathbf{v}_x$	$\mathbf{v}_y$	**Classe**
$\mathbf{v}_{\text{cat}}$	$\mathbf{v}_{\text{animal}}$	POSITIVO
$\mathbf{v}_{\text{flower}}$	$\mathbf{v}_{\text{jasmine}}$	POSITIVO

Tabela 2.2: Exemplo de um par de instâncias.

Fu et al. (2014) também usam vetores de palavra, mas não é usada uma operação simples. Eles mostram que um deslocamento dos vetores de um par (hipônimo, hiperônimo) é distante do deslocamento de outro par, indicando que a relação de hiperônimo é mais complicada do que um simples deslocamento. Essa dificuldade é mostrada na Tabela 2.3, em alguns casos o deslocamento de um par para o outro é próximo, mas no terceiro caso isso não ocorre.

Exemplos		
$\mathbf{v}_{\text{shrimp}} - \mathbf{v}_{\text{prawn}}$	$\approx$	$\mathbf{v}_{\text{fish}} - \mathbf{v}_{\text{gold fish}}$
$\mathbf{v}_{\text{laborer}} - \mathbf{v}_{\text{carpenter}}$	$\approx$	$\mathbf{v}_{\text{actor}} - \mathbf{v}_{\text{clown}}$
$\mathbf{v}_{\text{laborer}} - \mathbf{v}_{\text{carpenter}}$	$\not\approx$	$\mathbf{v}_{\text{fish}} - \mathbf{v}_{\text{gold fish}}$

Tabela 2.3: Exemplo de deslocamentos entre pares de vetores.

Um problema das modelagens supervisionadas é um fenômeno chamado *Lexical Memorization* (Levy et al., 2015b) no qual os modelos aprendem padrões inerentes aos datasets anotados. O classificador aprende que uma palavra em uma posição específica é um forte indicador da sua classe. Por exemplo, se há muitos exemplos positivos que aparecem na forma (x, animal) a palavra animal está sempre na segunda posição do par, desse modo o classificador aprende que qualquer coisa que aparecer nessa forma será um exemplo positivo, mesmo que o exemplo seja claramente negativo como $(\text{cadeira}, \text{animal})$.

2.2 Descoberta de Hiperônimos

O *International Workshop on Semantic Evaluation* (SemEval) apresenta anualmente diversas tarefas de avaliação semântica, em que os participantes submetem sistemas computacionais para serem avaliados nestas tarefas. No Se-

mEval 2018 (Camacho-Collados et al., 2018), doze tarefas de avaliação semântica foram abordadas. A Tarefa 9 naquele ano foi baseada na formulação de Espinosa-Anke et al. (2016) para *descoberta de hiperônimo*. Nesta tarefa, foi disponibilizado um dataset anotado contendo um conjunto de termos (hipônimos) e, para cada termo, uma lista de hiperônimos. Um exemplo deste dataset é apresentado na Tabela 2.4.

Termo	**Lista de Hiperônimos**
nina simone	musicista, pianista, pessoa
green day	artista, banda de rock, banda
morango	fruta, comida, alimento
desertor	malfeitor, desistente, pessoa má
magnetômetro	instrumento de medição, dispositivo de medição

Tabela 2.4: Hipônimos e lista de hiperônimos correspondentes.

Desta forma, diferente da detecção de hiperônimo baseada em pares de palavras, é fornecido apenas uma palavra e a tarefa consiste em descobrir quais outras palavras são hiperônimos da entrada fornecida. É fornecido também um corpus de onde foram retirados os hipônimos e hiperônimos do dataset. Além disso, um vocabulário de termos é fornecido. Este vocabulário contém todos os termos que ocorrem pelo menos cinco vezes no corpus. Todos os hipônimos e hiperônimos do dataset estão nesse vocabulário.

O dataset com hipônimos e suas listas de hiperônimos foi construído semi-automaticamente a partir de algumas bases semânticas e do corpus. Inicialmente a construção foi feita de forma automática utilizando as mais famosas bases semânticas, como WordNet (Miller, 1995), Wikidata (Vrandečić and Krötzsch, 2014), MultiBibi (Flati et al., 2016), dentre outras. Após isso, é feita uma validação manual, selecionando os melhores exemplos extraídos da dessas fontes. Essa validação é feita inicialmente por humanos em um esforço coletivo. E depois é feito uma segunda validação utilizando especialistas, pessoas que tem o conhecimento técnico no idioma e por consequência na relação semântica.

É comum bases semânticas serem organizadas em conjuntos de palavras sinônimas, os quais são denominados *synsets*. Na Tabela 2.5, é exemplificado um conjunto de synsets (denominados A, B, C, D e E) presentes em uma base semântica fictícia. As relações hipônimo $\rightarrow$ hiperônimo entre estes cinco synsets são representadas na Figura 2.1 através de arestas (A $\rightarrow$ E, por exemplo, indica que E é hiperônimo de A). Para gerar o dataset da tarefa de descoberta de hiperônimo, usa-se estes dois recursos. Por exemplo, para gerar a lista de hiperônimos do termo tigre, são considerados todos os synsets aos quais ele pertence, ou seja, A e B. Então, a lista de hiperônimos de tigre no dataset será composta pela união de todos os termos nos synsets que são hiperônimos de

Synset	**Termos**
A	tigre, canguru, elefante
B	tigre, gato, leão
C	mamífero, felino
D	animal, quadrúpede
E	animal, selvagem

Tabela 2.5: Synsets e seus termos.

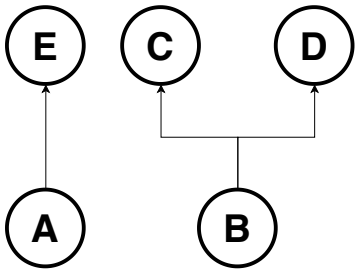


Figura 2.1: Representação em árvore dos synsets

A e B, ou seja, os synsets C, D e E. Desta forma, a lista de hiperônimos de tigre será composta por: animal, mamífero, felino, quadrúpede e selvagem.

Qiu et al. (2018) utilizam um método simples para a descoberta de hiperônimos em um corpus. Eles utilizam vetores de palavra treinados em um corpus de propósito geral e também dados anotados fornecidos pelo SemEval 2018. Para a descoberta de uma lista de hiperônimos para um determinado hipônimo desconhecido, é feito o cálculo de similaridade de cosseno entre o hipônimo não conhecido e os hipônimos do conjunto de treinamento. Os hipônimos do conjunto de treinamento são então ordenadas de acordo com estas similaridades. Suponha, por exemplo, que um hipônimo de entrada seja florianópolis. Após o calculo de similaridade entre o vetor $v_{florianopolis}$ e os vetores dos hipônimos anotados, o mais próximo será o vetor v_{praga}. Assim, os hiperônimos de Florianópolis serão os mesmo hiperônimos de praga.

Apesar da tarefa de descoberta de hiperônimo ser relevante, a criação de um dataset para esta tarefa é substancialmente custosa. Particularmente, a necessidade de revisar manualmente quais relações extraídas da base realmente ocorrem no corpus torna esta tarefa difícil de ser reproduzida em outros contextos. Por isso, neste trabalho, é focado na tarefa de detecção de hiperônimo.

CAPÍTULO

3

Referencial Teórico

Neste capítulo será abordada a tarefa de detecção de hiperônimos junto com uma explicação da principal métrica de avaliação utilizada neste trabalho. Também será explicada as diferentes técnicas em que este trabalho foi baseado, sendo elas: (i) os padrões linguísticos conhecidos como padrões de Hearst e (ii) o modelo contextual de linguagem BERT. Além disso serão apresentados os datasets no idioma inglês utilizados neste trabalho. Por fim, será melhor descrito o algoritmo DIVE utilizado na tarefa não-supervisionada de detecção de hiperônimos.

3.1 Detecção de Hiperônimos

A tarefa de detecção de hiperônimos consiste em, dado dois termos, predizer se há ou não uma relação de hiperônimo entre eles. Para se resolver esta tarefa, foram propostos métodos supervisionados e não-supervisionados. Métodos supervisionados fazem uso de dados anotados, ou seja, pares de termos anotados com rótulo positivo ou negativo. Um grave problema destes métodos é denominado *memorização léxica* (*lexical memorization*) que foi estudado a fundo por Levy et al. (2015b). Em alguns datasets anotados, existem palavras específicas, como fruta por exemplo, que estão frequentemente associadas a pares com a mesma classe (positiva ou negativa). Em Levy et al. (2015b) estas palavras são chamadas de *protótipos* e, nos datasets analisados, elas são geralmente hiperônimos associados a vários hipônimos mas também existem algumas palavras que são frequentemente hipônimos. Este viés nos dados permite que o classificador aprenda estes padrões espúrios, predizendo a classe de um exemplo com base em apenas uma das palavras do par. Isto

dificulta que o classificador aprenda padrões úteis para o problema de detecção de hiperônimo que é, naturalmente, uma relação que depende das duas palavras no par. Outro problema de métodos supervisionados é o custo em criar dados anotados de boa qualidade (dados que condizem com a linguagem). Sendo que a qualidade, do ponto de vista de uma aplicação, é altamente dependente do domínio da aplicação. Isto é, um conjunto de treinamento que seja bom para uma aplicação pode ser ruim para uma segunda aplicação de outro domínio, pois os termos usados neste novo domínio são diferentes.

Devido a essas limitações, existe um esforço na comunidade científica para o desenvolvimento de métodos não-supervisionados. Nestes métodos, os modelos aprendem a detectar a relação de hiperônimo por meio de uma grande quantidade de texto puro, sem anotação explícita desta relação. Na literatura, existem algumas abordagens deste tipo como a abordagem baseada em padrões linguísticos que frequentemente ocorrem entre pares de termos que possuem a relação de hiperônimo (Hearst, 1992). Outras abordagens não-supervisionadas para detecção de hiperônimo são baseadas no conceito de representação distribuída de palavras (Shwartz et al., 2016; Chang et al., 2018). Estas abordagens utilizam o aprendizado de vetores de palavra (*word embeddings*), baseado nos contextos em que a palavra ocorre em um grande corpus, juntamente com restrições adicionais para incorporar a noção de hiperônimo.

Neste trabalho, é proposto um novo algoritmo de detecção de hiperônimo (*BHearst*) que, até certo ponto, combina padrões de Hearst com vetores de palavras. Mais adiante neste capítulo, são explicados essas duas abordagens.

3.1.1 Métrica de Avaliação

Neste trabalho, seguindo trabalhos anteriores (Shwartz et al., 2017; Chang et al., 2018), é usado a métrica *Average Precision* (AP) que é bastante popular na área de aprendizado de máquina em geral. Para facilitar o entendimento, na Figura 3.1, é apresentado um dataset contendo pares de palavras, onde cada par é indicado com a classe positivo se o par contém uma relação de hiperônimo ou negativo caso contrário.

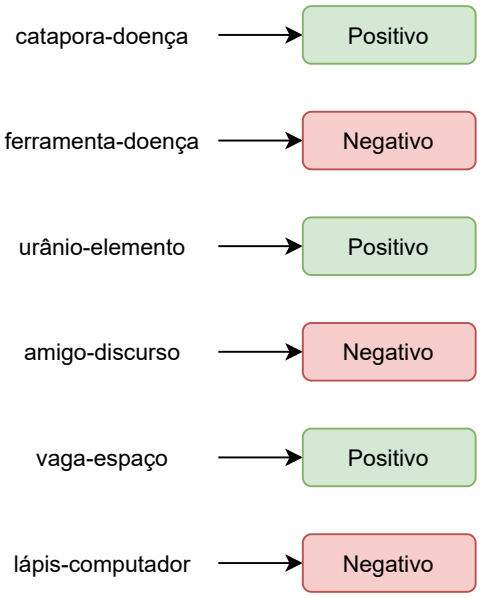


Figura 3.1: Exemplo de dataset para detecção de hiperônimo.

Antes de explicar a métrica AP é preciso entender uma outra métrica mais básica denominada precisão (*precision*) (Olson and Delen, 2008). Considere um classificador binário que, dado um dataset de pares candidatos, *seleciona* alguns destes pares como sendo positivos, ou seja, o classificador prediz que os pares *selecionados* são positivos. Observe a Figura 3.2, na qual são ilustrados os exemplos selecionados por um modelo fictício dentre o dataset da Figura 3.1. Nesta figura são apresentados pares preditos como positivos (conjunto predito) e cada par é indicado se ele de fato é positivo ou negativo (conjunto correto).

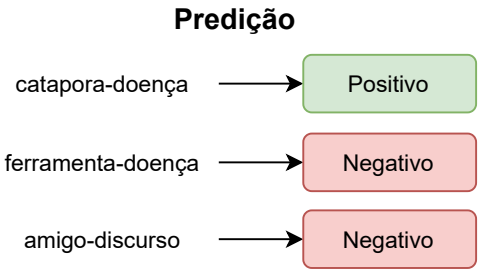


Figura 3.2: Exemplo de três pares preditos como positivos por um modelo fictício. Como apenas um deles é realmente positivo, a precisão é igual a $1/3$.

Precisão é definida como a porcentagem de pares positivos dentre todos os pares selecionados por um modelo. Desta forma, no exemplo anterior, a precisão deste modelo é $1/3$, pois temos um par que de fato é positivo dentre os três selecionados. Mais precisamente, a definição de precisão é dada pela Equação 3.1:

$$\text{precisao} = \frac{TP}{|S|}, \tag{3.1}$$

onde S é o conjunto de pares selecionados pelo modelo e TP (*true positives* ou

verdadeiros positivos) é a quantidade de pares positivos em S. No exemplo, temos $S = \{$catapora-doença, ferramenta-doença, amigo-discurso$\}$ (portanto $|S| = 3$) e $TP = 1$ (catapora-doença).

Por definição, o cálculo da precisão considera apenas os pares selecionados por um modelo. Por outro lado, a maioria dos classificadores binários fornece, para um dado par de entrada, não somente uma saída binária, mas um valor real que expressa a probabilidade (ou a confiança) do par ser positivo. Este valor é denominado como *pontuação* de um par candidato, pois nem sempre este valor é uma probabilidade. Geralmente, quando um classificador binário é usado na prática, é preciso definir um limiar de pontuação e, desta forma, somente os pares com pontuação maior ou igual ao limiar definido são selecionados pelo modelo. Entretanto, o limiar ideal depende da aplicação em questão e, sem um limiar específico, fica difícil comparar diferentes modelos usando apenas a métrica de precisão. A métrica AP vem justamente no sentido de sanar esta limitação. Simplificadamente, AP é uma média da precisão para todos os limiares possíveis.

Para calcular AP de um modelo, é preciso ordenar todos os pares do dataset em ordem decrescente da pontuação dada pelo modelo. Na Figura 3.3, é apresentado uma lista ordenada dos pares do nosso exemplo de acordo com a pontuação dada por um modelo fictício.

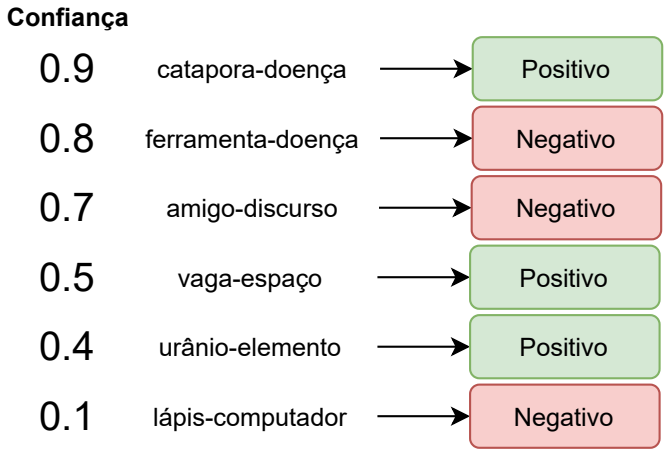


Figura 3.3: Lista de predição contendo os pares do dataset da Figura 3.1 ordenados de acordo com a pontuação dada por um modelo fictício.

Com base nessa lista ordenada, também chamada de *lista de predição*, é calculada a precisão@k para diferentes valores de $k \in \{1, 2, \ldots, n\}$, onde n é o número de pares no dataset (no exemplo, temos $n = 6$). Como pode ser observado na Figura 3.4, a precisão@k é calculada selecionando-se os k primeiros pares na lista de predição. Para calcular AP, é considerado precisão@k somente quando o k-ésimo par da lista de predição é positivo. Portanto, no exemplo, é calculado precisão@k para $k \in \{1, 4, 5\}$. Quando $k = 1$, o modelo

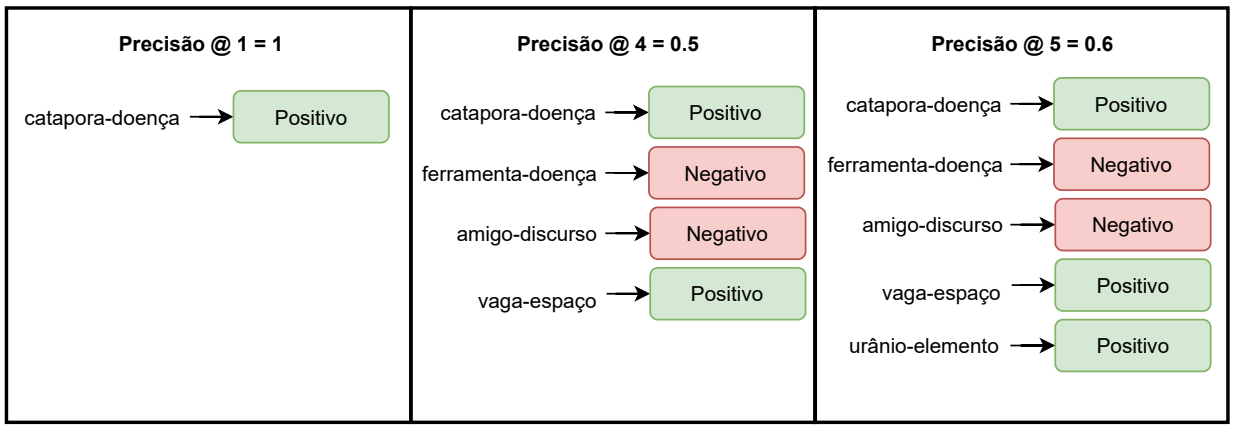


Figura 3.4: Valores de precisão@k para os valores k tal que o k-ésimo par, na lista de predição, é positivo.

seleciona um par positivo e temos que precisão@$1 = 1$. Quando $k = 4$, o modelo seleciona dois pares positivos e temos que precisão@$4 = 2/4 = 0.5$. E, por fim, quando $k = 5$, o modelo seleciona todos os três pares positivos e temos que precisão@$5 = 3/5$. A AP é a média destes valores, portanto, no exemplo anterior, temos AP=$(1+2/4+3/5)/3 = 0,7$. Formalmente, a AP é definida como a média da precisão@k para $k \in P$, onde P é o conjunto de valores $k \in \{1, 2, \ldots, n\}$ para os quais o k-ésimo elemento da lista de predição é positivo. Desta forma, AP é definida como:

$$\text{AP} = \frac{\sum_{k \in P} \text{precisao@}k}{|P|}.$$

A métrica AP é também conhecida como AP@all. Para Shwartz et al. (2017), esta métrica é mais informativa sobre a qualidade de uma lista de predição. Neste trabalho é seguido essa ideia, pois AP é independente do limiar de predição do classificador binário, expressando em um único número a capacidade de um modelo atribuir pontuações maiores para pares positivos do que para pares negativos.

