UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sobre o uso de conhecimento especialista para auxiliar no aprendizado de *Word Embeddings*

Dissertação de Mestrado

Flávio Arthur Oliveira Santos

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Flávio Arthur Oliveira Santos

Sobre o uso de conhecimento especialista para auxiliar no aprendizado de *Word Embeddings*

Dissertação de Mestrado

Orientador(a): Dr. Hendrik Teixeira Macedo

São Cristóvão – Sergipe

2019

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Santos, Flávio Arthur Oliveira
S237s Sobre o uso de conhecimento especialista para auxiliar no aprendizado de *Word Embeddings* / Flávio Arthur Oliveira Santos ; orientadora Hendrik Teixeira Macedo. – São Cristóvão, SE, 2019.
69 f. : il.

Dissertação (mestrado em Ciências da computação) – Universidade Federal de Sergipe, 2019.

O
1. Computação. 2. Processamento de linguagem natural (Computação). 3. Conhecimento e aprendizagem - morfologia. 4. Paráfrase. II.Macedo, Hendrik Teixeira, orient. III. Título.

CDU 004.41

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato:FLÁVIO ARTHUR OLIVEIRA SANTOS

Em 31 dias do mês de Julho do ano de dois mil e dezoito, com início às 10h00min, realizou-se na Sala de Seminários do DCOMP da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Flávio Arthur Oliveira Santos**, que desenvolveu o trabalho intitulado: ***"Sobre o uso de conhecimento especialista para auxiliar no aprendizado de Word Embeddings"***, sob a orientação do Prof. Dr. Hendrik Teixeira Macedo. A Sessão foi presidida pelo Prof. Dr. Hendrik Teixeira Macedo (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. André Britto de Carvalho (PROCC/UFS) e, em seguida, ao Prof. Dr. Cleber Zanchettin (UFPE). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO "*(aprovado/reprovado)*" SEM "*(com/sem)*" ressalvas. Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 25/2014/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos*", 31 de Julho de 2018.

Prof. Dr. Hendrik Teixeira Macedo
(PROCC/UFS)
Presidente

Prof. Dr. André Britto de Carvalho
(PROCC/UFS)
Examinador Interno

Prof. Dr. Cleber Zanchettin
(UFPE)
Examinador Interno

Flávio Arthur Oliveira Santos
Candidato

UNIVERSIDADE FEDERAL DE SERGIPE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - PROCC
Sala do Seminário do DCOMP – Departamento de Computação / UFS, Av. Marechal Rondon, S/N - Jardim Rosa Elze - Tel. (79) 3194-6353. CEP: 49100-000 - São Cristóvão - Sergipe - Brasil
E-mail: secretaria.pos@dcomp.ufs.br

Observações (em caso de aprovação com ressalvas):

Prof. Dr. Hendrik Teixeira Macedo
(PROCC/UFS)
Presidente

Prof. Dr. André Britto de Carvalho
(PROCC/UFS)
Examinador Interno

Prof. Dr. Clerber Zanchettin
(UFPE)
Examinador Interno

Flávio Arthur Oliveira Santos
Candidato

I dedicate this work to all my parents, professors and friends who gave me the opportunity to learn.

Agradecimentos

Agradeço primeiramente ao meu irmão Kleber Tarcísio, por ter me apresentado a Ciência da Computação, área pela qual sou completamente apaixonado.

Aos meus pais, por terem me ensinado a verdade e os valores que norteiam meu caráter, por serem presentes e por perceberem o valor da educação na vida dos filhos.

A minha família como um todo, pois acredito que uma boa família é o primeiro passo para uma boa educação.

Ao meu orientador, professor Dr. Hendrik Teixeira Macedo, pela paciência, enorme partilha de conhecimento e principalmente pelas oportunidades de pesquisa e confiança. Sou muito grato por tudo. Muito obrigado!

A todos os meus professores pelos ensinamentos e críticas que me fizeram progredir.

A Universidade Federal de Sergipe, pois através dela conheci pessoas de nobreza incalculável.

Resumo

Representações de palavras são importantes para muitas tarefas de Processamento de Linguagem Natural (PLN). Obter boas representações é muito importante uma vez que a maioria dos métodos de aprendizado de máquina responsáveis pelas soluções dos problemas de PLN consistem de modelos matemáticos que fazem uso dessas representações numéricas capazes de incorporar as informações sintáticas e semânticas das palavras. Os chamados Word Embeddings, vetores de números reais gerados através de modelos de aprendizado de máquina, é um exemplo recente e popularizado dessa representação. GloVe e Word2Vec são modelos bastante difundidos na literatura que aprendem tais representações. Porém, ambos atribuem uma única representação vetorial para cada palavra, de forma que: (i) ignoram o conhecimento morfológico destas e (ii) representam paráfrases a nível de palavra com vetores diferentes. Não utilizar o conhecimento morfológico das palavras é considerado um problema porque este conhecimento é composto de informações muito importantes, tais como, radical, desinência de gênero e número, vogal temática e afixos. Palavras com essas características em comum devem ter representações semelhantes. As representações de paráfrases a nível de palavra devem ser semelhantes porque são palavras com escritas diferentes mas que compartilham o significado. O modelo FastText representa uma palavra como uma bag dos n-grams dos caracteres na tentativa de resolver o problema (i); assim, cada um destes n-gram é representado como um vetor de números reais e uma palavra é representada pela soma dos vetores dos seus respectivos n-grams. Entretanto, utilizar todos os n-grams possíveis dos caracteres é uma solução de força bruta, sem qualquer embasamento científico e que compromete (ou inviabiliza) a performance do treinamento dos modelos na maioria das plataformas computacionais existentes em instituições de pesquisa, por ser extremamente custoso. Além disso, alguns n-grams não apresentam qualquer relação semântica com suas respectivas palavras de referência. Para resolver este problema, este trabalho propõe o modelo Skip-Gram Morfológico. A hipótese de pesquisa levantada é a de que ao se trocar a bag dos n-grams dos caracteres pela bag de morfemas da palavra, palavras com morfemas e contextos similares também irão ser similares. Este modelo foi avaliado com 12 tarefas diferentes. Essas tarefas tem como finalidade avaliar o quanto os word embeddings aprendidos incorporam as informações sintáticas e semânticas das palavras. Os resultados obtidos mostraram que o modelo Skip-Gram Morfológico é competitivo se comparado ao FastText, sendo 40% mais rápido. Para tentar resolver o problema (ii), este trabalho propõe o método GloVe Paráfrase, onde uma base de dados de paráfrases a nível de palavra é utilizada para enriquecer o método GloVe original com esta informação e, assim, os vetores das paráfrases tornarem-se mais semelhantes. Os resultados da aplicação deste método mostraram que o GloVe Paráfrase necessita de menos épocas de treinamento para obter boas representações vetoriais.