3.2 Padrões de Hearst

Alguns trabalhos (Hearst, 1992; Roller et al., 2018) utilizam padrões linguísticos para detectar a relação de hiperônimo. Estas técnicas exploram a coocorrência de um par de palavras em um texto de acordo com alguns padrões que expressam, de alguma maneira, esta relação. Os padrões mais conhecidos são os propostos em Hearst (1992), os quais são expressões regulares que indicam uma relação de hiperônimo. Nos padrões abaixo, por exemplo, existe um forte indício de que a palavra y é hiperônimo de x.

- such y as x - Este padrão pode ser aplicada à seguinte frase: ... works by such authors as Shakespeare. Ele implica que existe uma relação de

hiperônimo entre o par de termos: (author, Shakespeare).

- x or other y - Este padrão pode ser aplicada à seguinte frase: ... this apple or other fruits .... Ele implica que existe uma relação de hiperônimo entre o par de termos: (apple, fruit).

Existem outros padrões descritos em Hearst (1992). Alguns padrões são mais complexos, usando expressões regulares com casamento opcional, casamento múltiplo, dentre outros recursos de expressão regular. Um problema de abordagens baseadas em padrões é a impossibilidade de capturar relações que não se encaixam exatamente aos padrões. Para a relação ser reconhecida, as palavras têm que coocorrer na exata configuração dada pelo padrão (Roller et al., 2018). Estes métodos têm alta precisão, mas baixa revocação (*recall*) (Navigli and Velardi, 2010), justamente por não conseguir capturar as relações de hiperônimo que não coocorrem de acordo com algum padrão considerado.

Seguindo a abordagem de Hearst (1992), Roller et al. (2018) extraíram e listaram alguns padrões em inglês. Uma das abordagem utilizada é simples e baseada em contagem. Com base nos padrões de linguagem é calculada a pontuação para um par candidato. Utilizando diversos padrões é realizada uma busca desse par em um corpus e a pontuação do par é definida pela quantidade de vezes que ele aparece em um corpus com esses padrões divido pela quantidade total de extrações realizadas de todos os pares. Por meio desta abordagem, é esperado que os pares positivos tenham uma pontuação alta, já que somente eles poderiam ser recuperados usando esses padrões.

Neste trabalho, selecionamos padrões listados em ambos os trabalhos Hearst (1992) e Roller et al. (2018). Os padrões selecionados foram traduzidos para o português e acrescentamos alguns padrões óbvios como x é um y e x é um tipo de y. Na Tabela 3.1 são exibidos os padrões em inglês e na Tabela 3.2 são mostrados os padrões em português que foram usados no nosso trabalho.

x or some other y	x or any other y	x and any other y
x is a type of y	x which is kind of y	x and some other y
x is a y	x a special case of y	x which is a example of y
x and others y	x which is called y	x or others y
x which is a class of y	x , a y	x including y

Tabela 3.1: Padrões usados em inglês.

3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) é um modelo profundo de representação de linguagem que pode ser treinado em um

x ou algum outro y	x ou qualquer outro y	x e qualquer outro y
x é um tipo de y	x que é um tipo de y	x e algum outro y
x é um y	x é um caso especial de y	x que é um exemplo de y
x e outros y	x que é chamado de y	x ou outros y
x que é uma classe de y	x , um y	x incluindo y

Tabela 3.2: Padrões usados em português.

corpus de texto puro (Devlin et al., 2018). Uma das principais características deste modelo é a bidirecionalidade, ou seja, a representação de uma palavra depende tanto do contexto esquerdo quanto do direito em todas as camadas do modelo. Um modelo pré treinado é um modelo treinado em tarefas específicas, por exemplo, dada a sentença "A universidade tem muitos x", o modelo, sabendo todas as palavras anteriores, pode predizer a palavra x como "alunos". A partir desse modelo é feito um ajuste fino para utilizá-lo em outras tarefas. Uma vantagem do BERT quando comparado a outros modelos de linguagem, como o GPT (Radford et al., 2018) por exemplo, é a utilização de uma abordagem bidirecional em vez de ler a entrada de texto sequencialmente, da esquerda pra direita. Para usar essa ideia de bidirecionalidade, a rede é treinada usando um modelo de linguagem mascarado (*masked language model* ou, simplesmente, MLM). No treinamento do MLM, algumas palavras de uma sentença são mascaradas e o objetivo é predizer essas palavras. Isto é, o modelo prediz qual é a palavra mascarada sabendo quais são as outras palavras que aparecem ao redor. Desta forma, o modelo não utiliza apenas as palavras que estão antes da palavra mascarada (contexto esquerdo), mas também as palavras que estão depois dela (contexto direito).

Para se obter um modelo robusto pré treinado, o BERT, além de utilizar a ideia de mascarar e predizer as palavras, utiliza-se de uma outra ideia. Dada duas sentenças A e B, predizer se a sentença B é a próxima sentença depois de A. Isso é feito para o modelo conseguir compreender melhor a relação entre sentenças. Essa tarefa é chamada predição de próxima sentença (*next sentence prediction* ou, simplesmente, NSP). No treinamento não supervisionado, metade das amostras são compostas por duas sentenças contínuas (exemplos positivos para a NSP); e a outra metade são de sentenças não contínuas (exemplos negativos). Desta forma o modelo BERT é pré treinado em duas tarefas não supervisionadas. A primeira tarefa consiste em predizer palavras mascaradas em uma sentença, como no exemplo 3.1 no qual o modelo deve predizer a palavra mascarada como final. E a segunda tarefa é predizer se uma sentença é a próxima em relação à outra. No exemplo 3.2, existe uma relação entre as sentenças. Já no exemplo 3.3, as duas sentenças não parecem ter vindo do mesmo texto. Essas duas tarefas não supervisionadas são treinadas conjuntamente.

Exemplo 3.1 *Muitas pessoas viajaram no [MASK] de semana*

Exemplo 3.2 *Eu amo o mar [SEP] Ele é imenso*

Exemplo 3.3 *Os pedestres respeitam o trânsito [SEP] O computador faz milhões de cálculos*

O BERT utiliza um algoritmo de tokenização baseado em subpalavras denominado *WordPiece* (Wu et al., 2016). Este algoritmo, assim como outros baseados em subpalavras, lida melhor com palavras desconhecidas (*out of vocabulary* ou, simplesmente, OOV), evitando que o modelo transforme estas palavras em um símbolo padrão. De maneira simplificada, o WordPiece divide uma palavra na menor quantidade possível de subpalavras tal que cada subpalavra seja suficientemente frequente. Desta forma, palavras frequentes não são divididas (são tokenizadas em apenas uma subunidade), enquanto palavras raras são divididas em algumas subpalavras que, por sua vez, são mais frequentes. Por exemplo, as palavras mão, antivírus, hipertensão e decrescer são tokenizadas, respectivamente, como mão, anti-vírus, hiper-tensão e de-cresc-er.

Durante o treinamento não supervisionado do BERT, uma entrada é denominada *sequência* e é composta por duas sentenças concatenadas. O primeiro passo consiste em tokenizar a sequência de entrada usando o algoritmo WordPiece. As duas sentenças são separados pelo token especial (SEP). Então algumas palavras são sorteadas para serem mascaradas, o que corresponde a substituí-las pelo token especial (MASK). Finalmente, o token especial (CLS) é incluído no início da sequência. Como pode ser observado na Figura 3.5, estes passos são responsáveis por criar a entrada do BERT. Em seguida, esta

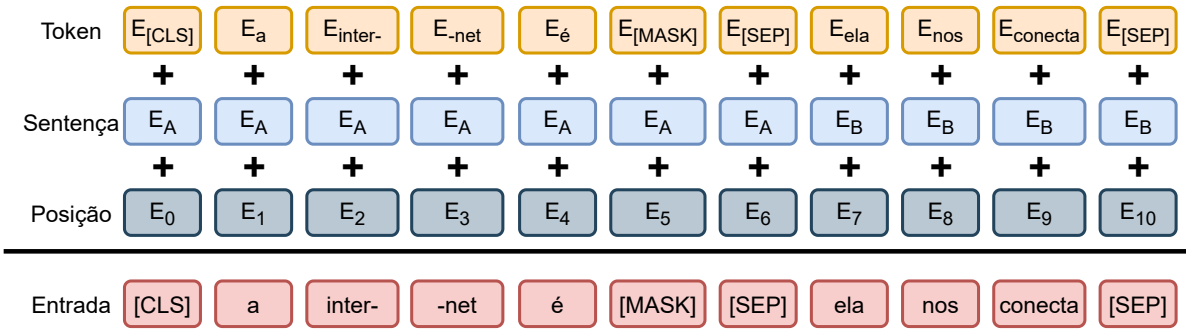


Figura 3.5: Ilustração da representação da entrada no BERT.

entrada é processada pela camada de entrada do BERT que transforma cada token da sequência de entrada em um vetor numérico. Para cada token, seu vetor é a soma de três vetores: token, sentença e posição. Cada um destes três vetores vêm de uma camada treinável do modelo BERT. O vetor de sentença indica a qual sentença o *token* pertence (a primeira sentença é denominada "A"

e a segunda “B”). O vetor de posição indica a posição do token na sequência de entrada. Originalmente, o BERT foi treinado com o máximo de 512 posições, ou seja, a camada de vetor de posição é composta por 512 vetores, um vetor para cada posição. Esta é a principal razão que limita o comprimento da entrada do modelo BERT. O vetor de posição é importante porque o BERT é baseado na arquitetura Transformers (Vaswani et al., 2017) que permite a consideração de todos os tokens ao mesmo tempo mas ignora suas posições. Dessa forma, a representação de um *token* é a soma dos vetores de: token, sentença e posição. Na Figura 3.6, é ilustrada a arquitetura geral do modelo BERT. Neste exemplo, por questões de apresentação, são exibidas apenas os

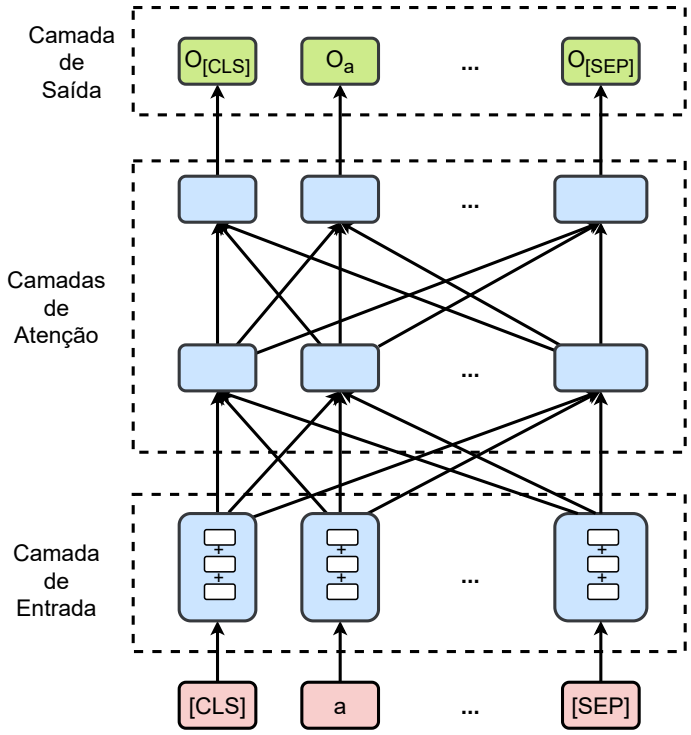


Figura 3.6: Representação das camadas do BERT.

dois primeiros e o último tokens. A camada de entrada é a responsável por receber a sequência de tokens e gerar uma representação vetorial *independente* para cada token, como mostrado na Figura 3.5. Estas representações são então tomadas como entrada para a primeira camada de atenção. Uma sequência de camadas de atenção são aplicadas uma após a outra. A saída da última camada de atenção consiste em um vetor de representação *contextual* O_v para cada token v da entrada.

Durante o treinamento do MLM, uma camada de classificação multi-classe é aplicada à representação de cada token mascarado. Considere, por exemplo, a sequência $X = (x_1, x_2, \ldots, x_6) = (\text{tigre}, \text{é}, \text{um}, \text{tipo}, \text{de}, \text{animal})$. Na Figura 3.7, é ilustrado a camada de classificação quando o sexto token desta sequência é mascarado ($x_6 = \text{animal}$). Esta sequência mascarada é denotada $X_{\setminus 6} = (\text{tigre}, \text{é}, \text{um}, \text{tipo}, \text{de}, (\text{MASK}))$. O intuito da camada de classificação é predizer qual a

palavra mascarada, ou seja, a saída do MLM para um token mascarado é uma distribuição de probabilidade sobre todos os tokens do vocabulário. Isto é, para cada token v do vocabulário V do BERT, há uma probabilidade $p(v \mid X_{\setminus 6})$ deste token ocorrer no contexto $X_{\setminus 6}$. No exemplo, os tokens em amarelo são alguns dos tokens do vocabulário.

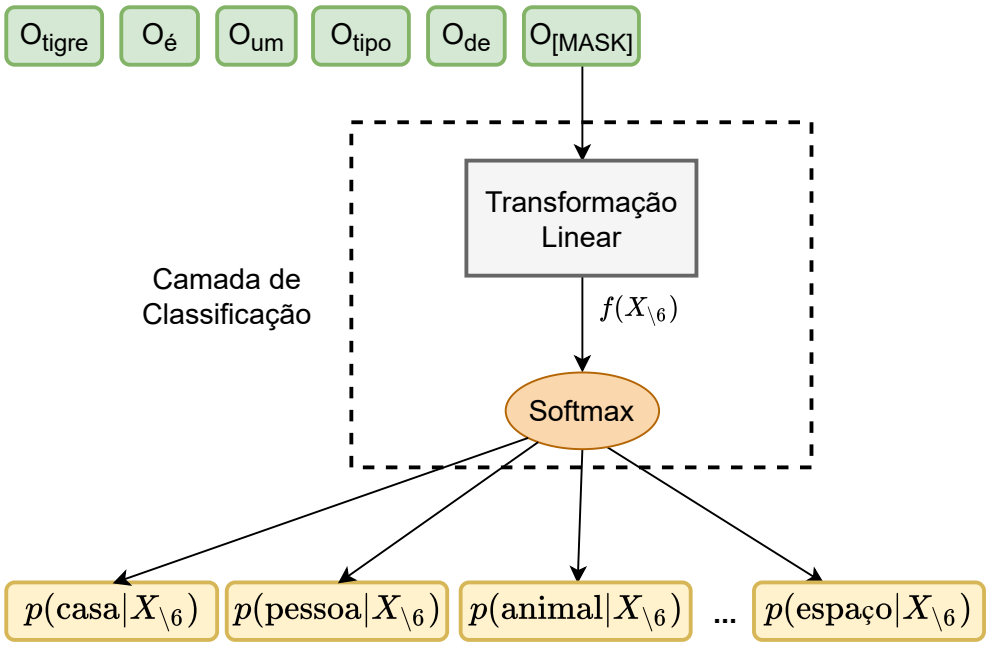


Figura 3.7: Camada de classificação do MLM para um token mascarado.

Como pode ser observado na Figura 3.7, a camada de classificação do MLM é composta por: (i) uma transformação linear (camada totalmente conectada) que transforma o vetor de representação da palavra mascarada ($O_{\text{[MASK]}}$) em um vetor $f(X_{\setminus 6})$ contendo uma pontuação para cada palavra do vocabulário; e (ii) uma função softmax para normalizar o vetor de pontuações em uma distribuição de probabilidade (soma igual a um). O vetor $f(X_{\setminus 6}) \in \mathbb{R}^{|V|}$, também conhecido como *logits*, é a saída da camada linear do classificador MLM, ou seja, o vetor contendo um peso $f(X_{\setminus 6})[v]$ (não normalizado) para cada palavra v do vocabulário V. Desta forma, a distribuição de probabilidades é dada através da normalização dos *logits* por meio da função softmax. Mais especificamente, a probabilidade de uma palavra $v \in V$ ocorrer na posição mascarada é dada por:

$$p(v \mid X_{\setminus 6}) = \frac{\exp(f(X_{\setminus 6})[v])}{\sum_{v' \in V} \exp(f(X_{\setminus 6})[v'])}.$$

Durante o treinamento do MLM, o objetivo é maximizar a probabilidade do token correto. No exemplo anterior, isto corresponde a maximizar o valor de $p(\text{animal} \mid X_{\setminus 6})$.