Keywords: Processamento de Linguagem Natural, Word Embeddings, Conhecimento Morfológico, Paráfrase.

Abstract

Word representations are important for many Natural Language Processing (NLP) tasks. Obtaining good representations is essential since most machine learning methods responsible for solving NLP tasks consist of mathematical models that use these numerical representations, which are capable of incorporating syntactic and semantic information from the words. The so-called Word Embeddings, vectors of real numbers generated by machine learning models, are a recent and popular example of the aforementioned representations. GloVe and Word2Vec are widespread models in literature that learn said representations. However, both attribute a single vectorial representation for each word, so that: (i) their morphological information is ignored and (ii) paraphrases at word level are represented by different vectors. Not using morphological knowledge is considered an issue because that knowledge is composed by very important information, such as: radical, gender and number ending, vowel themed, affixes. Words sharing such features must have similar representations. Paraphrase representations at word level must be similar because they consist of words written differently that share the same meaning. The FastText model tries to solve problem (i) by representing a word as a bag of character n-grams; thus, each n-gram is represented as a vector of real numbers and a word is represented by the sum of its respective n-gram vectors. Nevertheless, using every possible character n-gram is a brute force solution, without any scientific basis, that compromises (or makes unviable) model training performance in most computing platforms available for research institutions since it is computationally costly. Besides, some n-grams do not show any semantic relation with their reference words. In order to tackle this issue, this work proposes the Morphological Skip-Gram model. The formulated research hypothesis states that exchanging the character bag of n-grams for the word bag of morpheme results in words with similar morphems and contexts having similar representations. This model was evaluated in terms of 12 different tasks. These tasks aim to evaluate how well the learned word embeddings incorporate syntactic and semantic information from the words. The obtained results show that the Morphological Skip-Gram model is competitive when compared to FastText, being 40% faster. In order to try solving problem (ii), this work proposes the GloVe Paraphrase method, where information from a paraphrase at word level dataset is used to reinforce the original GloVe method and, as a result, paraphrase vectors end up more similar. The experimental results show that GloVe Paraphrase requires less training epochs to obtain good vectorial representations.

Keywords: Natural Language Processing, Word Embeddings, Morphological Knowledge, Paraphrase.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PLN	Processamento de Linguagem Natural
NELL	*Never Ending Language Learning*
NLM	*Neural Language Model*
NTN	*Neural Tensor Network*
CNN	*Convolutional Neural Network*
RNN	*Recurrent Neural Network*
MLP	*Multilayer Perceptron*
SGM	*Skip-Gram Morfológico*
SGD	*Stochastic Gradient Descent*
REN	*Reconhecimento de Entidades Nomeadas*
LSTM	*Long-Short Term Memory*
GBDT	*Gradient Boosting Decision Tree*

Sumário

1

Introdução

Processamento de Linguagem Natural (PLN) é uma subárea da Inteligência Artificial (IA) responsável por dotar máquinas com capacidade para processar e compreender textos escritos em linguagem humana usual. A questão a respeito da possibilidade de existir um computador com capacidade para se comunicar de forma natural com seres humanos tem sido discutida e levantado questões filosóficas sobre o fato de computador poder compreender a semântica ou não. No entanto, atualmente existem máquinas que conseguem traduzir texto de uma língua A para uma língua B (WU et al., 2016), responder a perguntas em avaliações controladas de *question&answering* (XIONG; ZHONG; SOCHER, 2016), descrever imagens (LU et al., 2016), sumarizar texto (PAULUS; XIONG; SOCHER, 2017) e ainda apresentar excelentes resultados em tarefas mais simples, mas não menos importantes, como reconhecimento de entidades nomeadas (JúNIOR et al., 2016) e análise de sentimento (LAKKARAJU; SOCHER; MANNING, 2014). O modelo de Aprendizado de Máquina (BISHOP, 2006) por trás de tais soluções é chamado de Aprendizagem Profunda (*Deep Learning*). *Deep Learning* é uma subárea de Aprendizado de Máquina que, dentre outros modelos, destacam-se os inspirados no cérebro humano, as chamadas Redes Neurais Artificiais (RNA). As principais arquiteturas de RNA são Perceptron Multicamadas (MLP), Redes Neurais Convolucionais (CNN) e Redes Neurais Recorrentes (RNN).

A maioria das soluções de PLN envolvendo *Deep Learning* tem em comum as Representações vetoriais de palavras (*word embeddings*). *Word embeddings* são vetores de números reais que representam uma palavra; assim, cada palavra tem seu *word embedding*. Os *word embeddings* são fundamentais para os algoritmos de aprendizado de máquina obterem bons resultados nas tarefas de PLN. Um dos primeiros usos de *word embeddings* foi no ano de 1986, apresentado no artigo de Rumelhart, Hinton e Williams (RUMELHART, 1986). Os métodos de avaliação dos *word embeddings* geralmente utilizam a distância entre os vetores que representam as palavras ou o ângulo entre esses vetores para capturar a relação entre as duas palavras e consequentemente

avaliar a qualidade dessas representações. Em 2013, (MIKOLOV et al., 2013a) propuseram um método de avaliação baseado nas analogias entre dois pares de palavras que mostra os padrões aprendidos pelos *word embeddings*. Um exemplo de analogia é "Rei está para rainha assim como homem está para mulher". Utilizando, por exemplo, as representações de rei, rainha, homem e mulher, nós podemos observar a seguinte igualdade nas operações aritméticas envolvidas: rei - rainha = homem - mulher.

Existem três principais abordagens para aprender os *Word Embeddings*:

1. Modelos baseados em janela de contexto;
2. Modelos baseados em relações semânticas;
3. Modelos baseados em distância no grafo.

Todos os métodos presentes nas três categorias têm desvantagens em seu desenvolvimento. Alguns métodos do item (1), como o *Neural Language Model* (COLLOBERT et al., 2011) e *Word2Vec* (MIKOLOV et al., 2013b), apresentam bons resultados na tarefa de analogia entre palavras, mas durante o seu treinamento eles não lidam bem com a distribuição das coocorrências de palavras, pois o seu treinamento é realizado utilizando o contexto local de cada palavra ao invés do contexto global e também não levam em consideração as informações da estrutura interna das palavras (morfologia). O FastText (BOJANOWSKI et al., 2016), método baseado no *Word2Vec*, propõe uma forma de utilizar a informação interna das palavras baseada no uso dos n-grams dos caracteres das palavras. O método GloVe (PENNINGTON; SOCHER; MANNING, 2014b), também presente na abordagem (1), utiliza a informação do contexto global da palavra. Porém, palavras com escritas diferentes e significados iguais (Paráfrases) têm representações muito diferentes devido a estarem presentes em contextos diferentes.

Os métodos presentes nos itens (2) e (3) utilizam bases de conhecimento como WordNet(MILLER, 1995) e Freebase (BOLLACKER et al., 2008) para aprender os *word embeddings*. A principal desvantagem deles é que eles apenas utilizam uma subparte das bases de conhecimento citadas; o projeto *Never-Ending Language Learning* (NELL), por exemplo, que apresenta uma quantidade superior de relações semânticas entre palavras quando comparado ao WordNet, não é usualmente levado em consideração. O trabalho presente na categoria (3) utiliza a distância Leacock-Chodorow (BUDANITSKY; HIRST, 2001) para capturar a informação semântica entre duas palavras. A sua principal desvantagem é que existem outras distâncias a serem consideradas, mas o trabalho não realiza experimentos com elas.

O problema central abordado nesta dissertação é o aprendizado de representações vetoriais de palavras. Baseado nos problemas observados nos métodos existentes e nas informações presentes nas bases de conhecimento encontradas, nossas hipóteses de trabalho são:

- H1: Alterando a *bag* dos n-grams do FastText pela *bag* dos morfemas das palavras nós conseguiremos aprender representações vetoriais de palavras com qualidade competitiva as aprendidas pelo FastText e precisaremos de menos tempo de treinamento.
- H2: Incorporando o conhecimento de paráfrase na matriz de coocorrência do GloVe, o modelo aprenderá boas representações de palavras em menos épocas de treinamento e as paráfrases irão ter representações semelhantes.

Assim, o objetivo desta dissertação é investigar métodos para incorporar estas informações nos modelos estado da arte encontrados. Para avaliar a qualidade das representações de palavras foram utilizados três quesitos diferentes, são eles: (i) Similaridade e associação; (ii) Analogia; (iii) Categorização. Os experimentos do primeiro quesito avaliam se as representações de palavras aprendidas têm o conhecimento da similaridade semântica entre elas. O segundo quesito tem como objetivo medir a capacidade de relação lógica entre as representações vetoriais de palavras e o terceiro avalia a qualidade dos *word embeddings* como *features* de classificação.

Como resultado de pesquisa, este trabalho contribui cientificamente com dois métodos para incorporar conhecimento especialista nos *word embeddings*:

1. GloVe Paráfrase
2. Skip-Gram Morfológico

Esta dissertação está organizada em 8 capítulos. No capítulo 2 é discutido sobre o *background* teórico necessário para compreender o conteúdo desenvolvido neste trabalho. O capítulo 3 apresenta os trabalhos relacionados mais relevantes para esta pesquisa. Nos capítulos 4 e 5 nós explicamos os métodos desenvolvidos neste trabalho e os experimentos realizados, respectivamente. Nós apresentamos os resultados e discussões a respeito dos experimentos no capítulo 6 e duas aplicações utilizando *word embeddings* no capítulo 7. As conclusões do trabalho são apresentadas no capítulo 8.

2

Background Teórico

Este capítulo descreverá os conceitos e ideias necessárias para a compreensão de todo o trabalho. A seção 2.1 apresenta a área de Processamento de Linguagem Natural com o enfoque nas bases de conhecimento. A seção 2.2 descreve as bases de conhecimento mais utilizadas. Na seção 2.3 nós apresentamos algumas formas de representação vetorial de palavras e, por fim, a seção 2.4 introduz as arquiteturas de redes neurais mais utilizadas.