Após o treinamento do MLM, a camada de classificação é descartada na maioria das aplicações pois, nestes casos, o BERT é usado apenas para forne-

cer uma representação contextual de uma sequência de tokens. Desta forma, apenas as representações de saída O_v são usadas como entrada para um modelo específico. Por outro lado, é possível usar a camada de classificação treinada por meio do MLM para calcular a probabilidade de um token mascarado em um contexto específico, assim como é feito no treinamento do MLM visto acima. É esta característica que é usada para derivar um algoritmo de detecção de hiperônimo (*BHearst*) neste trabalho. A sequência na Figura 3.7 é um bom exemplo para entender a ideia básica do algoritmo proposto. A frase x é um tipo de y é um padrão de Hearst. Suponha que desejamos avaliar a probabilidade do par (tigre, animal) possuir a relação hipônimo-hiperônimo. Nesse exemplo, substituímos x por tigre (hipônimo do par em questão) e mascaramos o token y (posição do hiperônimo no padrão). Desta forma, o modelo BERT devolve a distribuição de probabilidade na posição mascarada para todas as palavras do vocabulário. Dentre as palavras do vocabulário temos a palavra animal que é a candidata a hiperônimo do nosso par em questão. Desta maneira, é obtido um valor importante: a probabilidade da palavra animal ocorrer na posição y deste contexto (padrão de Hearst) dado que a palavra tigre aparece na posição x. Esta probabilidade é dada por $p(\text{animal} \mid X_{\setminus 6})$. Existem alguns desafios relevantes para aplicar esta ideia básica para a detecção de hiperônimo. Estes desafios são discutidos, e algumas soluções são propostas, na próxima seção e no capítulo seguinte.

3.4 BERT pode Falar

Diferente de tradicionais modelos de linguagem que são treinados de forma a predizer uma palavra em uma sequência dadas as palavras anteriores, o BERT é treinado usando a abordagem MLM. Nesta abordagem, o modelo de linguagem é treinado para predizer uma palavra dadas as palavras à esquerda e à direita dela. Como mencionado acima, isto é uma vantagem deste modelo quando comparado a modelos tradicionais (unidirecionais) de linguagem. Por outro lado, esta característica dificulta uma aplicação que está diretamente relacionada a este trabalho: calcular a probabilidade de uma sequência de palavras.

Recentemente, Wang and Cho (2019) mostraram que o BERT permite o cálculo da probabilidade $p(X)$ de uma sequência $X = (x_1, x_2, \ldots, x_T)$ de comprimento T. Mais especificamente, eles mostram que o BERT pode ser interpretado como um *Markov Random Field* (MRF) (Jernite et al., 2015), tal que o potencial $\phi(X)$ sobre toda a sequência X pode ser decomposto em T log-

potenciais:

$$\phi(X) = \prod_{t=1}^{T} \phi_t(X) = \exp\left(\sum_{t=1}^{T} \log \phi_t(X)\right). \tag{3.2}$$

Desta forma, devido às propriedades do MRF, temos que:

$$p(X) \propto \phi(X), \tag{3.3}$$

isto é, a probabilidade $p(X)$ é proporcional ao potencial do MRF sobre a sequência completa. Para obter $p(X)$, é necessário normalizar o potencial $\phi(X)$ da seguinte forma:

$$p(X) = \frac{\phi(X)}{Z}, \tag{3.4}$$

onde Z é a constante de normalização que transformará o potencial em uma probabilidade. Mais adiante, é falado um pouco mais sobre esta constante.

Agora, é mostrado como os log-potenciais $\log \phi_t(X)$ são obtidos a partir do BERT. Para isto, considere novamente a frase tigre é um tipo de animal. É necessário mascarar cada palavra nesta frase, uma a uma, gerando assim uma sequência para cada palavra da frase. Na Tabela 3.3, é mostrado estas seis sequências.

$X_{\setminus t}$	**Sequência Mascarada**					
$X_{\setminus 1}$	[MASK]	é	um	tipo	de	animal
$X_{\setminus 2}$	tigre	[MASK]	um	tipo	de	animal
$X_{\setminus 3}$	tigre	é	[MASK]	tipo	de	animal
$X_{\setminus 4}$	tigre	é	um	[MASK]	de	animal
$X_{\setminus 5}$	tigre	é	um	tipo	[MASK]	animal
$X_{\setminus 6}$	tigre	é	um	tipo	de	[MASK]

Tabela 3.3: Exemplo de uma frase na qual cada palavra é mascarada para gerar uma sequência de entrada para o BERT. $X_{\setminus t}$ representa uma sequência X com seu t-ésimo token mascarado.

Em seguida, é aplicado o BERT a cada uma destas sequências e assim é obtido seis vetores de *logits*: $f(X_{\setminus t})$, um para cada posição $t = 1, 2, \ldots, 6$ da sequência. Particularmente, o interesse está nas seis pontuações:

$$\begin{aligned} &f(X_{\setminus 1})[\text{tigre}], \\ &f(X_{\setminus 2})[\text{é}], \\ &f(X_{\setminus 3})[\text{um}], \\ &\cdots, \\ &f(X_{\setminus 6})[\text{animal}], \end{aligned}$$

ou seja, na pontuação de cada palavra da sequência original quando esta

palavra é mascarada. Wang and Cho (2019) definem os log-potenciais como:

$$\begin{aligned}
\log \phi_1(X) &= f(X_{\setminus 1})[\text{tigre}], \\
\log \phi_2(X) &= f(X_{\setminus 2})[\text{é}], \\
\log \phi_3(X) &= f(X_{\setminus 3})[\text{um}], \\
&\cdots, \\
\log \phi_6(X) &= f(X_{\setminus 6})[\text{animal}].
\end{aligned}$$

Considere agora o caso mais genérico de uma sequência $X = (x_1, x_2, \ldots, x_T)$ de comprimento T. Assim como no caso específico, após aplicar o BERT nas T sequências mascaradas $X_{\setminus 1}$, $X_{\setminus 2}$, ..., $X_{\setminus T}$, obtemos um vetor de logits $f(X_{\setminus t})$ para cada $t = 1, 2, \ldots, T$. Desta maneira, podemos definir:

$$\log \phi_t(X) = f(X_{\setminus t})[x_t], \tag{3.5}$$

para $t = 1, 2, \ldots, T$. Combinando a equação acima com as Equações 3.2 e 3.3, podemos então definir a probabilidade desejada como:

$$p(X) = \frac{\exp\left(\sum_{t=1}^{T} f(X_{\setminus t})[x_t]\right)}{Z}. \tag{3.6}$$

Resta agora definir a constante de normalização Z. Considerando apenas sequências de tamanho T e usando a Equação 3.4, podemos definir Z de uma maneira simples:

$$Z = \sum_{X'} \phi(X') = \sum_{X'} \exp\left(\sum_{t=1}^{T} f(X'_{\setminus t})[x'_t]\right), \tag{3.7}$$

isto é, a soma dos potenciais de todas as sequências possíveis $X' = (x'_1, x'_2, \ldots, x'_T)$ de comprimento T. Apesar da simplicidade, calcular Z é impraticável na maioria dos casos. A quantidade de sequências deste tipo, considerando todas as combinações de palavras do vocabulário, é igual a $|V|^T$. Considerando que o tamanho de V é geralmente em torno de algumas dezenas de milhares, calcular exatamente esta constante de normalização é intratável mesmo para valores pequenos de T. Outro problema desta abordagem é que ela considera apenas sequências de um tamanho fixo T. No próximo capítulo, é apresentado algumas ideias para contornar estes problemas para o caso de estimar a probabilidade de alguns pares de palavras em um contexto específico (padrão de Hearst).

3.5 Bases de Semântica Lexical

Existem diversas bases de relações semânticas entre palavras que incluem a relação de hiperônimo. A mais conhecida é, provavelmente, a WordNet (Miller, 1995) na qual são relacionados substantivos, verbos, adjetivos e advérbios em conjuntos de palavras sinônimas denominados *synsets*. Esses synsets são interligados por relações semânticas, sendo uma delas a relação de hiperônimo. A relação mais frequente entre os synsets presentes na WordNet é a relação de hipônimo e hiperônimo. Através desta relação, os synsets mais genéricos ({animal, mamífero}, por exemplo) são relacionados aos synsets mais específicos ({cachorro, gato}).

BabelNet (Navigli and Ponzetto, 2012) é uma enciclopédia multilíngue e também uma base semântica, semelhante à WordNet. Ela tem cerca de 16 milhões de *babel synsets* sendo que cada um deles contém palavras sinônimas que representam o mesmo sentido em diferentes idiomas. Esta rede agrupa conhecimento de diversas outras fontes como: WordNet, Wikipedia,OmegaWiki (um dicionário colaborativo multilíngue), ImageNet (uma base de dados de imagens organizada de acordo com a hierarquia da WordNet), entre outras.

Onto.PT (Gonçalo Oliveira and Gomes, 2014) é uma base de relações semânticas em português. Ela integra conhecimentos extraídos de dicionários online e possui mais de 109 mil synsets conectados por mais mais de 173 mil relações de tipos diferentes. A Onto.PT é maior do que a WordNet, incluindo mais tipos de relações. Isto é devido ao fato de ela ser construída de forma automática.

ConceptNet (Speer et al., 2017) é um grafo de conhecimento que conecta palavras e frases com arestas rotuladas por diversas relações e com suporte para vários idiomas, incluindo o português. Esta base usa diversas fontes de conhecimento, como Wikitionary, WordNet, entre outros. A ConceptNet inclui 3.757 relações de hiperônimo entre palavras em português (Lima et al., 2018).

3.6 DIVE

A hipótese de inclusão distributiva (*distributional inclusion hypothesis* ou, simplesmente, DIH), proposta por Geffet and Dagan (2005), sugere que, dado um corpus suficientemente grande e diverso, o conjunto de contextos de um hipônimo neste corpus (frases onde esta palavra é usada, por exemplo) tende a ser um subconjunto do conjunto de contextos dos seus hiperônimos. Na Tabela 3.4 é ilustrado um exemplo. Em princípio, qualquer adjetivo aplicado a poodle também pode ser aplicado a cachorro, porque cachorro é hiperônimo de poodle (ambos podem ser obedientes). No entanto, o inverso não é

necessariamente verdade, um cachorro pode ter pelos lisos mas um poodle não. Portanto, a palavra cachorro tende a ser usada em um conjunto de con-

Termo	**Contextos**
Poodle	O Poodle deve comer ração regularmente. Seu Poodle faz xixi no lugar certo?
Cachorro	O cachorro deve comer ração regularmente. Seu cachorro faz xixi no lugar certo? Cachorro assusta pessoas na praça. A segurança da casa é feita pelo cachorro.

Tabela 3.4: Exemplos de contextos de poodle e seu hiperônimo cachorro.

textos mais amplo do que poodle. A DIH sugere que o conjunto de contextos de um hiperônimo tende a conter o conjunto de contexto de seus hipônimos. Na tabela anterior é possível visualizar um conjunto de contextos contendo o outro. A palavra cachorro inclui os contextos da palavra poodle. Isto ocorre justamente porque a palavra cachorro é mais genérica do que poodle. Esta hipótese sugere um novo tipo de abordagem para detecção de hiperônimo.

Distributional inclusion vector embedding (DIVE) é um método não supervisionado que calcula vetores de palavras (*word embeddings*) com base na DIH (Chang et al., 2018). Estes vetores apresentam características que preservam esta propriedade como, por exemplo, a não negatividade. O DIVE é utilizado para a detecção da relação de hiperônimo entre palavras, já que utiliza essas propriedades da inclusão de contextos. Esse método foi avaliado em onze datasets em inglês e conseguiu superar outros vetores de palavra não-supervisionados como o *Gaussian embedding* (Vilnis and McCallum, 2014) e o *Word2vec* (Mikolov et al., 2013). Neste trabalho, o algoritmo DIVE foi implementado, aplicado aos novos datasets em português, e os resultados alcançados foram comparados aos resultados obtidos pelo *BHearst*.

Para definir formalmente a DIH, definimos a contagem $\#(x, c)$ de coocorrências contextuais entre as palavras x e c. Desta maneira, a contagem $\#(x, c)$ de um hipônimo x tende a ser menor ou igual do que a contagem $\#(y, c)$ de um hiperônimo correspondente y:

$$x \preceq y \iff \forall c \in V, \#(x, c) \leq \#(y, c),$$

em que $x \preceq y$ significa x é hipônimo de y (e y é hiperônimo de x), V é o conjunto do vocabulário e $\#(x, c)$ indica o número de vezes que uma palavra c ocorre no contexto da palavra x, sendo que o contexto de uma palavra é definido como uma janela de tamanho $|W|$ em um corpus (Chang et al., 2018). Essa é a propriedade de inclusão definida para o DIVE.

O objetivo do DIVE é produzir um vetor que preserve a propriedade de inclusão, de modo que o vetor de palavra $\mathbf{v}_y \in \mathbb{R}^D$ de um hiperônimo y é maior ou igual ao vetor $\mathbf{v}_x \in \mathbb{R}^D$ do seu hipônimo x em todas as dimensões $i = 1, 2, \ldots, D$. Mais especificamente, podemos definir esta propriedade como:

$$x \preceq y \iff \mathbf{v}_x[i] \leq \mathbf{v}_y[i], \forall i = 1, 2, \ldots, D,$$

em que D é o número de dimensões no espaço vetorial. Também foram adicionadas restrições de não negatividade: $\mathbf{v}_x[i] \geq 0, \forall i \in \{1, ..., D\}$. Segundo os autores do DIVE, isto aumenta a interpretabilidade do vetor pois cada dimensão corresponde aproximadamente a um tópico. A hipótese de inclusão distributiva sugere que palavras genéricas, como hiperônimos, aparecem em contextos mais diversificados, e como o DIVE preserva esta hipótese, os vetores de uma palavra genérica, que é hiperônimo de muitas outras palavras, tende a ter grandes valores nessas dimensões.

Os métodos populares para a criação de vetores de palavras não supervisionados são baseados em fatoração de matriz (Levy et al., 2015a). Esses métodos consideram uma estatística de coocorrência entre a palavra w e a palavra de contexto c representada por uma matriz M tal que $M[w, c]$ corresponde à estatística escolhida. A definição do contexto é arbitrária. Mas a definição mais comum, e que é usada no DIVE, é de uma janela de tamanho fixo ao redor da palavra w. Ou seja, c são todas as palavras que coocorrem com w em uma janela de tamanho fixo. O tamanho da janela é geralmente um hiperparâmetro a ser definido. A estatística normalmente utilizada em $M[w, c]$ é a *pointwise mutual information* (PMI) (Levy et al., 2015a). A estatística aplicada à matriz M do DIVE é dada por:

$$M[w, c] = \log\left(\frac{P(w, c)}{P(w) \cdot P(c)} \cdot \frac{\#(w)}{k_I \cdot Z}\right) = \log\left(\frac{\#(w, c) \cdot |V|}{\#(c) \cdot k_I}\right), \tag{3.8}$$

onde $P(w, c) = \frac{\#(w,c)}{|D|}$, $P(w) = \frac{\#(w)}{|D|}$, $P(c)$ é análoga à $P(w)$, $\#(w) = \sum_{c \in V} \#(w, c)$ é a frequência que a palavra w coocorre com qualquer outra em um determinado tamanho de janela, $|D| = \sum_{w \in V} \sum_{c \in V} \#(w, c)$ é o número de pares de palavras que coocorrem no corpus, k_I é uma constante, $Z = \frac{|D|}{|V|}$ é a média de frequência de todas as palavras, $|V|$ é o tamanho do vocabulário e $\frac{\#(w)}{Z}$ é chamado de *inclusion shift*.

O DIVE utiliza *non-negative matrix factorization* (NMF) (Lee and Seung, 2001). O NMF permite fatorar a matriz M em vetores de palavras de tal forma que $\mathbf{v}_w^\top \mathbf{v}_c \approx M[w, c]$, para os vetores de palavra $\mathbf{v}_w, \mathbf{v}_c \in \mathbb{R}^D$ das palavras $w, c \in V$, respectivamente. A propriedade de inclusão do DIVE é certamente mantida se a matriz for reconstruída perfeitamente, isto é, se $\mathbf{v}_w^\top \mathbf{v}_c = M[w, c]$. Desta ma-

neira, se o vetor $\mathbf{v}_y$ do hiperônimo y é maior ou igual ao vetor $\mathbf{v}_x$ do hipônimo x em todas as dimensões ($\mathbf{v}_y[i] \geq \mathbf{v}_x[i], i = 1, 2, \ldots, D$), então $\mathbf{v}_y^\top \mathbf{v}_c \geq \mathbf{v}_x^\top \mathbf{v}_c$ para uma palavra de contexto $c \in V$, já que os vetores de palavras são não negativos.