2.1 Processamento de Linguagem Natural

Nos primórdios da inteligência artificial, uma abordagem usual para concepção de máquinas inteligentes era a construção manual de grandes bases estruturadas de conhecimento codificada em alguma linguagem lógica formal e métodos de raciocínio automático para derivar continuamente novos fatos a partir dessa base. Com o surgimento do mundo online contemporâneo e o consequente crescimento vertiginoso de textos online escritos em linguagem natural humana, esse esforço original de confecção de bases de conhecimento foi naturalmente sendo desencorajado e substituído por abordagens que permitissem um processamento automático de todo esse texto por parte de computadores, mesmo que alguns esforços de construção manual dessas bases ainda sejam observados (BOTSTEIN et al., 2000). Esse processamento sugere, particularmente, alta capacidade de compreensão do texto de modo a permitir a confecção de aplicações de utilidade prática como, por exemplo, os sistemas de sumarização automática (HAGHIGHI; VANDERWENDE, 2009), sistemas de perguntas e respostas (QA - *Question Answering*) (FADER; ZETTLEMOYER; ETZIONI, 2014) e sistemas de extração de informação (IE - *Information Extraction*) (ETZIONI et al., 2011). A Wikipedia e a Medline (HUNTER; COHEN, 2006) são dois exemplos dentre vários de gigantescos repositórios de textos em linguagem natural humana. O primeiro consiste de conteúdo de propósito geral enquanto o segundo consiste de conteúdo fruto de descobertas científicas organizadas em forma de artigos e que man-

tém crescimento vertiginoso de conteúdo. Claramente, a sumarização automática de todo esse conteúdo e a extração relevante de informação seriam de grande ajuda para que pesquisadores pudessem assimilar mais rapidamente todo esse conteúdo, gerar novas hipóteses de pesquisa e, eventualmente, descobrir novas correlações importantes entre os resultados obtidos nesses trabalhos.

Um exemplo da grande capacidade de revelar padrões e descobrir conexões desconhecidas em grandes volumes de informação textual por parte de um sistema de compreensão automática de textos em linguagem natural humana foi recentemente demonstrado em (SPANGLER et al., 2014). No trabalho, 70 mil artigos científicos sobre a proteína p53, associada a muitos tipos de câncer, foram processados. Em poucas semanas, o sistema permitiu a identificação precisa de várias proteínas que modificam a p53. Levando-se em conta que nos últimos 30 anos os cientistas descobriram em média uma proteína por ano, fica evidente o potencial de um sistema desse tipo. Os pesquisadores da Baylor College of Medicine conseguiram então selecionar 6 destas proteínas para serem investigadas com mais profundidade quanto à capacidade de desativar a p53. O sistema de computação então denominado IBM Watson (FERRUCCI et al., 2010) foi utilizado no trabalho em questão. Watson consiste de um sistema QA, composto por uma arquitetura de processamento paralelo de altíssimo poder aliado a bases variadas de conhecimento que contemplam mais de 200 milhões de páginas de conteúdo em linguagem natural.

A Aprendizagem Profunda (*Deep Learning*) vem se estabelecendo como estado da arte do aprendizado de máquina (LECUN; BENGIO; HINTON, 2015). Apesar de sua notoriedade ter sido primariamente estabelecida em virtude dos grandes feitos em sistemas de reconhecimento automático de imagens (RUSSAKOVSKY et al., 2015) e reconhecimento automático de fala (HINTON et al., 2012), atividades ligadas ao processamento de linguagem natural também têm se beneficiado da técnica e obtidos avanços significativos (COLLOBERT et al., 2011), (JEAN et al., 2014). O termo 'profundo' refere-se à enorme quantidade de camadas intermediárias de neurônios que um modelo desse tipo costuma possuir. Cada uma das camadas amplifica aspectos bem discriminatórios da entrada fornecida e suprime variações que se mostram irrelevantes. Após longo tempo de treinamento, a máquina é capaz de generalizar um modelo de classificação que, usualmente, trata-se da generalização de uma função não-linear complexa para novos exemplos que vierem a ser apresentados.

A maioria das soluções de aplicações de PLN envolvendo aprendizagem profunda tem uma característica em comum: o uso de *word embeddings* (MIKOLOV et al., 2013a). *Word embeddings* é uma forma de representação de palavras que geralmente é utilizada na primeira camada de uma rede neural artificial. Eles são obtidos a partir de padrões sintáticos extraídos de um corpus textual. O corpus textual pode ser uma coleção de páginas da Wikipédia ou até um grande conjunto de textos de notícias.

2.2 Bases de Conhecimento

Uma base de conhecimento é um conjunto de dados ou conhecimento acumulado sobre um determinado assunto. Na literatura de PLN existem bases de conhecimento que armazenam informações sobre palavras, como por exemplo informações semânticas a respeito de hiperônimos e hipônimos de palavras, antônimos, lemmas, relação parte-de, indicando que um determinado objeto A é parte de outro objeto B, entre outras. As bases de conhecimento de PLN mais utilizadas são: WordNet (MILLER, 1995), Freebase (BOLLACKER et al., 2008), DBpedia (AUER et al., 2007) e a base do projeto Never-Ending Language Learning (NELL) (CARLSON et al., 2010).

O Freebase é uma base de conhecimento que foi desenvolvida de forma colaborativa pela comunidade. Ela consiste de um conhecimento global que permite pessoas e máquinas (agentes inteligentes) acessar informações a respeito de uma determinada entidade. Em sua totalidade, ela compõe 1,9 bilhão de triplas da forma <sujeito, predicado, objeto>.

DBpedia é uma base de conhecimento composta por dados estruturados extraídos do Wikipédia. Ela permite que *queries* sejam executadas na sua base de conhecimento para que possam ser extraídas informações dela. A DBpedia organiza a sua informação na forma de uma ontologia que foi criada manualmente baseada nas informações mais comuns dos *infobox* das páginas da Wikipédia. Atualmente ela contém 685 classes que formam uma hierarquia e são descritas por 2.975 propriedades.

O projeto NELL é composto por um agente inteligente que aprende com o passar do tempo lendo informações da web. Todo dia ele executa duas tarefas principais: (1) ler novos fatos das páginas da web e (2) confirmar esses novos fatos para adicionar a sua base de conhecimento. A sua base de conhecimento é composta de um conjunto de triplas da seguinte forma <entidade, relação, valor>, como por exemplo (George Harrison, playsInstrument, Guitar).

O WordNet é uma base de conhecimento léxica, onde nomes, verbos, advérbios e adjetivos são agrupados em conjuntos de sinônimos cognitivos chamados *synsets*. Cada *synset* expressa um conceito diferente. *Synsets* podem ser interligados de acordo com o significado da palavra e relações léxicas. O WordNet armazena algumas relações entre palavras, sendo o sinônimo a principal delas. Outras relações encontradas são: antônimos, hipônimos, hiperônimos, é-um, parte-de. A biblioteca NLTK (BIRD, 2006), presente na linguagem de programação Python, implementa uma interface para o WordNet e também apresenta um conjunto de métodos para calcular a similaridade semântica entre duas palavras baseada nas informações do WordNet. Alguns métodos implementados são: similaridade de Resnik, Leacock-Chodorow, Wu-Palmer, Jiang-Conrath, Lin Similarity e a similaridade do caminho baseada nas relações hipérnimo e hipônimo.

2.3 Representação de Palavras

Esta seção descreve algumas formas de representação vetorial de palavras mais comuns na literatura de PLN. Sendo elas o *one-hot*, matriz de coocorrência e *word embeddings*.

2.3.1 *One-hot*

O vetor *one-hot* é a forma mais simples de representar uma palavra. Supondo que tenhamos um *corpus* cujo tamanho do vocabulário é N. Para representar a palavra cujo índice é i, o vetor *one-hot* dessa palavra será:

$$oneHot(i) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.1}$$

Ou seja, a representação será um vetor de dimensão N contendo zeros, exceto a posição i que terá o valor 1. Dessa forma, podemos perceber que o vetor *one-hot* trata cada palavra como um objeto isolado e não tem relação entre elas, por exemplo, a similaridade do cosseno entre quaisquer duas palavras sempre será 0.

2.3.2 Matriz de coocorrência

Uma alternativa ao vetor *one-hot* é representar uma palavra a partir de seus vizinhos utilizando o conceito de janela (*window*) de vizinhança de tamanho t. Considere o seguinte *corpus*:

1. Eu gosto de *Deep Learning*.
2. Eu gosto de PLN.
3. Eu gosto de dormir.

A partir desse corpus, nós formaríamos a seguinte matriz M de coocorrência baseada na janela de vizinhança de tamanho 1:

palavra	Eu	gosto	de	*deep*	*learning*	PLN	dormir	.
eu	0	3	0	0	0	0	0	0
gosto	3	0	3	0	0	0	0	0
de	0	3	0	1	0	1	1	0
deep	0	0	1	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
PLN	0	0	1	0	0	0	0	1
dormir	0	0	1	0	0	0	0	1
.	0	0	0	0	0	1	1	0

Tabela 1 – Matriz *M* de coocorrência.

Essa forma de representação de palavras parte do princípio que uma palavra pode ser representada a partir das palavras no seu contexto.

2.3.3 *Word Embeddings*

Word embeddings é uma forma de representar palavras através de vetores de valores reais. A busca por um bom modelo para aprender os *word embeddings* tem um longo histórico na literatura. (HINTON; MCCLELLAND; RUMELHART, 1986), (JURGENS et al., 2012), (MAAS et al., 2011). Alguns trabalhos como (COLLOBERT et al., 2011), (COLLOBERT; WESTON, 2008), (TURIAN; RATINOV; BENGIO, 2010) mostram que modelos de aprendizagem profunda obtém melhores resultados quando utilizamos *word embeddings* como forma de representar as palavras de entrada do modelo. Assim, obter bons *word embeddings* se tornou fundamental para aplicações de PLN. O word2vec (MIKOLOV et al., 2013a) é uma das formas mais comum utilizada para aprender os *word embeddings*, ele é composto por dois modelos neurais de uma única camada intermediária, *Continuous Bag-of-Word* (CBOW) e Skip-gram. A figura 1 apresenta uma visualização gráfica de ambas as arquiteturas.

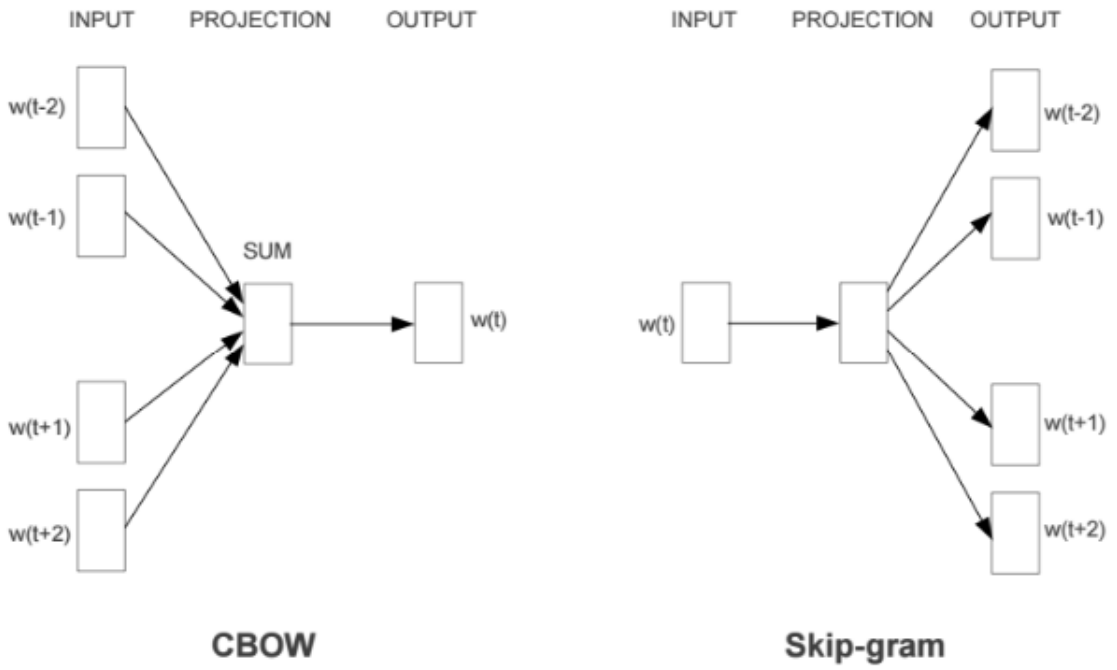

Figura 1 – Arquiteturas do modelo word2vec. Retirado de (MIKOLOV et al., 2013a).