Foi demonstrado em Levy et al. (2014) que o algoritmo *word2vec skipgram* treinado com *negative sampling* é equivalente a fatorar uma matriz $M^{'}$ definida como:

$$M'[w, c] = \log\left(\frac{P(w, c)}{P(w) \cdot P(c)} \cdot \frac{1}{k'}\right). \tag{3.9}$$

Substituindo k' por $\frac{k_I \cdot Z}{\#(w)}$ temos:

$$M'[w, c] = \log\left(\frac{P(w, c)}{P(w) \cdot P(c)} \cdot \frac{\#(w)}{K_I \cdot Z}\right). \tag{3.10}$$

Observe que temos assim $M = M'$. Os vetores de palavras do DIVE são otimizados usando uma variação da função objetivo usada pelo *word2vec skip-gram* com *negative sampling* (SGNS). A função objetivo do SGNS é:

$$\begin{aligned} l_{SGNS} = &\sum_{w \in V} \sum_{c \in V} \#(w, c) \log \sigma\left(\mathbf{v}_w^\top \mathbf{v}_c\right) + \\ &\sum_{w \in V} k^{'} \sum_{c_N \in V} \#(w, c)\, \mathbb{E}_{c_N \sim p_D}\left[\log \sigma\left(-\mathbf{v}_w^\top \mathbf{v}_{c_N}\right)\right], \end{aligned} \tag{3.11}$$

onde c_N é uma palavra amostrada usando *negative sampling*; $\mathbf{v}_w, \mathbf{v}_c, \mathbf{v}_{c_N} \in \mathbb{R}^D$ são vetores das palavras w, c e c_N, respectivamente; σ é a função logística (sigmoid); e k' é um hiperparâmetro que indica a taxa entre exemplos positivos e negativos. Desta maneira, a função objetivo do DIVE é:

$$\begin{aligned} l_{DIVE} = &\sum_{w \in V} \sum_{c \in V} \#(w, c) \log \sigma\left(\mathbf{v}_w^T \mathbf{v}_c\right) + \\ &k_I \sum_{w \in V} \frac{Z}{\#(w)} \sum_{c \in V} \#(w, c)\, \mathbb{E}_{c_N \sim p_D}\left[\log \sigma\left(-\mathbf{v}_w^T \mathbf{v}_{c_N}\right)\right], \end{aligned} \tag{3.12}$$

onde k_I é um hiperparâmetro. Esta função objetivo é otimizada pelo algoritmo ADAM que é computacionalmente eficiente e tem um consumo de memória considerado baixo (Kingma and Ba, 2017). As restrições de não negatividade $\mathbf{v}_w, \mathbf{v}_c, \mathbf{v}_{c_N} \geq 0$ são garantidas diretamente nos vetores de palavras produzidos. A cada atualização de peso durante o processo de treinamento é verificado se algum desses pesos ficou negativo. Caso isso ocorra, esse peso negativo é substituído por zero. Por meio destas alterações, é possível treinar um modelo DIVE em um corpus grande de forma similar ao word2vec. E, assim como no word2vec, ao invés de se criar uma matriz com bilhões de pesos e fatorar ela, os vetores de palavras são aprendidos minimizando a função objetivo dada acima. Com essas alterações o algoritmo consegue gerar vetores de palavras que preservam a DIH e as restrições de não-negatividade.

Porém ainda há um problema descrito em Chang et al. (2018): quando uma palavra é muito frequente, ela terá muitas outras palavras vizinhas que não tem um significado semântico. Isso acontece principalmente quando usamos uma janela de contexto grande. Essas palavras de contexto podem causar um ruído nos vetores de palavras, isto é, eles podem não conseguir capturar a DIH de forma satisfatória. O algoritmo DIVE lida com esse problema filtrando essas palavras de contexto por meio do cálculo dado na equações seguintes, em que k_f é um hiperparâmetro. Uma palavra c será filtrada do contexto de w quando a condição dessa equação for verdadeira, isto é, quando o PMI da coocorrência dessas palavras for pequeno:

$$\log\left(\frac{P(w,c)}{P(w)\cdot P(c)}\right) < \log\left(k_f\right).$$

Considerando as definições de $P(w,c)$, $P(w)$ e $P(c)$ dadas anteriormente, e realizando algumas simplificações, temos que:

$$\log\left(\frac{\#(w,c)\cdot D}{\#w\cdot\#c}\right) < \log\left(k_f\right).$$

Em seguida, removendo os logs e movimentando alguns termos, concluímos que:

$$\#(w,c)\cdot D < k_f\cdot\#w\cdot\#c.$$

Quando essa condição é verdadeira a contagem de coocorrência na função objetivo presente na Equação 3.12 é $\#(w,c) = 0$. Esse cálculo para filtrar essas palavras usando PMI evidencia a necessidade de se criar uma contagem de coocorrência entre as palavras que ocorrem em um contexto. Esse é um dos motivos para determinarmos o tamanho do corpus construído neste trabalho (mais detalhes na Seção 4.3.1).

3.7 Datasets em Inglês

Neste trabalho, foram utilizados alguns datasets de detecção de hiperônimos em inglês. Novos datasets em português foram desenvolvidos neste trabalho, mas estes são descritos na Seção 4.2. A escolha dos datasets em inglês se deu pelo trabalho de Chang et al. (2018), em que a comparação do DIVE com outros algoritmos foi mostrada através desses datasets. Quatro datasets usados no nosso trabalho foram retirados de Shwartz et al. (2017):

- BLESS (Baroni and Lenci, 2011),
- EVALution (Santus et al., 2015),
- Lenci/Benotto (Benotto, 2015) e

- Weeds (Weeds et al., 2014).

Outros quatro datasets foram retirados do repositório *H-feature detector* (Roller and Erk, 2016):

- Medical (i.e levy2014) (Levy et al., 2014),
- LEDS (i.e baroni2012) (Baroni et al., 2012),
- TM14 (i.e turney2014) (Turney and Mohammad, 2014) e
- Kotlerman 2010 (Kotlerman et al., 2010).

Além desses oito datasets, foram utilizados os conjuntos de treino e teste do dataset HypeNet (Shwartz et al., 2016) e o conjunto de teste do dataset WordNet (Vendrov et al., 2015), todos disponíveis no repositório do DIVE (Chang et al., 2018).

Os datasets foram pré-processados pois alguns incluem etiquetas morfossintáticas (*POS tags*), por exemplo se a palavra é um verbo ou substantivo. Foi feito um pré-processamento simples para remover: rótulos de POS e pares repetidos. Na Tabela 3.5, é apresentado o número de pares em cada dataset antes e depois do pré-processamento. No geral, os datasets não foram tão afetados por esse processo.

Dataset	**Antes**	**Depois**
BLESS	26.554	26.532
Medical	12.602	12.602
EVALution	13.675	7.107
LEDS	2.770	2.770
Lenci/Benotto	5.010	5.010
TM14	2.188	2.158
Weeds	2.928	2.233
Kotlerman	2.940	2.940
HypeNet	17.670	17.670
WordNet	8.000	7.990

Tabela 3.5: Número de pares em cada em dataset antes e depois do pré-processamento.

O dataset BLESS foi criado selecionando pares de palavras e suas relações da WordNet. Em Shwartz et al. (2017) os pares que tinham a relação de hiperônimo foram categorizados como positivos e os pares que tinham outros tipo de relação, como merônimos, e pares aleatórios foram categorizados como negativos. EVALution foi criado usando pares da WordNet e ConceptNet, filtrado automaticamente e validado manualmente (crowdsourcing) para melhorar sua qualidade. Lenci/Benotto também foi construído usando bases de conhecimento, nele os pares positivos são relações de hiperônimo e os pares

negativos são relações de sinônimo e antônimo. Weeds, similar aos anteriores, os pares positivos foram retirados da WordNet e os pares negativos são outros tipos de relações.

Medical (i.e levy2014) (Levy et al., 2014) foi criado utilizando textos de origem médica e tem uma alta qualidade de anotação. Nesse dataset os pares positivos, além de ter as relações de hiperônimo, também tem relações como merônimo e sinônimo. LEDS (baroni2012) (Baroni et al., 2012), foi criado similarmente ao BLESS. Os pares hipônimo-hiperônimo foram retirados da WordNet e os pares negativos foram gerados de forma aleatória utilizando esses mesmos pares positivos. TM14 (turney2014) (Turney and Mohammad, 2014) foi feito com pares de palavras com vários tipos de relações, entre elas a relação de hiperônimo. Nesse dataset os pares positivos incluem outros tipo de relações como sinônimo, e os pares negativos são relações mais incomuns, como causa-efeito, entre outros. Kotlerman 2010 (Kotlerman et al., 2010) é um dataset anotado manualmente, similar ao TM14, ele foi feito para capturar uma noção mais ampla de inferência, entre elas a relação de hiperônimo.

HypeNet foi construído com o objetivo de treinar e validar um modelo supervisionado de detecção de hiperônimos. Foram extraídas relações de hiperônimo da WordNet e DBPedia, além de outros recursos (Auer et al., 2007). Todos os exemplos neste dataset estão presentes em pelo menos uma dessas bases semânticas. Nesse dataset há uma divisão aleatória dos exemplos entre conjuntos de treino e teste, e esses foram os conjuntos utilizados nesse trabalho. O conjunto de treino foi utilizado para seleção de modelos e o conjunto de teste foi utilizado para avaliar os melhores modelos. WordNet é um conjunto de teste em que é extraído pares da WordNet como positivos e outros pares de forma aleatória como negativos.

Na Tabela 3.6, é mostrado uma estatística descritiva de cada dataset. Podemos ver o número de pares positivos, negativos e também a porcentagem de pares positivos em relação ao total de pares do dataset.

Dataset	positivo	negativo	% positivo
BLESS	1337	25195	5.03
LEDS	1385	1385	50.00
WordNet	3994	3996	49.98
EVALution	1911	5196	26.88
Kotlerman	880	2060	29.93
HypeNet	3512	14158	19.87
Lenci/Benotto	1933	3077	38.58
Medical	945	11657	7.49
Weeds	1119	1114	50.11
TM 14	1058	1100	49.02

Tabela 3.6: Distribuição entre pares positivos e negativos em cada dataset em inglês.

CAPÍTULO

4

BHearst

Neste capítulo, é descrito o algoritmo BHearst proposto neste trabalho para detecção de hiperônimo que combina o modelo BERT com padrões de Hearst. Além disso, são descritos os datasets em português criados neste trabalho. Por fim, é descrita a implementação do algoritmo DIVE em PyTorch.

4.1 BERT e Padrões de Hearst

BERT é um modelo pré-treinado de representação de linguagem. Ele tem sido usado para treinar modelos que atingem o estado da arte em várias tarefas da área de PLN. A escolha deste modelo se pela característica de modelo de linguagem mascarado (MLM) descrito no capítulo anterior. É possível notar similaridades entre a estratégia de treinamento do MLM e os padrões de Hearst. Ambas as ideias consistem em uma frase com algumas palavras a serem preenchidas. Considere, por exemplo, o padrão de Hearst $P^{(x,y)} = (x, \text{é}, \text{um}, \text{tipo}, \text{de}, y)$ e o par $(\text{tigre}, \text{animal})$ como um candidato a possuir a relação de hiperônimo. A ideia básica por trás do *BHearst*, algoritmo proposto neste trabalho para detecção de hiperônimo, consiste em aplicar a técnica descrita na Seção 3.4 para calcular a probabilidade da sentença $P^{(\text{tigre},\text{animal})} = (\text{tigre}, \text{é}, \text{um}, \text{tipo}, \text{de}, \text{animal})$, que consiste em instanciar o padrão $P^{(x,y)}$ usando o par $(\text{tigre}, \text{animal})$. Então a probabilidade $p(P^{(\text{tigre},\text{animal})})$ pode ser usada para predizer se o par considerado é positivo ou não. Pelos princípios dos padrões de Hearst e do modelo BERT, pares positivos deveriam obter probabilidade alta, enquanto pares negativos deveriam obter probabilidade baixa. Inicialmente, nesta seção, é considerado o caso de um único padrão. Posteriormente, será apresentado uma técnica para combinar múltiplos padrões.

Lembrando que a entrada para a tarefa de detecção de hiperônimo é um conjunto de pares candidatos, como ilustrado na Tabela 4.1. Um modelo a ser avaliado nesta tarefa deve retornar uma pontuação para cada par candidato para que seja possível ordenar os pares de acordo com esta pontuação e então calcular a métrica AP.

x	y	**Classe**
terror	filme	POSITIVO
fruta	osso	NEGATIVO
vinho	bebida	POSITIVO
jóia	farelo	NEGATIVO
vegetal	planta	POSITIVO
talher	relação	NEGATIVO
tigre	animal	POSITIVO
gramínea	grupo	NEGATIVO

Tabela 4.1: Exemplo de dataset contendo oito pares com suas respectivas classes.

Como mencionado acima, o *BHearst* determina a pontuação de um dado par candidato (x, y) por meio de um padrão de Hearst $P^{(x,y)}$ instanciado com este par. Mais especificamente, este algoritmo utiliza a probabilidade $p(P^{(x,y)})$ da sentença resultante. Por questões de estabilidade numérica, e como é preciso apenas ordenar os pares candidatos, a pontuação deste algoritmo para um par (x, y) será dada por $\log p(P^{(x,y)}))$ que, considerando a Equação 3.6, é:

$$\log p(X) = \sum_{t=1}^{T} f(X_{\setminus t})[x_t] - \log Z, \tag{4.1}$$

onde $\log Z$, considerando a Equação 3.7, é dado por:

$$\log Z = \log \sum_{X'} \exp\left(\sum_{t=1}^{T} f(X'_{\setminus t})[x'_t]\right). \tag{4.2}$$

Ainda é necessário lidar com a dificuldade de calcular Z pois isto envolve um somatório sobre todas as sequências possíveis de tamanho T. A seguir, é proposto algumas estratégias para calcular uma aproximação deste valor de maneira eficiente.

4.1.1 Constante de Normalização

Primeiramente, é necessário discutir o problema de que a probabilidade $p(X)$ é definida para um tamanho T fixo. Em primeira análise, pode-se imaginar que isto não é um problema para esta abordagem, já que um padrão de Hearst é uma sequência de tamanho fixo com duas posições (x, y) a serem

preenchidas por duas palavras. Ou seja, pode-se pensar que as sequências geradas por meio do padrão x é um tipo de y sempre possuem comprimento seis. Entretanto, é preciso se atentar para o fato de que algumas palavras podem ser divididas em sub-tokens pelo tokenizador *WordPiece*. E como o padrão envolve duas palavras que serão definidas por diferentes pares candidatos, cada palavra deste par pode ser dividida em diferentes quantidades de sub-tokens. Por exemplo, as palavras tigre e animal são tokenizadas como ti-gre e a-ni-mal, respectivamente. Portanto, temos que $P^{(tigre,animal)} = (\text{ti, -gre, é, um, tipo, de, a, -ni, -mal})$, ou seja, uma sequência composta por nove tokens. Desta maneira, o cálculo do primeiro termo da Equação 4.1 para este caso envolve a aplicação do BERT às nove sequências mascaradas apresentadas na Tabela 4.2.

$X_{\setminus t}$	**Sequência Mascarada**								
$X_{\setminus 1}$	[MASK]	-gre	é	um	tipo	de	a	-ni	-mal
$X_{\setminus 2}$	ti	[MASK]	é	um	tipo	de	a	-ni	-mal
$X_{\setminus 3}$	ti	-gre	[MASK]	um	tipo	de	a	-ni	-mal
$X_{\setminus 4}$	ti	-gre	é	[MASK]	tipo	de	a	-ni	-mal
$X_{\setminus 5}$	ti	-gre	é	um	[MASK]	de	a	-ni	-mal
$X_{\setminus 6}$	ti	-gre	é	um	tipo	[MASK]	a	-ni	-mal
$X_{\setminus 7}$	ti	-gre	é	um	tipo	de	[MASK]	-ni	-mal
$X_{\setminus 8}$	ti	-gre	é	um	tipo	de	a	[MASK]	-mal
$X_{\setminus 9}$	ti	-gre	é	um	tipo	de	a	-ni	[MASK]

Tabela 4.2: Exemplo de sequências mascaradas usando o padrão x é um tipo de y para o par (tigre, anserimal). A palavra tigre é tokenizada em dois sub-tokens (ti e -gre); enquanto a palavra animal em três sub-tokens (a, -ni e -mal).

Por outro lado, temos que $P^{(vinho,bebida)} = (\text{vinho, é, um, tipo, de, be, -bida})$, ou seja, uma sequência composta por sete tokens. Considerando que o comprimento (da parte fixa) do padrão é invariável, definimos como *comprimento de um par* (x, y) o número de sub-tokens obtidos quando x e y são tokenizados pelo *WordPiece*. O par (tigre,animal), por exemplo, tem comprimento cinco, enquanto o par (vinho,bebida) tem comprimento três. Na Tabela 4.3, é apresentado estes valores para os pares da Tabela 4.2.

Para lidar com este problema, são definidos constantes de normalização específicas para cada comprimento de par L. Nos datasets, temos que $2 \leq L \leq 18$. Então, definimos a log-probabilidade para uma sentença $X = P^{(x,y)}$ gerada a partir de um par (x, y) de comprimento L como:

$$\log p(X) = \sum_{t=1}^{|X|} f(X_{\setminus t})[x_t] - \log Z_L, \tag{4.3}$$

onde $|X|$ é o comprimento da sequência X e Z_L é a constante de normalização

Comprimento (L)	x	y	**Classe**
2	terror	filme	POSITIVO
2	fruta	osso	NEGATIVO
3	vinho	be-bida	POSITIVO
3	jóia	fare-lo	NEGATIVO
4	ve-getal	plan-ta	POSITIVO
4	ta-lher	rela-ção	NEGATIVO
5	ti-gre	a-ni-mal	POSITIVO
5	gra-mí-nea	gru-po	NEGATIVO

Tabela 4.3: Exemplos de pares e seus comprimentos.

específica para pares de comprimento L. O valor de $\log Z_L$ é então definido como:

$$\log Z_L = \log \sum_{X' \,:\, |X'|=|X|} \exp\left(\sum_{t=1}^{|X'|} f(X'_{\setminus t})[x'_t]\right). \tag{4.4}$$

Observe a restrição $|X'| = |X|$ que indica que o somatório é sobre todas as sequências X' cujo comprimento é igual ao comprimento da sequência X (que é a sequência cuja log-probabilidade está sendo calculada). Observe ainda que, como estamos considerando apenas um padrão, o comprimento de X varia somente em função do comprimento do par (x, y) usado para gerar X (pois $X = P^{(x,y)}$). Isto significa que o valor de $\log Z_L$ é igual para todos os pares de comprimento L (por esta razão que definimos esta constante em função de L).

Neste momento, é importante lembrar que o objetivo final das log probabilidades $\log p(X)$ dadas aos pares é apenas ordená-los entre si. Podemos concluir então que a utilidade das constantes de normalização $\log Z_L$ é permitir uma comparação adequada entre pares com comprimentos diferentes. Pois, se todos os pares tivessem o mesmo comprimento L', consequentemente, teríamos uma única constante $\log Z_{L'}$ que seria decrementada da pontuação de todos os pares. Portanto, se ignorássemos esta constante no cálculo das pontuações, a ordem entre os pares não seria alterada.