O modelo CBOW aprende as representações de palavras prevendo a palavra central de acordo com as suas palavras do contexto dentro de uma sentença. Na arquitetura da figura 1 é utilizado as duas palavras anteriores ($w(t-2)$ e $w(t-1)$) e posteriores ($w(t+1)$ e $w(t+2)$) para prever a palavra $w(t)$. Considerando a sentença *eu gosto de deep learning* e um contexto de tamanho 2, nós utilizaríamos os vetores das palavras eu, gosto, *deep*, *learning* para prever a palavra central de. A previsão da palavra central é modelada como um problema de regressão logística, assim, prevendo que a palavra $w(t)$ está presente no centro das palavras do contexto. Antes de começar o treinamento, os vetores de cada palavra são inicializados aleatoriamente e o seu aprendizado segue através do método Adagrad (DUCHI; HAZAN; SINGER, 2011).

Devido aos *Word Embeddings* serem o foco central dessa dissertação, o capítulo 3 foi destinado apenas para discutirmos outros métodos para aprendizado de *word embeddings* de formar mais detalhada.

2.4 *N-Grams*

N-Grams (JURAFSKY; MARTIN, 2009) é uma sequência contígua de n itens retirados de um texto dado. O tamanho dessa sequência varia de 1 até o valor n escolhido pelo usuário, onde que sequência de tamanho 1, 2, e 3 são chamadas unigram, bigram e trigram, respectivamente.

Considerando a sentença "Eu gosto de Deep Learning", nós podemos extrair os seguintes conjuntos:

$$Unigram = \{Eu, gosto, de, deep, learning\} \tag{2.2}$$

$$Bigram = \{(Eu, gosto), (gosto, de), (de, deep), (deep, learning)\} \tag{2.3}$$

Uma outra abordagem comum, é extrair os conjuntos n-grams dos caracteres de uma palavra, chamados de Char N-gram. Por exemplo, se extraírmos o conjunto trigram dos caracteres das palavra gosto, nós obeteremos:

$$Trigram = \{gos, ost, sto\} \tag{2.4}$$

2.5 Aprendizado Profundo

Esta seção tem como objetivo apresentar uma introdução às arquiteturas mais utilizadas das redes neurais artificiais, que são: Perceptron Multicamadas, Redes Neurais Convolucionais e Redes Neurais Recorrentes (RNN). Nós utilizaremos essas arquiteturas no Capítulo 7 para aplicar os modelos de *word embeddings* desenvolvidos neste trabalho aos problemas de Reconhecimento de Entidades Nomeadas e Detecção de Discurso de Ódios.

2.5.1 Perceptron Multicamadas

Perceptron Multicamadas (do inglês *Multilayer Perceptron* (MLP)), também chamados redes neurais *feedfoward*, é a arquitetura principal do aprendizado profundo. O objetivo dela é aproximar uma função f^*. Supondo que f^* seja uma classificador de vinhos, dado um vetor x representando um vinho, f^* irá mapear x na sua categoria de vinho y, ou seja, $y = f^*(x)$. Dessa forma, o perceptron multicamadas $f(x;\theta)$ irá aprender os parâmetros θ para aproximar o máximo possível da função f^*. Uma MLP tem três tipos de camadas: camada de entrada, camada intermediária/oculta e camada de saída, sendo que ela pode apresentar mais de uma camada intermediária/oculta. A figura 2 apresenta uma versão gráfica da arquitetura de uma MLP com 3 camadas.

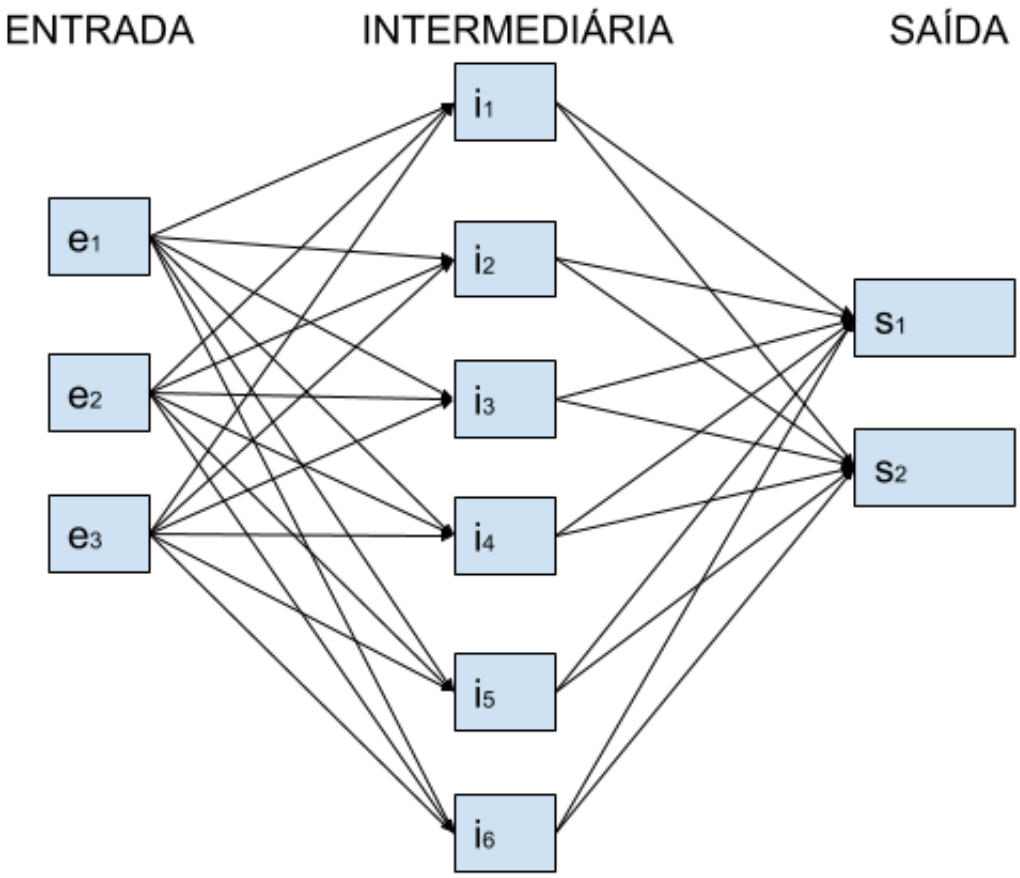

Figura 2 – Arquitetura de uma MLP com 3 camadas.

A MLP presente na figura 2 é composta de 3 neurônios na camada de entrada, 6 neurônios na camada intermediária e 2 neurônios na camada de saída. A partir da representação gráfica presente na figura 2, podemos ver que cada neurônio i_x da camada intermediária recebe informação dos três neurônios da camada de entrada, assim como cada neurônio s_z da camda de saída recebe informação de todos os neurônios da camada intermediária.

2.5.2 Redes Neurais Convolucionais

Redes neurais convolucionais (do inglês *Convolutional Neural Networks* (CNN)) (SERMANET; CHINTALA; LECUN, 2012) é uma arquitetura específica de redes neurais artificiais que processam dados no formato de *grids*, como por exemplo imagens 2D. A sua característica fundamental é ter uma camada de convolução, ou seja, ao invés de multiplicar uma determinada entrada por um matriz de pesos, a camada convolucional tem operações de convolução discreta em suas unidades e cada unidade é responsável por aplicar a convolução em um fragmento da entrada. Outra camada importante presente nas CNNs é a camada de *Pooling* (SCHERER; MÜLLER; BEHNKE, 2010), que tem como principal objetivo tornar a rede neural invariante a pequenas translações nas entradas dos dados, por exemplo, em uma rede de reconhecimento

facial não importa em qual posição da imagem está o olho, e sim que naquela imagem tem um olho presente. Um dos tipos da camada de *Pooling* é o *Max Pooling*, que retorna o valor máximo em uma determinada vizinhança do valor de entrada.

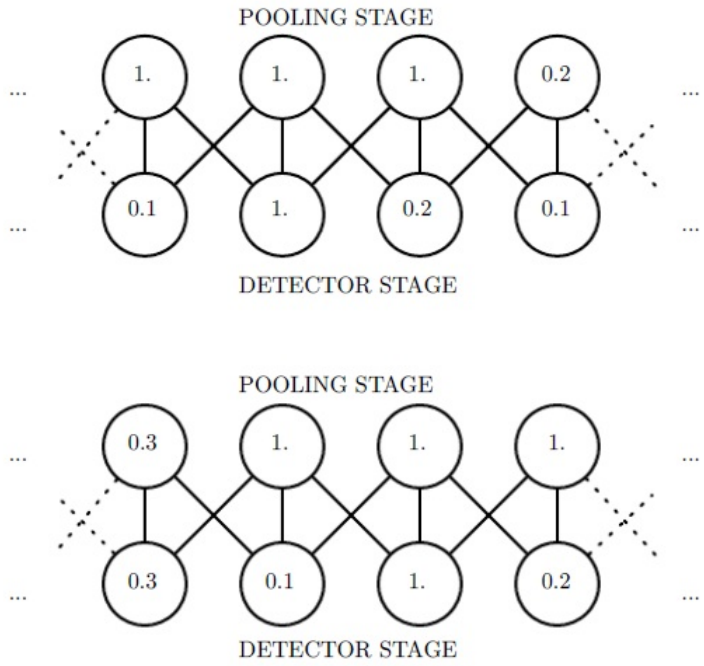

Figura 3 – Exemplo de uma operação de *Max-Pooling*. Retirado de (GOODFELLOW; BENGIO; COURVILLE, 2016).

A figura 3 apresenta um exemplo de *Max Pooling*, podemos observar que o valor 0.3 do *pooling stage* é referente ao maior valor entre 0.3 e 0.1 do *detector stage*, assim como 1.0 é referente ao maior valor entre 0.3, 0.1, 1.0, e assim sucessivamente. Esta camada é chamada de *Max Pooling* devido ao seus valores de saída serem o maior valor entre um conjunto de valores de entrada.

As CNNs tem três características importantes, são elas: *sparse interactions*, *parameter sharing* e *equivariant*.

***Sparse interactions*:** Como cada *kernel* de uma camada convolucional é responsável por aplicar a operação de convolução em uma determinada região do *grid* de entrada, as conexões da camada de entrada para a camada convolucional não são todos para todos, e sim N unidades de entrada para cada *kernel* da camada convolucional.

***Parameter sharing*:** Cada camada convolucional é responsável por extrair uma determinada informação do *grid* de entrada, logo, todas as unidades de uma camada convolucional compartilham os mesmos parâmetros.

***Equivariant*:** A posição da imagem não deve influenciar em uma CNN, assim, elas são equivariantes.

A figura 4 apresenta uma forma gráfica de uma sequência de operações de uma CNN, onde inicialmente nós aplicamos a operação de convolução e a função de ativação ReLU (NAIR; HINTON, 2010), em seguida, utilizamos a função *Max Pooling*. Uma das formas de treinar estes modelos é utilizando a descida do gradiente estocástica (do inglês *Stochastic Gradient Descent* (SGD) (BOTTOU, 2010)) para atualizar os pesos presentes no seu *kernel*.