Como já mencionado, calcular o valor exato destas constantes de normalização é inviável na prática. A proposta deste trabalho para aproximar estas constantes usa duas ideias principais. A primeira ideia vem do fato de que queremos ordenar pares de palavras (x, y) com base na log probabilidade de sentenças geradas a partir do mesmo padrão $P^{(x,y)}$. No padrão x é um tipo de y, a parte é um tipo de é fixa e apenas as palavras x e y são variáveis. Desta forma, no somatório da equação de $\log Z_L$, é considerado somente sentenças X' no formato do padrão, ou seja, é considerado apenas X' na forma $P^{(x,y)}$. Desta maneira, para calcular $\log Z_L$, nos limitaremos a gerar diferentes pares (x, y) com comprimento L e então utilizaremos a log probabilidade da sentença

$P^{(x,y)}$. Esta ideia tem um embasamento probabilístico. Estamos calculando, ao invés de $P(X)$, a probabilidade condicional $P(X \mid X = P^{(x,y)})$ para um padrão específico $P^{(x,y)}$. Isto é válido pois o objetivo é apenas ordenar os pares em relação a um padrão específico. Por outro lado, mesmo com esta otimização, como os datasets possuem pares de comprimento até 18 sub-tokens, ainda é muito custoso considerar todos os possíveis pares (x, y) com comprimento L.

Portanto, a segunda ideia consiste em usar uma amostra de pares (x, y) de comprimento L para estimar o valor de $\log Z_L$. Para justificar esta ideia, é preciso compreender que $\log Z_L$ consiste em uma operação log-sum-exp (LSE) (Blanchard et al., 2020). A função LSE para uma sequência de números $(a_1, \ldots, a_n)$ consiste em:

$$\mathrm{LSE}(a_1, \ldots, a_n) = \log \sum_{i=1}^{n} \exp(a_i). \tag{4.5}$$

A equação de $\log Z_L$ é uma LSE sobre a sequência de pontuações não normalizadas dada pelo BERT para diferentes sentenças X'. Para ver isto, basta definir cada a_i como $\sum_{t=1}^{|X'|} f(X'_{\setminus t})[x'_t]$ para cada X' usada no somatório externo da Equação 4.4. A LSE é um tipo de softmax, ou seja, uma aproximação suave para a função max. Portanto, temos que $\mathrm{LSE}(a_1, \ldots, a_n) \approx \max(a_1, \ldots, a_n)$. Isto implica que é possível ter uma boa estimativa para $\log Z_L$ se conseguirmos amostrar pares de palavras (x, y) que gerem sentenças $X' = P^{(x,y)}$ com alta probabilidade $p(X')$, pois $p(X')$ cresce junto com $\sum_{t=1}^{|X'|} f(X'_{\setminus t})[x'_t]$. Como é usado um dataset de pares, esperamos que alguns destes pares gerem sentenças altamente prováveis (pois são positivos), são usados justamente estes pares para estimar $\log Z_L$. Por exemplo, o valor $\log Z_2$ será igual ao LSE das pontuações de todos os pares de comprimento $L = 2$ no dataset em questão.

Agora, vamos discutir um problema que surge desta ideia de usar os pares do próprio dataset para estimar $\log Z_L$. Existe uma grande diferença na frequência dos pares com diferentes comprimentos L. Por exemplo, existem muitos pares com comprimento $L = 2$ enquanto há poucos pares com comprimento $L = 15$. Isto é algo comum em linguagens naturais, pois a frequência das palavras varia bastante. Por isto, nos datasets usados neste trabalho, notamos que existem poucos pares para alguns comprimentos específicos, particularmente para valores grandes de L. Nestes casos, a estimativa de $\log Z_L$ proposta aqui pode ser ruim. Para aliviar este problema, é proposto usar pares de palavras aleatórias com comprimento L, além dos pares presentes no dataset, para calcular $\log Z_L$. Mais especificamente, para cada dataset, é calculado a quantidade de pares por comprimento e é verificado o maior valor M, ou seja, M é a maior quantidade de pares do mesmo comprimento. Então, para cada comprimento de par L, será selecionado de forma aleatória pares

de palavras com comprimento L a partir de um corpus grande e esses pares serão incluídos no conjunto de pares usados para calcular $\log Z_L$, de tal forma que todos os conjuntos possuam tamanho M. Desta maneira, a quantidade de pares usados para estimar $\log Z_L$ será a mesma para todos os comprimentos L.

4.1.2 Combinando Múltiplos Padrões

Até este ponto, só foi considerado a ordenação de pares candidatos usando um único padrão. Por outro lado, como já mencionado, neste trabalho são usados diversos padrões. Para cada padrão, o *BHearst* produz uma lista de predição onde os pares candidatos (x, y) aparecem em ordem decrescente da probabilidade estimada $p(P^{(x,y)})$. Isto é exemplificado na Figura 4.1. Nesta figura, é ilustrado quatro listas de predição produzidas por quatro padrões diferentes.

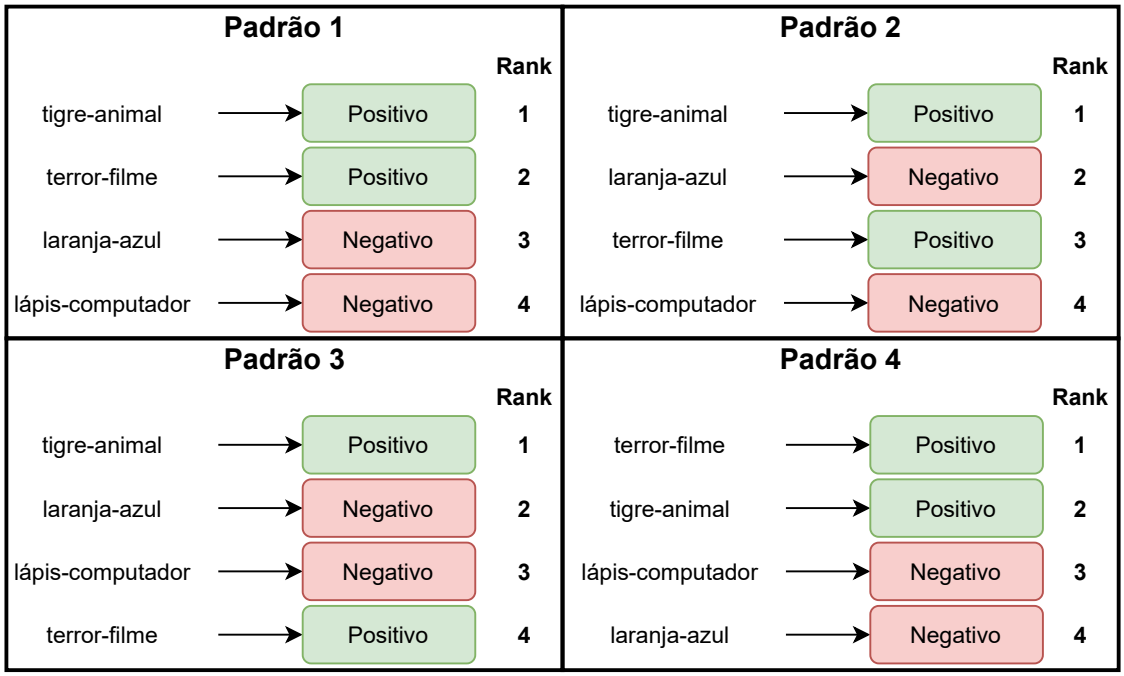


Figura 4.1: Lista de predição para cada padrão.

Dessa forma, é proposto duas abordagens para combinar os diferentes padrões com base nas posições dos pares nas listas de predição. Repare que um par pode aparecer na posição 1 na lista de um padrão e na posição 2 na lista de outro padrão. A primeira estratégia é denominada Average Rank pois usa a média das posições de um par para calcular uma nova pontuação para este par. Por exemplo, o par (tigre,animal) aparece nas posições $1, 1, 1$ e 2. Assim, o Average Rank desse par é $1,25$. A outra estratégia é denominada Min Rank, pois usa o mínimo das posições. Usando o mesmo exemplo, o Min Rank de (tigre,animal) é 1. Por fim, é criada uma nova lista de predição final ordenada

de forma crescente de acordo com essas pontuações. Essa lista é então avaliada com a métrica AP. Na Tabela 4.4, são apresentadas as pontuações destas duas estratégias para os pares do nosso exemplo.

Par Candidato	Padrão 1	Padrão 2	Padrão 3	Padrão 4	Min Rank	Average Rank
tigre, animal	1	1	1	2	1	1.25
terror, filme	2	3	4	1	1	2.50
laranja, azul	3	2	2	4	2	2.75
lápis, computador	4	4	3	3	3	3.50

Tabela 4.4: Combinando padrões usando o ranking de cada par.

4.2 Datasets em Português

Na Seção 3.7, são descritos os datasets em inglês utilizados neste trabalho. A maioria destes datasets são criados a partir de bases semânticas de palavras. A dificuldade de se criar datasets em português é que não existem muitas bases semânticas para esta língua. Pensando nisto, foram escolhidas as principais bases disponíveis em português para gerar datasets de detecção de hiperônimos. Para este trabalho, utilizamos as bases ConceptNet e Onto.PT, que foram descritas na Seção 3.5. ConceptNet é uma, das várias bases semânticas citadas, que usa dados da WordNet. Além disso nela são utilizados dados de diferentes fontes e tem suporte para o idioma português. Os dados da ConceptNet são organizados em forma de grafo. Cada aresta desse grafo é considerado uma unidade de conhecimento. Ela mostra uma relação entre dois termos de acordo com a fonte utilizada. Essa base de dados contém uma gama de relações, entre elas está a relação de hiperônimo. Algumas relações foram excluídas para evitar ruído, dado que muitas relações são similares à relação de hiperônimo. Em alguns casos, é até difícil definir se as relações são diferentes. Como a ConceptNet claramente define a relação de hiperônimo como IsA, usamos esta relação para a geração de pares positivos.

Na Tabela 4.5 são mostradas as relações utilizadas. A relação *IsA* é a relação de hiperônimo. Na Tabela 4.6 é exibida uma lista das relações ignoradas. Vale ressaltar a relação *RelatedTo*, essa é a relação mais geral dessa base semântica. Nesta categoria estão pares de palavras que podem ou não conter alguma relação e, se houver alguma relação, a ConceptNet não consegue definir qual é.

Com esses dados foi feito um pré-processamento simples, removendo palavras compostas, verificando se todos os caracteres são alfabéticos e contém, pelo menos, dois caracteres. Também foram removidos elementos repetidos. Após esses processamentos é obtido 2.323 pares de hiperônimos, conforme pode ser observado na Tabela 4.7.

Relações	Descrição
Synonym	Relação de sinônimo
UsedFor	Forma: x é usado para y
IsA	Relação de hiperônimo
EtymologicallyRelatedTo	x e y tem uma origem comum
Antonym	Relação de antônimo
NotUsedFor	Forma: x não é usado para y
SimilarTo	x é similar a y
HasSubevent	x e y são eventos e y acontece como subevento de x
ObstructedBy	x é um objetivo que pode ser evitada por y e y é um obstáculo para x
MotivatedByGoal	x é um passo para conquistar y
Causes	x e y são eventos e normalmente o evento x causa o evento y
PartOf	x é parte de y
DistinctFrom	x e y são membros distintos de um conjunto. Algo que é x não é y
Desires	x é uma entidade que normalmente quer y
EtymologicallyDerivedFrom	x é derivado de y etimologicamente
HasPrerequisite	y é uma dependência de x

Tabela 4.5: Relações presentes no ConceptNet e consideradas para este trabalho.

São utilizadas duas abordagens para criação de dois datasets a partir da ConceptNet. Primeiramente, seguindo a ideia de (Baroni et al., 2012) para criação do dataset LEDS em inglês, todos os pares com a relação IsA são utilizados como exemplos positivos. Então, é gerado a mesma quantidade de exemplos negativos selecionando aleatoriamente palavras a partir do conjunto de palavras presentes nos pares positivos. Garantindo que os pares negativos não aparecem como positivos. Este dataset é denominado ConceptNet. Na Tabela 4.8, são mostradas algumas estatísticas para este dataset. Por construção, este dataset é balanceado.

A segunda abordagem para gerar um dataset a partir da base ConceptNet é baseada nos trabalhos (Benotto, 2015; Weeds et al., 2014; Baroni and Lenci, 2011), os quais propuseram os datasets Lenci/Benotto, Weeds e BLESS, respectivamente. Os pares positivos são os mesmos da abordagem anterior. Já os pares negativos são gerados a partir de outras relações semânticas (diferentes de IsA) presentes nesta base. Mais especificamente, são usadas as relações apresentadas na Tabela 4.5. Desta forma, foi criado o ConceptNet-all, que é desbalanceado e mais difícil. A dificuldade deste dataset não está relacionada apenas ao desbalanceamento. A dificuldade vem principalmente do fato de que os pares negativos possuem relações válidas. Mesmo ignorando relações que são muito similares à relação de hiperônimo, este dataset é substancialmente mais difícil do que o dataset com pares negativos aleatórios. Na Tabela 4.9, são apresentados algumas estatísticas do dataset ConceptNet-all. Diferente do dataset ConceptNet, este dataset contém apenas $\sim 7,5\%$ dos pares

Relações	Descrição
ExternalURL	URL para a relação
MadeOf	Formato: x é feito de y
HasContext	x é uma palavra usada no contexto de y que pode ser um tópico, campo técnico ou dialeto regional
MannerOf	Similar à relação de hiperônimo e usada para verbos
DerivedFrom	x é uma palavra que aparece dentro da palavra y
HasProperty	x é uma propriedade de y
RelatedTo	Relação geral
FormOf	x é uma forma flexionada de y
AtLocation	x é uma localização de y
DefinedAs	Os significados de x e y se sobrepõem de forma considerável

Tabela 4.6: Relações presentes no ConceptNet e não consideradas para este trabalho.

Pré-processamento ConceptNet	
# hiperônimos	**# outras relações**
2.323	28.151

Tabela 4.7: Número de pares após o pré-processamento.

como exemplos positivos.

Outra base semântica utilizada para a criação de um dataset foi a *Onto.PT* (Gonçalo Oliveira and Gomes, 2014). Esta base é baseada na ideia da WordNet que usa agrupamento de palavras em synsets. Os synsets são grupos de palavras sinônimas e as relações de hiperônimo são definidas entre um par de synsets. Na Figura 4.2, são exemplificados dois synsets: A e B. Neste exemplo, o synset A é hiperônimo de B.

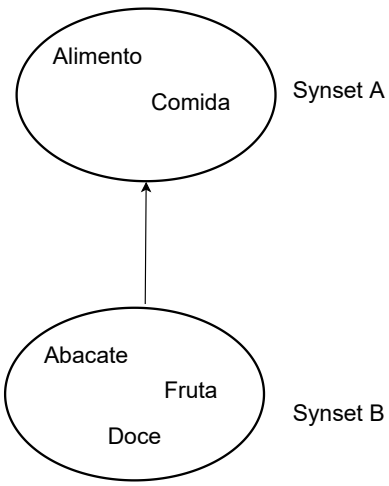


Figura 4.2: Exemplos de um par de synsets com uma relação de hiperônimo.

Com este pequeno exemplo é possível formar muitos pares de palavras que tem uma relação de hiperônimo. Na Tabela 4.10, são mostrados os pares formados usando estes dois synsets.

ConceptNet		
# positivos	**# negativos**	**# total**
2.323	2.323	4.646

Tabela 4.8: Balanceamento e total de pares.

ConceptNet-all		
# positivos	**# negativos**	**# total**
2.323	28.151	30.474

Tabela 4.9: Balanceamento e total de pares.

Pares Positivos	
abacate	alimento
abacate	comida
doce	alimento
doce	comida
fruta	alimento
fruta	comida

Tabela 4.10: Pares de palavras formadas usando o par de synsets da Figura 4.2.

Para criar um dataset com esses dados, foi usada a mesma metodologia do SemEval 2018 (Camacho-Collados et al., 2018). Primeiro são identificados quais são os synsets e suas relações de hiperônimo. Em seguida, são listadas todas as palavras desses dois synsets e um produto cartesiano é feito com essas duas listas.

Após essa primeira etapa é feito um pré-processamento igual ao procedimento dos outros datasets. Para diminuir a quantidade de pares, e também diminuir possíveis erros dessa base semântica, foi utilizado um dicionário de palavras que apareciam pelo menos 10 vezes em um corpus criado para esse trabalho (ver Seção 4.3.1). Desta forma, para se evitar um viés neste pré-processamento, foram selecionados de forma aleatória pares de palavras em que ambas estavam presentes nesse dicionário. Posteriormente, foi selecionado também de forma aleatória o mesmo número de pares em que as duas palavras não estavam neste dicionário. Estas etapas adicionais são fundamentais, pois a base semântica usada foi criada de forma automática. Apesar de ter uma etapa de validação manual, ela ainda é uma base semântica em que a confiabilidade é menor do que uma base semântica construída de forma manual. Por meio deste processo, foram obtidos 472.226 pares positivos. Os pares negativos foram gerados de forma aleatória a partir dos pares positivos, da mesma forma como foi feito para o dataset ConceptNet. Ao final, geramos 944.452 pares no total.

Em (Shwartz et al., 2016), utiliza-se uma parte do dataset HypeNet em in-

glês para seleção de modelo. Seguindo esta ideia, o dataset OntoPT foi dividido em dois subconjuntos: OntoPT-val para validação e OntoPT-teste para teste. Para realizar esta divisão, foram selecionados aleatoriamente 10% dos pares para validação e os outros 90% para teste. Na Tabela 4.11, são mostradas estatísticas destes datasets.

	# Negativos	**# Positivos**	**# Total**
OntoPT-val	47.347	47.099	94.446
OntoPT-teste	424.879	425.127	850.006

Tabela 4.11: Número de pares positivos e negativos dos datasets de validação e teste gerados a partir da base OntoPT.

4.3 DIVE em PyTorch e Word2vec

Neste trabalho, o algoritmo DIVE, proposto em (Chang et al., 2018), foi implementando utilizando a versão 3.7 da linguagem de programação *Python*. Anteriormente, este algoritmo foi disponibilizado nessa mesma linguagem porém em uma versão mais antiga, já descontinuada. Assim, esse algoritmo tem mais suporte para futuras melhorias e testes. Além disso, foi utilizado um *framework* diferente. Originalmente esse algoritmo foi escrito utilizando *TensorFlow* na versão 1.4 e nesse trabalho ele foi reescrito com o framework *PyTorch* na versão 1.2.

Uma parte importante deste trabalho é comparar o *BHearst* com as abordagens não supervisionadas DIVE e word2vec. O algoritmo DIVE detém atualmente os melhores resultados para a maioria dos datasets em inglês. O algoritmo word2vec é um método alternativo, também baseado em vetores de palavra, que não é específico para detecção de hiperônimo. Foi utilizamos a implementação do DIVE desenvolvida neste trabalho para gerar os vetores de palavra em português. Já para o inglês, foi utilizado os vetores disponibilizados por Chang et al. (2018), que foram gerados com a implementação original.

O algoritmo word2vec (Mikolov et al., 2013) aprende vetores de palavras que capturaram informações semânticas das palavras. Este algoritmo tem duas variantes e a utilizada em Chang et al. (2018) é a *skip-gram* com negative sampling. Como este trabalho seguirá a abordagem de Chang et al. (2018), foi utilizado essa mesma variante. Para aprender os vetores de palavras, o word2vec utiliza um grande corpus não anotado. Durante o treinamento, o algoritmo considera janelas de tamanho fixo no corpus. Dada uma janela, o objetivo é predizer quais palavras estão na janela dada a palavra no centro da janela. Na Figura 4.3, é exibida uma sequência da mesma sentença em que, cada vez, uma palavra alvo está em negrito. As palavras sublinhadas estão

no contexto da palavra em negrito, ou seja, na janela considerada. Assim o objetivo é, usando apenas a palavra alvo, predizer as palavras do contexto, uma de cada vez. No exemplo, a janela possui tamanho seis (três palavras à esquerda e três à direita).

a internet é um lugar de aprendizado

a **internet** é um lugar de aprendizado

a internet **é** um lugar de aprendizado

a internet é **um** lugar de aprendizado

a internet é um **lugar** de aprendizado

a internet é um lugar **de** aprendizado

a internet é um lugar de **aprendizado**

Figura 4.3: Exemplo do treinamento do word2vec para uma sentença.

Para usar o word2vec na detecção de hiperônimo, é necessário definir como é construída a lista de predição a partir de uma lista de pares candidatos. Chang et al. (2018) propõem calcular a pontuação de um par (x, y) como a similaridade de cosseno entre os vetores das palavras do par $(\mathbf{v}_x, \mathbf{v}_y)$ dados pelo word2vec:

$$P^{\text{w2v}}(x, y) = \cos(\mathbf{v}_x, \mathbf{v}_y) = \frac{\mathbf{v}_x^\top \mathbf{v}_y}{\|\mathbf{v}_x\|_2 \cdot \|\mathbf{v}_y\|_2} = \frac{\sum_{i=1}^{D} \mathbf{v}_x[i] \cdot \mathbf{v}_y[i]}{\sqrt{\sum_{i=1}^{D} \mathbf{v}_x[i]^2} \cdot \sqrt{\sum_{i=1}^{D} \mathbf{v}_y[i]^2}}. \tag{4.6}$$

Desta forma, a lista de predição é construída ordenando-se os pares de acordo com esta pontuação.

Com relação ao algoritmo DIVE, Chang et al. (2018) propõem diferentes estratégias para calcular a pontuação de um par dados os vetores de suas palavras. A estratégia com o melhor desempenho é baseada na similaridade de cosseno entre os dois vetores multiplicada por uma diferença entre os elementos dos dois vetores. Mais especificamente, a pontuação do DIVE para o par de palavras (x, y) é dada por:

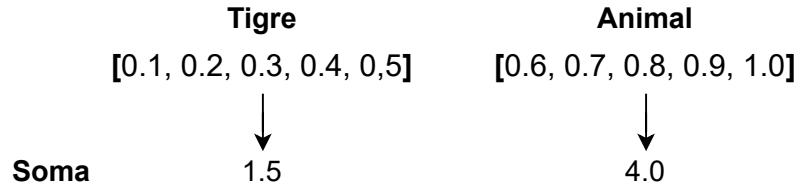
$$P^{\text{DIVE}}(x, y) = \cos(\mathbf{v}_x, \mathbf{v}_y) \cdot \Delta S(\mathbf{v}_x, \mathbf{v}_y), \tag{4.7}$$

onde $\cos(\mathbf{v}_x, \mathbf{v}_y)$ é a similaridade de cosseno entre os dois vetores e $\Delta S(\mathbf{v}_x, \mathbf{v}_y)$ é

a distância de Manhattan entre estes vetores, que é definida como:

$$\Delta S(\mathbf{v}_x, \mathbf{v}_y) = \|\mathbf{v}_y - \mathbf{v}_x\|_1 = \sum_{i=1}^{D} (\mathbf{v}_y[i] - \mathbf{v}_x[i]). \tag{4.8}$$

A intuição por trás desta função de pontuação com dois termos é a seguinte. Dado um par de palavras que possuem a relação de hiperônimo, essas palavras tendem a ser similares, ou seja, compartilham muitos contextos e, consequentemente, seus vetores devem ser parecidos. Portanto, a similaridade de cosseno entre os dois vetores deve ser grande. Ao mesmo tempo, o hiperônimo y tende a ser mais geral do que o hipônimo x e, com isto, o algoritmo DIVE tende a garantir que $\mathbf{v}_y[i] \geq \mathbf{v}_x[i]$, para $i = 1, \ldots, D$. Consequentemente, teremos que $\Delta S(\mathbf{v}_x, \mathbf{v}_y)$ tende a ser maior para pares positivos. Na Figura 4.4 é mostrado um exemplo do cálculo para pontuar o par (tigre,animal)



$$P^{\text{DIVE}}(\text{tigre,animal}) = \cos(\text{tigre, animal}) \cdot \Delta S(\text{tigre, animal})$$
$$\cos(\text{tigre, animal}) = 0.96$$
$$\Delta S(\text{tigre, animal}) = 4.0 - 1.5$$
$$P^{\text{DIVE}}(\text{tigre,animal}) = 2.4$$

Figura 4.4: Exemplo de cálculo da pontuação $P^{\text{DIVE}}(\text{tigre, animal})$.

4.3.1 Corpus para Treinamento Não Supervisionado

Uma vantagem do DIVE, assim como o word2vec, é poder ser treinado em corpus contendo texto puro. Para treinar o DIVE e word2vec em português foi utilizado textos da Wikipédia em português. Wikipédia é um site que contém textos em um amplo espectro de assuntos, ela contém textos informativos que foram construídos de forma colaborativa. É comum utilizar textos de fontes semelhantes quando se quer treinar um modelo não-supervisionado para a criação de vetores de palavra. Para coletar os dados dessa fonte foi utilizado um extrator disponibilizado publicamente (Attardi, 2015). Essa coleta foi feita em julho de 2020, foi coletado um último dump em português da Wikipédia.

Os documentos recuperados foram selecionados de maneira sequencial para compor o corpus final. Na Tabela 4.12 é mostrado o número de documentos e palavras que esse corpus contém.

Corpus Wikipédia	
# documentos	962.187
# palavras	240.000.033

Tabela 4.12: Dados do corpus criado.

A criação de um corpus foi definida pela quantidade de palavras devido à característica descrita na Seção 3.6. Foram criados corpus de 30, 60, 120 e 240 milhões de palavras. Para os experimentos realizados neste trabalho, foi considerado apenas o maior deles, contendo 240 milhões de palavras.

CAPÍTULO

5

Resultados Experimentais

Neste capítulo, são apresentados os experimentos realizados neste trabalho e os resultados obtidos são discutidos. O *BHearst* é comparado com os dois principais métodos não supervisionados presentes na literatura: DIVE e word2vec. Também são apresentados vários outros experimentos com o intuito de analisar diferentes aspectos no algoritmo proposto. E, antes de discutir todos estes experimentos, será descrito o ambiente experimental, incluindo quais modelos BERT foram usados e os valores dos hiperparâmetros para os algoritmos DIVE e word2vec. Para facilitar a leitura, as estatísticas descritivas dos datasets analisados são apresentadas no Apêndice A.1.

5.1 Ambiente Experimental

Todos os experimentos foram realizados em um servidor com as características presentes na Tabela 5.1. Todos os algoritmos utilizaram a linguagem de programação Python na versão 3.7, que é uma versão recente e estável. Junto com essa linguagem foi utilizado o *framework* PyTorch na versão 1.2 (Paszke et al., 2017).

Configurações do Servidor	
Processador	Intel(R) Xeon(R) CPU E5-1620 v4
Memória RAM	62 GB
Armazenamento HD	2.7 TB
Armazenamento SSD	229 GB
Placa de Vídeo	4 x NVIDIA GTX 1080TI 11GB

Tabela 5.1: Informações de hardware do servidor.

Para os experimentos com o BERT foram utilizados modelos disponibilizados na biblioteca Transformers da HuggingFace (Wolf et al., 2020). Para o idioma em inglês foi utilizado o modelo pré-treinado *bert-base-uncased* (Devlin et al., 2018). Para o português foi utilizado o modelo pré-treinado *BERTimbau*[1] da NeuralMind[2] (Souza et al., 2020). Ambos os modelos foram treinados utilizando corpus com bilhões de palavras. O BERTimbau foi utilizado em outras tarefas da área de PLN como, por exemplo, reconhecimento de entidades nomeadas.

Como foi realizado alguns estudos preliminares para seleção de modelos e ajuste de hiperparâmetros, se faz necessário a definição de uma metodologia de validação de modelos separada dos testes nos datasets principais. Seguindo Chang et al. (2018), será utilizado o dataset HypeNet como corpus de desenvolvimento (DEV) para o inglês. Para o português, é utilizado o dataset OntoPT-val como DEV. Os outros datasets são denominados como *testes*. Como visto anteriormente a métrica utilizada neste trabalho é AP (*Average Precision@all*).

O processo de treinamento do DIVE é similar ao treinamento de outros algoritmos de vetores de palavras. O algoritmo recebe um corpus como entrada que é pré-processado com os seguintes procedimentos:

- *Stopwords.* Remoção de palavras muito frequentes que geralmente não apresentam significado relevante para a tarefa. Exemplos de *stopwords* são artigos e pontuações;
- *Caracteres Alfanuméricos.* Somente palavras compostas exclusivamente por caracteres alfabéticos são mantidas;
- *Frequência Mínima.* São removidas as palavras com frequência menor do que um limiar K. Neste trabalho definimos $K = 10$.

Seguindo as mesmas estratégias de Chang et al. (2018), esse corpus pré-processado é divido de forma que cada linha contém o mesmo número de palavras. Em cada passo de atualização de pesos dos vetores, são selecionados 128 linhas para serem processadas juntas (*mini batch*). Outros hiperparâmetros usados neste trabalho podem ser vistos na Tabela 5.2.

É importante salientar que, para avaliar os algoritmos DIVE e word2vec nos datasets em inglês, foram utilizados os vetores de palavra disponibilizados por Chang et al. (2018)[3]. Para avaliar o DIVE nos datasets em português, foi utilizado os vetores de palavras gerados pela implementação desenvolvida neste trabalho do algoritmo DIVE treinado no corpus descrito na Seção 4.3.1.

[1] https://huggingface.co/neuralmind/bert-base-portuguese-cased
[2] https://github.com/neuralmind-ai/portuguese-bert
[3] https://github.com/iesl/Distributional-Inclusion-Vector-Embedding

Hiperparâmetros do DIVE	
Taxa de aprendizado	10^{-3}
Número de épocas	15
Tamanho da janela	20
Tamanho do vetor	100
Número de exemplos negativos	15
Frequência mínima	10
Tamanho do batch	128
Filtro PMI k_f	15
Epsilon	10^{-8}

Tabela 5.2: Hiperparâmetros utilizados no treinamento do DIVE para o idioma português.

Já para a avaliação do word2vec nos datasets em português, foi utilizada a implementação disponível na biblioteca Gensim (Řehůřek and Sojka, 2010). Os hiperparâmetros do word2vec podem ser vistos na Tabela 5.3. Para todos os outros hiperparâmetros, foram utilizados os valores padrão da biblioteca.

Hiperparâmetros do word2vec	
Número de épocas	15
Tamanho da janela	20
Frequência mínima	10
Número de exemplos negativos	15
Tamanho do vetor	100

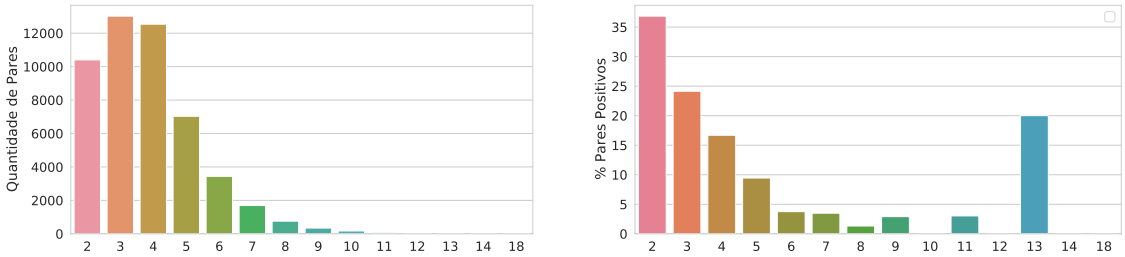
Tabela 5.3: Hiperparâmetros utilizados no treinamento do word2vec para o idioma português.

5.2 Análise dos Datasets DEV

Como visto na Seção 4.1, o comprimento de um par é definido pela soma da quantidade de sub-tokens produzidos pelo tokenizador *WordPiece* para as duas palavras do par. Na Figura 5.1 (a), é apresentado a frequência de pares por comprimento no dataset DEV em inglês. Para este mesmo dataset, na Figura 5.1 (b), é mostrado o balanceamento entre pares positivos e negativos por comprimento do par.

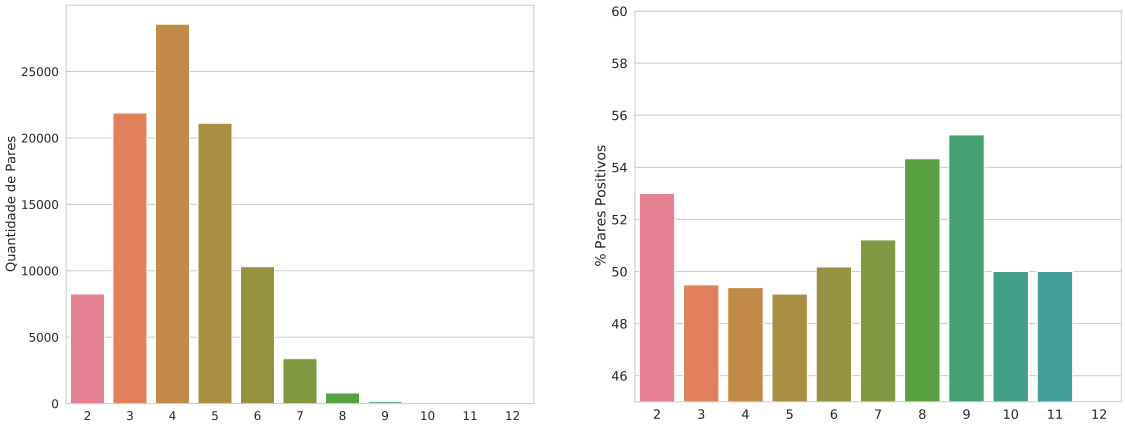
Já no dataset DEV em português, a distribuição da quantidade de pares para cada comprimento pode ser visto na Figura 5.2 (a). Na Figura 5.2 (b) é mostrado a porcentagem de pares positivos para cada comprimento.

Para os datasets de teste, tanto de inglês quanto de português, estas estatísticas são apresentadas na Seção A.1.



(a) (b)

Figura 5.1: Estatísticas de pares no dataset DEV em inglês: (a) distribuição de pares por comprimento; (b) porcentagem de pares positivos por comprimento.



(a) (b)

Figura 5.2: Estatísticas de pares no dataset DEV em português: (a) distribuição de pares por comprimento; (b) porcentagem de pares positivos por comprimento.

5.3 Combinando Múltiplos Padrões

Como apresentado na Seção 4.1, é proposto duas estratégias para combinar múltiplos padrões no algoritmo BHearst. Estas estratégias geram uma pontuação para um par candidato combinando as posições deste par em cada uma das listas de predição obtidas pelos diferentes padrões considerados. A estratégia Average Rank gera a pontuação de um par por meio da média das posições do par. Enquanto a estratégia Min Rank se baseia na posição mínima do par. Em experimentos preliminares, é notado que a utilização de muitos padrões prejudica o desempenho final do algoritmo já que os melhores padrões e a combinação deles é feita no mesmo dataset DEV.

Para selecionar quais padrões utilizar, é avaliado, nos datasets DEV de inglês e português, o desempenho obtido pelo algoritmo ao utilizar um padrão de cada vez. Nas Figuras 5.3 e 5.4, é apresentado o desempenho (AP) obtido por cada padrão nos datasets DEV de inglês e português, respectivamente.

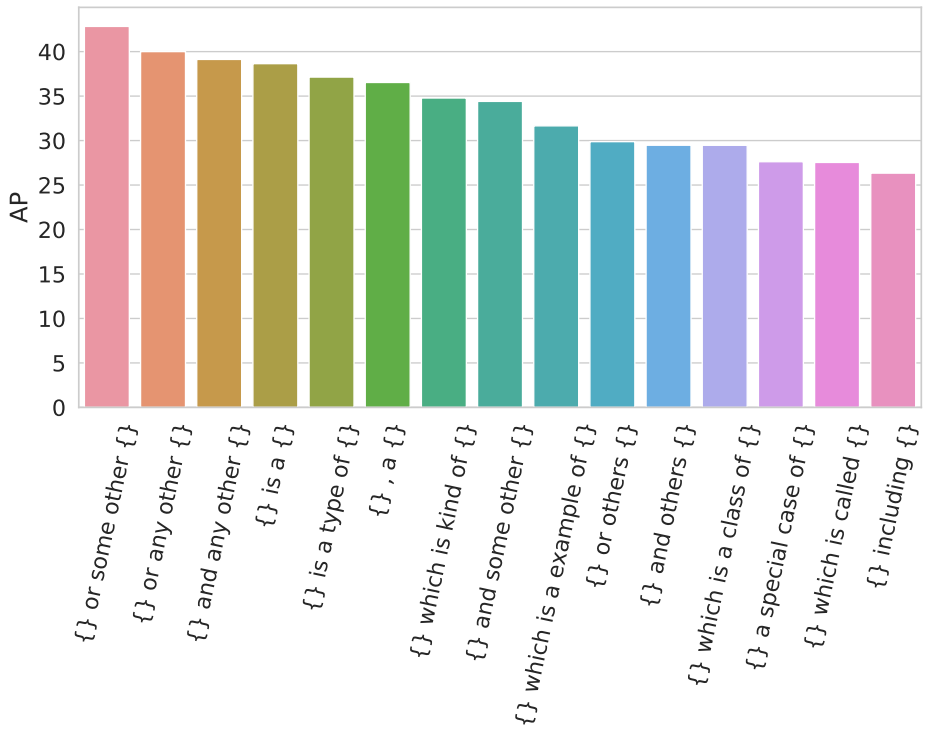


Figura 5.3: Desempenho de cada padrão no dataset DEV em inglês.

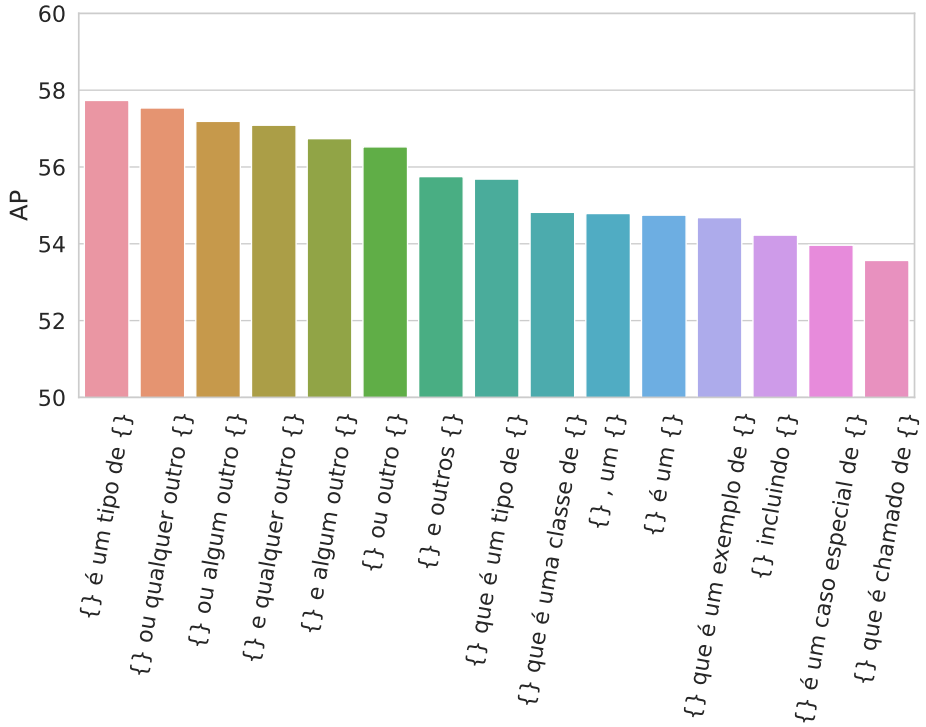


Figura 5.4: Desempenho de cada padrão no dataset DEV em português.

Em experimentos preliminares, foi descoberto que a combinação de um a quatro padrões, geralmente, resulta nos melhores desempenhos. Desta forma, foram selecionados os quatro melhores padrões para cada língua de acordo

com os resultados apresentados nas Figuras 5.3 e 5.4. Então, para as avaliações nos datasets de teste, os resultados do *BHearst* foram reportados combinando os k melhores padrões tanto para a estratégia Average Rank quanto para a Min Rank. Estes métodos foram nomeados como AR(k) e MR(k), onde k denota o número de padrões usados, e AR e MR denotam, respectivamente, Average Rank e Min Rank.

Nas Tabelas 5.4 e 5.5, são apresentados a métrica AP nos datasets de teste em inglês e português, respectivamente, para o algoritmo utilizando as estratégias AR(k) e MR(k) para $k = 1, 2, 3, 4$. Nas linhas rotuladas como *Média* e *Std* destas tabelas, são apresentados a média e o desvio padrão do desempenho em todos os datasets para cada método, respectivamente. É possível observar

Dataset	**Método**							
	AR(1)	AR(2)	AR(3)	AR(4)	MR(1)	MR(2)	MR(3)	MR(4)
BLESS	**32,87**	31,05	30,35	26,62	**32,87**	29,43	30,58	30,30
EVALution	**39,06**	38,26	38,28	37,93	**39,06**	38,01	38,27	38,25
HypeNet	**42,35**	41,99	40,86	41,20	**42,35**	40,67	39,46	36,54
Kotlerman	40,73	42,02	41,60	40,67	40,73	41,58	42,47	**42,55**
LEDS	80,69	80,26	79,60	**81,00**	80,69	79,53	79,14	80,15
LenciBenotto	48,84	48,17	48,76	**49,01**	48,84	47,93	48,49	48,66
Medical	15,68	15,33	14,63	15,81	15,68	15,30	14,94	**16,52**
TM 14	64,60	64,70	**64,80**	63,84	64,60	63,87	64,44	61,07
Weeds	60,98	61,13	61,46	61,46	60,98	61,11	61,33	**62,24**
WordNet	**53,80**	53,29	53,21	53,09	**53,80**	52,12	51,91	51,86
Média	**47,96**	47,62	47,36	47,06	**47,96**	46,96	47,10	46,81
Std	17,28	17,44	17,60	17,98	17,28	17,41	17,38	**17,23**

Tabela 5.4: Desempenho usando as estratégias Average Rank (AR) e Min Rank (MR) para combinar até os quatro melhores padrões nos datasets de teste em inglês. Os valores k em AR(k) e MR(k) indicam quantos padrões são combinados, sempre em ordem de desempenho no DEV.

Dataset	**Método**							
	AR(1)	AR(2)	AR(3)	AR(4)	MR(1)	MR(2)	MR(3)	MR(4)
OntoPT-teste	57,92	57,53	57,38	57,29	57,92	58,00	**58,15**	58,11
ConceptNet	75,61	76,74	76,64	76,35	75,61	77,88	**77,96**	77,81
ConceptNet-all	23,34	26,93	27,10	**27,81**	23,34	26,70	26,50	27,60
Média	52,29	53,73	53,70	53,81	52,29	54,19	54,20	**54,50**
Std	21,70	20,51	20,39	**19,96**	21,70	21,06	21,19	20,65

Tabela 5.5: Desempenho usando as estratégias Average Rank (AR) e Min Rank (MR) para combinar até os quatro melhores padrões nos datasets de teste em português. Os valores k em AR(k) e MR(k) indicam quantos padrões são combinados, sempre em ordem de desempenho no DEV.

que não existe uma única combinação de padrões, nem mesmo uma estraté-

gia de combinação (AR ou MR), que obtém o melhor desempenho em todos os datasets. Este resultado indica que determinados padrões funcionam melhor em alguns datasets e pior em outros. Até certo ponto, isto faz sentido, pois estes datasets envolvem domínios variados. Mas podemos observar que, no inglês, os métodos AR(1) e MR(1) obtêm o melhor desempenho médio. Observe que estes dois métodos são idênticos, pois ambos utilizam apenas o melhor padrão e, consequentemente, não há uma combinação de diferentes padrões. Já para o português, temos que o método MR(4) obtém o melhor desempenho médio. Mesmo este método não sendo o melhor método em nenhum dataset, ele sempre consegue ficar próximo do melhor desempenho Desta maneira, nas próximas seções, por simplicidade, será reportado apenas os resultados do algoritmo AR/MR(1) para os datasets em inglês e do algoritmo MR(4) para os datasets em português. Tanto para o inglês quanto para o português, estes métodos serão denotados indistintamente como *BHearst*.

5.4 Comparação com Estado da Arte

O algoritmo DIVE (Chang et al., 2018) detém atualmente os melhores resultados, dentre os métodos não supervisionados, para a maioria dos datasets em inglês na tarefa de detecção de hiperônimo. Considerando os três datasets em português que são propostos neste trabalho, ainda não existem resultados publicados. Nesta seção, são comparados os algoritmos DIVE e word2vec com o *BHearst* usando dez datasets em inglês e três em português. São reportados os resultados do método AR/MR(1) para os datasets em inglês e do método MR(4) para os datasets em português. Esses métodos são denotados aqui como *BHearst*.

Na Tabela 5.6, é comparado o desempenho do método BHearst com os métodos DIVE e word2vec (W2V) para os datasets de teste em inglês.

Podemos notar que método BHearst supera o word2vec em 9 dos 10 datasets, obtendo pior desempenho apenas no dataset WordNet. O word2vec é surpreendentemente eficaz no dataset WordNet, ficando apenas 0,42 ponto abaixo do desempenho do DIVE neste dataset. Em comparação com o algoritmo DIVE, o algoritmo BHearst o supera em metade dos datasets. Além disso, é possível notar que, na média, o algoritmo BHearst supera o algoritmo DIVE em mais de 1 ponto e o word2vec em mais de 9 pontos.

Na linha *AVG* da Tabela 5.6, é mostrado o ranking médio de cada método. E na linha rotulada como *%DifMédia*, é apresentado a média da diferença percentual entre o desempenho do método e o desempenho do melhor método no dataset. Considerando um método com desempenho d em um dataset

Dataset	Método		
	BHearst	DIVE	W2V
BLESS	**32,87**	15,52	9,11
EVALution	**39,06**	34,15	25,46
HypeNet	**42,35**	37,31	25,59
Kotlerman	**40,73**	36,59	39,50
LEDS	80,69	**83,44**	71,76
LenciBenotto	48,84	**52,86**	41,78
Medical	15,68	**19,24**	11,24
TM 14	**64,60**	56,83	51,32
Weeds	60,98	**69,75**	52,32
WordNet	53,80	**57,25**	56,83
Média	**47,96**	46,29	38,49
AVG	**1,60**	**1,60**	2,80
%DifMédia	**4,80**	9,94	27,25

Tabela 5.6: Desempenho do método BHearst comparado aos métodos DIVE e word2vec (W2V) nos datasets de teste em inglês.

específico, a diferença percentual %Dif do método é dada por:

$$\%\text{Dif} = 100 \cdot \frac{d_{\max} - d}{d_{\max}},$$

onde $d_{\max}$ é o desempenho máximo dentre todos os métodos para o dataset considerado. O valor %DifMédia é, por sua vez, a média do valor %Dif em todos os datasets. Quanto menor este valor, melhor é o desempenho médio do método, pois mais próximo do desempenho máximo. Considerando esta medida, o método BHearst fica 5,14% mais próximo do melhor método, na média de todos os datasets, do que o algoritmo DIVE.

Na Tabela 5.7, é comparado o desempenho do método BHearst com os métodos DIVE e word2vec (W2V) para os datasets de teste em português.

Dataset	Método		
	BHearst	DIVE	W2V
OntoPT-teste	58,11	57,16	**61,03**
ConceptNet	**77,81**	58,66	68,63
ConceptNet-all	**27,60**	12,83	9,16
Média	**54,50**	42,88	46,27
AVG	**1,33**	2,33	2,00
%DifMédia	**1,59**	28,15	26,20

Tabela 5.7: Desempenho do método BHearst comparado aos métodos DIVE e word2vec (W2V) nos datasets de teste em português.

Nesses resultados é possível notar que o método BHearst conseguiu supe-

rar o DIVE nos três datasets, e na média ficou melhor tanto do DIVE quanto do Word2vec. Um ponto interessante é que o método BERT conseguiu um ótimo resultado no dataset ConceptNet-all, com quase o dobro da pontuação do DIVE. Esse dataset foi construído de forma que os pares negativos são pares de palavras que de fato há algum tipo de relação semântica. Nesse mesmo dataset vemos que o DIVE mostra a sua capacidade superando o Word2vec e mesmo assim fica atrás dos métodos BHearst. Uma outra característica desse algoritmo que é perceptível nesses resultados é a capacidade com que eles lidam com palavras fora do vocabulário. O BERT, antes de processar qualquer sentença, faz uma tokenização da palavra. Essa característica permite que o modelo lide com palavras mais raras, ele divide essa palavra em subunidades para conseguir capturar o seu significado. Já o DIVE e o Word2vec usa apenas representações de palavras que viram durante o processo de treinamento não supervisionado. Durante o processo de avaliação, caso as duas palavras do par não estejam no vocabulário do DIVE esse par é movido para o final da lista de predição, assumindo que não uma relação de hiperônimo entre elas. Esse processo de mover o par para o final da lista de predição também é feita no Word2vec afim de se obter uma comparação justa entre os dois algoritmos.

5.5 Constante de Normalização

Na Seção 4.1.1, são apresentadas algumas ideias para computar uma aproximação da constante de normalização $\log Z_L$ de maneira eficiente. Como discutido naquela seção, o impacto desta constante é restrito à comparação de pares com comprimentos diferentes. Se todos os pares tivessem o mesmo comprimento, esta constante não teria nenhum efeito no BHearst. Nesta seção, é analisado o impacto desta constante no algoritmo. Na Tabela 5.8, é apresentado o desempenho do BHearst e de uma versão dele que ignora a constante $\log Z_L$ (coluna *Sem* $\log Z_L$). Podemos notar que o desempenho do algoritmo sofre uma enorme queda, em todos os datasets, quando esta constante não é considerada. Portanto, fica evidente que o impacto do comprimento do par nas pontuações não normalizadas dadas pelo BERT é substancial. Ao mesmo tempo, a solução proposta neste trabalho parece funcionar bem.

Outro problema discutido na Seção 4.1.1 é a questão dos desbalanceamento entre pares com diferentes comprimentos nos datasets. Como é usado pares com mesmo comprimento L para estimar $\log Z_L$, conjecturamos que este desbalanceamento pode ser prejudicial, principalmente para valores altos de L, para os quais existem poucos pares nos datasets. Por isso, é proposto uma ideia de gerar pares aleatoriamente para balancear estes conjuntos. Na coluna *Desbalanceado* da mesma tabela, é apresentado o desempenho do BHe-

Dataset	BHearst	Desbalanceado	Sem log Z_t
BLESS	**32,87**	32,40	11,53
EVALution	**39,06**	38,43	34,75
HypeNet	**42,35**	41,86	15,60
Kotlerman	**40,73**	40,73	30,39
LEDS	**80,69**	79,79	62,92
LenciBenotto	**48,84**	48,43	35,33
Medical	**15,68**	15,39	6,09
TM 14	**64,60**	64,13	55,15
Weeds	60,98	**61,27**	52,20
WordNet	**53,80**	51,78	19,86
Média	**47,96**	47,42	32,38

Tabela 5.8: Impacto do balanceamento de pares por comprimento para o cálculo da constante de normalização $\log Z_L$ (coluna *Desbalanceado*) e o impacto da própria constante como um todo (coluna *Sem* $\log Z_L$) no desempenho do método BHearst nos datasets em inglês.

arst quando não é incluído os pares aleatórios para balancear os conjuntos por comprimento.

Na Tabela 5.9, é apresentado uma análise de impacto das constantes de normalização para os datasets em português. Pode-se notar, mais uma vez, um efeito substancial destas constantes.

Dataset	BHearst	Sem log Z_L
OntoPT-teste	**58,11**	57,38
ConceptNet	**77,81**	71,75
ConceptNet-all	**27,60**	11,26
Média	**54,50**	46,79

Tabela 5.9: Impacto da constante de normalização $\log Z_L$ no desempenho do método BHearst nos datasets em português.

Por limitações de tempo para executar os experimentos nos datasets em português (o dataset OntoPT-teste é particularmente grande), não foi avaliado o BHearst sem o balanceamento de pares por comprimento nestes datasets.

CAPÍTULO

6

Conclusão e Trabalhos Futuros

Neste capítulo é apresentado uma conclusão sobre este trabalho junto com possíveis ideias que podem ser abordadas no futuro.

6.1 Conclusão

É possível notar que a simples ideia de combinar um modelo de linguagem mascarada com padrões da linguagem pode conseguir bons resultados, se comparado com outras abordagens não supervisionadas. Essa mesma ideia pode obter resultados até melhores do que outros métodos específicos para essa tarefa de detecção de hiperônimos. Com essa ideia simples foi obtido, em cinco datasets em inglês, o melhor AP superando um outro algoritmo não-supervisionado DIVE. E em nove dos dez datasets em inglês esse método superou o famoso algoritmo Word2vec, que apesar de não ser um algoritmo especializado nessa tarefa, obteve pontuações satisfatórias.

Neste trabalho também foram desenvolvidos três datasets de detecção de hiperônimos em português e em dois deles o método BHearst conseguiu a melhor pontuação.

Foi utilizado um modelo pré-treinado, sem a necessidade de buscar um corpus para treinamento, além disso não foi preciso calibrar nenhum hiperparâmetro. Mesmo com essas características é possível notar que os resultados são comparáveis com outros métodos que requerem todos esses passos citados. Um feito interessante deste trabalho foi como combinar pares de diferentes comprimentos de forma que seria possível compará-los. Utilizando o cálculo de mitigação $\log Z_L$ é possível obter bons resultados.

6.2 Trabalhos Futuros

Ainda é necessário explorar outras formas de como mitigar o efeito do comprimento do par e também de como combinar padrões de linguagem. Em vez de se utilizar pares aleatórios pode-se usar uma ideia de criar pares que ocorrem em uma mesma sentença em um corpus, por exemplo.

Uma outra ideia seria explorar esses padrões de linguagem e descobrir novos padrões, em vez de se utilizar adaptações de padrões em inglês para o português.

Também existe a necessidade de verificar qual é o impacto do modelo contextual pré-treinado, como escolher esses modelos e como adaptá-los para essa tarefa. Além de se utilizar outros modelos de linguagem, como por exemplo o GPT, para calcular a probabilidade de uma sentença. Neste trabalho foi explorado apenas a característica do *Masked Language Model* do BERT. Seria interessante explorar a outra característica presente neste modelo: a *Next Sentence Prediction*.

Após a publicação do artigo um dos autores de Wang and Cho (2019) publicou em um blog pessoal sobre um erro no artigo. Em uma das equações apresentadas é argumentado que ela é uma *conditional distribution of the MRF*. No entanto o autor chega a conclusão que isso é um erro. Essa equação é na verdade uma distribuição sobre n-ésimo token dado todos os outros tokens. Este erro provavelmente não afeta o nosso trabalho, no entanto seria interessante investigar melhor.

Também existe a necessidade de comparar o BHearst com outros métodos não-supervisionados para a detecção de hiperônimos, especialmente os métodos propostos em Shwartz et al. (2017) e Roller et al. (2018).

Referências Bibliográficas

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org. Citado na página 5.

Giuseppe Attardi. Wikiextractor. `https://github.com/attardi/wikiextractor`, 2015. Citado na página 44.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. Citado na página 30.

Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics, 2011. Citado nas páginas 7, 28, e 39.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, 2012. Citado nas páginas 29, 30, e 39.

Giulia Benotto. Distributional models for semantic relations: A sudy on hyponymy and antonymy. 2015. Citado nas páginas 28 e 39.

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 08 2020. ISSN 0272-4979. doi: 10.1093/imanum/draa038. URL https://doi.org/10.1093/imanum/draa038. draa038. Citado na página 36.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. Semeval-2018 task 9: Hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 712–724, 2018. Citado nas páginas 9 e 41.

Haw-Shiuan Chang, ZiYun Wang, Luke Vilnis, and Andrew McCallum. Distributional inclusion vector embedding for unsupervised hypernymy detection, 2018. Citado nas páginas 3, 12, 25, 28, 29, 42, 43, 47, e 52.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. *arXiv preprint arXiv:1606.08359*, 2016. Citado na página 2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Citado nas páginas 3, 4, 17, e 47.

Luis Espinosa-Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. Supervised distributional hypernym discovery via domain adaptation. In *Conference on Empirical Methods in Natural Language Processing; 2016 Nov 1-5; Austin, TX. Red Hook (NY): ACL; 2016. p. 424-35.* ACL (Association for Computational Linguistics), 2016. Citado na página 9.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. Multiwibi: The multilingual wikipedia bitaxonomy project. *Artificial Intelligence*, 241:66–102, 2016. Citado nas páginas 4 e 9.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1199–1209, 2014. Citado na página 8.

Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics, 2005. Citado na página 24.

Hugo Gonçalo Oliveira and Paulo Gomes. ECO and Onto.PT: A flexible approach for creating a Portuguese wordnet automatically. *Language Resources and Evaluation Journal*, 48(2):373–393, 2014. URL http://dx.doi.org/10.1007/s10579-013-9249-9. Citado nas páginas 4, 24, e 40.

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992. Citado nas páginas 2, 12, 15, e 16.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics, 2008. Citado na página 2.

Yacine Jernite, Alexander Rush, and David Sontag. A fast variational approach for learning markov random field language models. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2209–2217, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/jernite15.html. Citado na página 21.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. Citado na página 27.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359, 2010. Citado nas páginas 29 e 30.

Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001. Citado na página 26.

Omer Levy, Ido Dagan, and Jacob Goldberger. Focused entailment graphs for open ie propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, 2014. Citado nas páginas 27, 29, e 30.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015a. Citado na página 26.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings*

of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 970–976, 2015b. Citado nas páginas 2, 8, e 11.

Thiago Lima, Sandra Collovini, Ana Leal, Evandro Fonseca, Xiaoxuan Han, Siyu Huang, and Renata Vieira. Analysing semantic resources for coreference resolution. In *International Conference on Computational Processing of the Portuguese Language*, pages 284–293. Springer, 2018. Citado na página 24.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. Citado nas páginas 3, 25, e 42.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. Citado nas páginas 4, 9, e 24.

Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012. Citado nas páginas 4 e 24.

Roberto Navigli and Paola Velardi. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327. Association for Computational Linguistics, 2010. Citado na página 16.

David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008. Citado na página 13.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. Citado nas páginas 5 e 46.

Simone Paolo Ponzetto and Michael Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 192–199. Association for Computational Linguistics, 2006. Citado na página 2.

Wei Qiu, Mosha Chen, Linlin Li, and Luo Si. Nlp_hz at semeval-2018 task 9: a nearest neighbor approach. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 909–913, 2018. Citado na página 10.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018. Citado na página 17.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`. Citado na página 48.

Stephen Roller and Katrin Erk. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. *arXiv preprint arXiv:1605.05433*, 2016. Citado na página 29.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. Hearst patterns revisited: Automatic hypernym detection from large text corpora. *arXiv preprint arXiv:1806.03191*, 2018. Citado nas páginas 3, 15, 16, e 57.

Mark Sammons, VG Vydiswaran, and Dan Roth. Recognizing textual entailment. multilingual natural language applications: From theory to practice, 2011. Citado na página 1.

Ivan Sanchez and Sebastian Riedel. How well can we predict hypernyms from word embeddings? a dataset-centric analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 401–407, 2017. Citado na página 8.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, pages 64–69, 2015. Citado na página 28.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. *arXiv preprint arXiv:1603.06076*, 2016. Citado nas páginas 12, 29, e 41.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. *EACL*, 2017. Citado nas páginas 1, 8, 12, 15, 28, 29, e 57.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020. Citado na página 47.

Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451, 2017. Citado nas páginas 4, 5, e 24.

Peter D Turney and Saif M Mohammad. Experiments with three approaches to recognizing lexical entailment. *arXiv preprint arXiv:1401.8269*, 2014. Citado nas páginas 29 e 30.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. Citado na página 19.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015. Citado na página 29.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014. Citado na página 25.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. Citado na página 9.

Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019. Citado nas páginas 21, 23, e 57.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259. Dublin City University and Association for Computational Linguistics, 2014. Citado nas páginas 29 e 39.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6. Citado na página 47.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. Citado na página 18.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. Learning term embeddings for hypernymy identification. In *IJCAI*, pages 1390–1397, 2015. Citado na página 1.

APÊNDICE

A

Resultados Adicionais

Neste capítulo, apresentamos alguns resultados adicionais.

A.1 Estatísticas dos Datasets de Teste

Para os datasets de teste em inglês, na Figura A.1, apresentamos a distribuição de pares por comprimento de par. Quanto maior o comprimento, menor é a quantidade de pares. Esta tendência é vista em todos os datasets.

Na Figura A.2, mostramos a distribuição de pares por comprimento nos datasets de teste em português. A tendência mostrada nos datasets em inglês também é observada aqui.

Na Figura A.3, apresentamos a porcentagem de pares positivos por comprimento de par nos datasets de teste em inglês. Normalmente existem poucos pares com comprimento elevado. Logo, em alguns destes casos, notamos que a porcentagem de pares positivos fica próxima de zero ou próxima de 100%.

Na Figura A.4, apresentamos a porcentagem de pares positivos por comprimento de par nos datasets de teste em português.

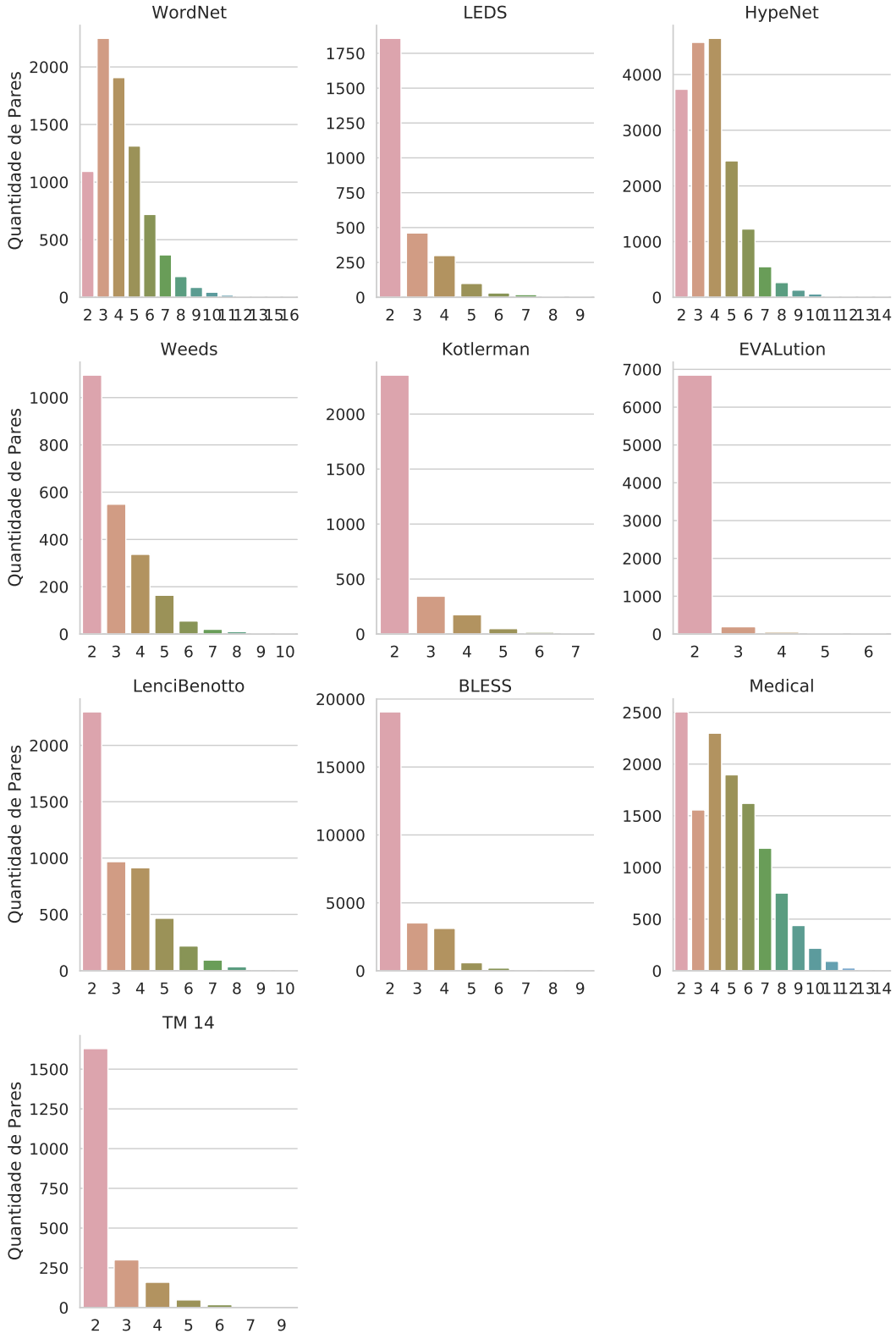


Figura A.1: No eixo X temos os comprimentos de par presentes em cada dataset em inglês e no eixo Y temos o número de pares para cada comprimento.

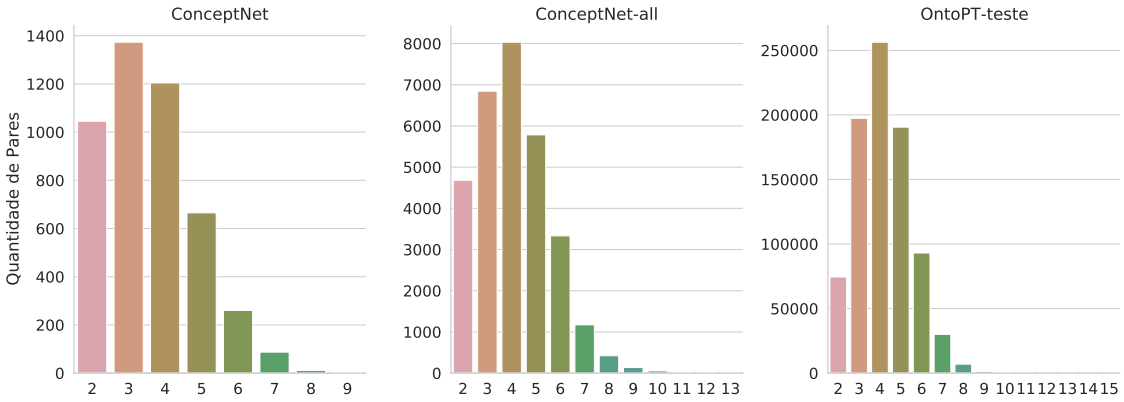


Figura A.2: Quantidade de pares para cada comprimento nos datasets de teste em português.

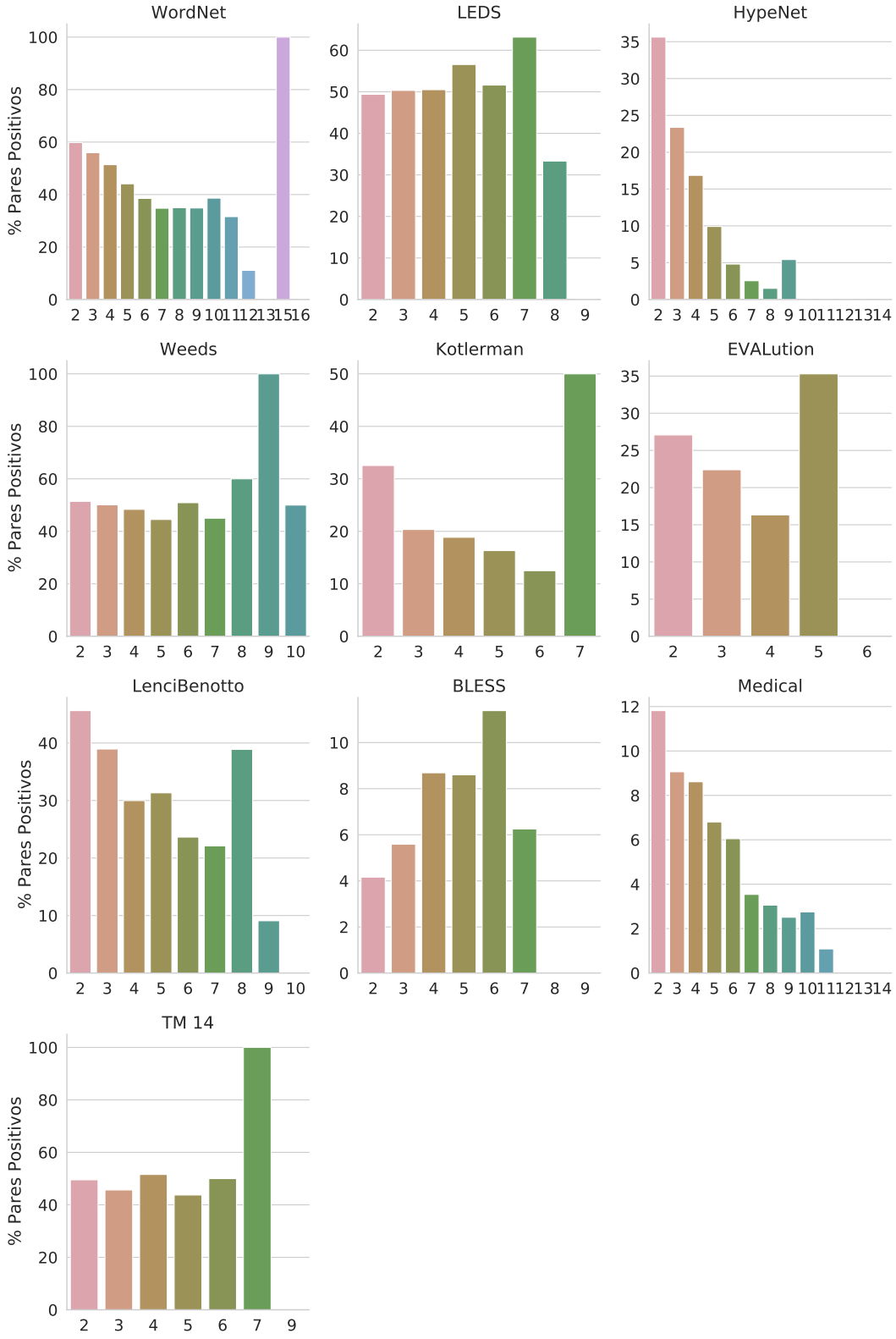


Figura A.3: Porcentagem de pares positivos para cada comprimento de par nos datasets em inglês.

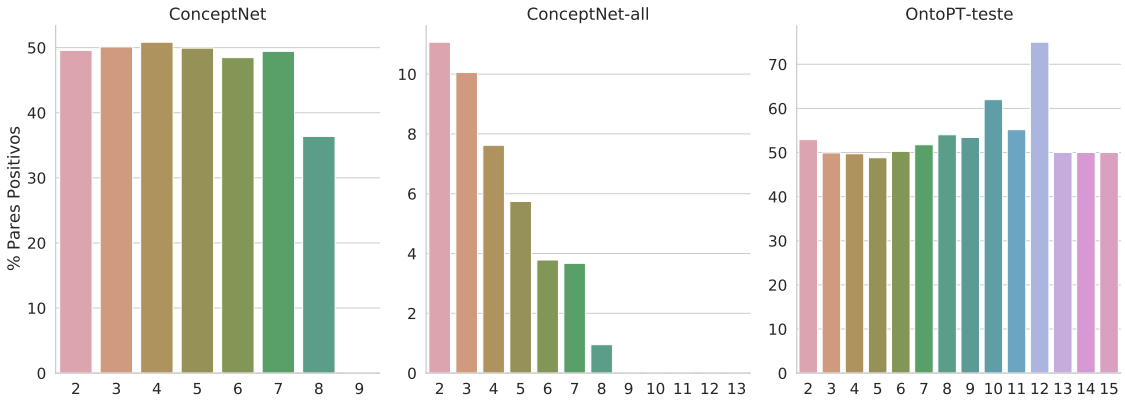


Figura A.4: Porcentagem de pares positivos para cada comprimento, usando todos os pares dos datasets.