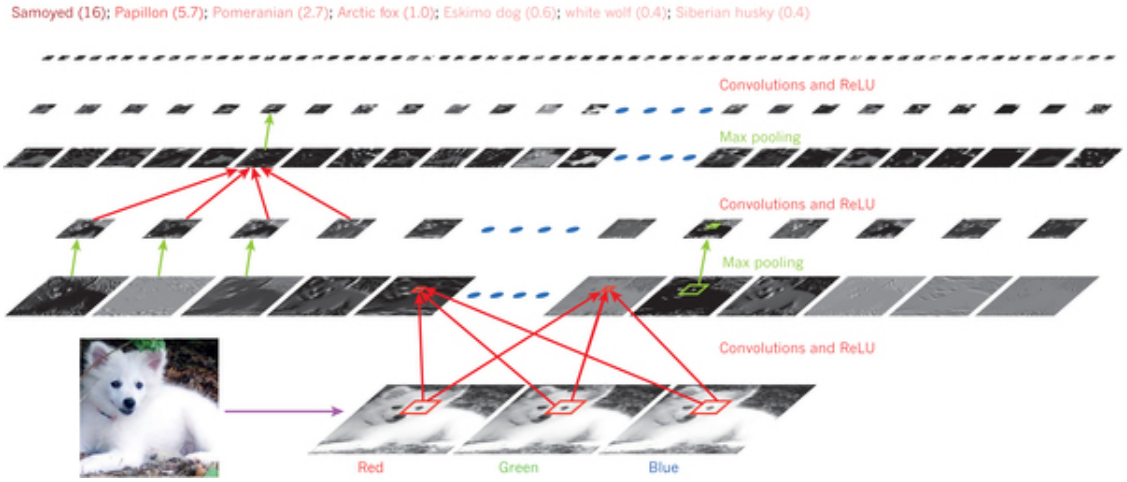

Figura 4 – Arquitetura de uma CNN.

Assim como a MLP, a CNN também é uma arquitetura do tipo *feedforward*, porque não existe recorrência em suas camadas. A figura 4 apresenta uma sequência de operações presentes em uma CNN. Inicialmente nós temos a imagem de um cachorro representada em por três matrizes, a primeira matriz contém os valores de intensidade do canal vermelho de cada pixel, a segunda do canal verde e a terceira do canal azul. Em seguida, é executado a convolução em cada uma dessas matrizes e calculado a função ReLU no resultado da convolução. Na etapa de convolução, são utilizados 3 kernels diferentes, um para cada matriz de cada canal e em então nós combinamos os resultados da convolução para uma mesma posição (x, y) das três matrizes de entrada. Após executar as etapas de convolução e ReLU, é calculado a função *Max Pooling*.

2.5.3 Redes Neurais Recorrentes

Redes Neurais Recorrentes (do inglês *Recurrent Neural Networks* (RNN)) é considerada a melhor arquitetura de redes neurais para tratar de problemas cujos dados são sequênciais (GOODFELLOW; BENGIO; COURVILLE, 2016). Uma das suas principais características é o compartilhamento de parâmetros entre diferentes tempos do modelo. Essa característica é fundamental para um problema cujos dados são sequênciais, pois se nós separarmos os parâmetros para cada tempo t do modelo a rede perde o poder de generalização dos dados

sequênciais. A figura 5 apresenta a arquitetura de uma *Vanilla* RNN. No lado direito da figura 5 nós podemos perceber que as matrizes de pesos W,U,V são compartilhadas durante todo o tempo, essa característica de compartilhamento de pesos é a responsável pela capacidade de memória da RNN.

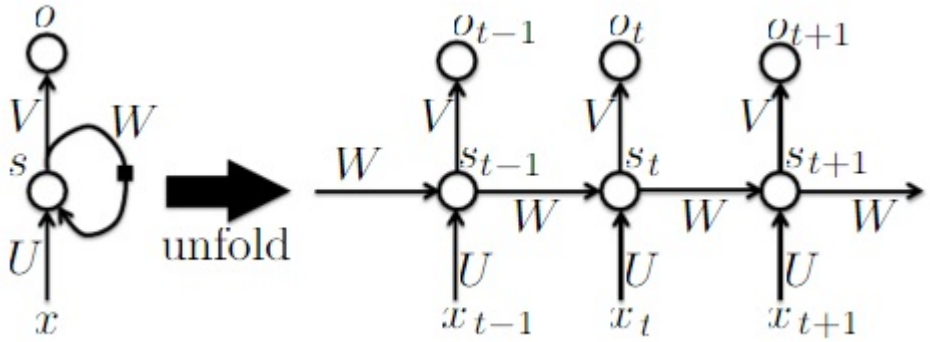

Figura 5 – Arquitetura de uma *Vanilla* RNN. Retirado de (GOODFELLOW; BENGIO; COURVILLE, 2016).

A arquitetura *Vanilla* contém três tipos de camadas: camada de entrada, camada recorrente e camada de saída. Uma das formas de treinar uma RNN é usando uma variação do algoritmo *backpropagation*, o *Backpropagation through time* (BPTT). A arquitetura de RNN mais utilizada é a *Long-Short Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997), porque ela resolve o problema do esquecimento do gradiente presente na arquitetura *Vanilla*. Discutidas as principais arquiteturas de redes neurais utilizadas, a seção 2.5 apresenta as principais métricas de avaliação que serão utilizas nos experimentos deste trabalho.

2.6 Métricas de Avaliação

Todas as métricas de avaliação utilizadas neste trabalho foram escolhidas por serem padrão dos *benchmarks* presentes no capítulo 5 ou por serem utilizadas pelos modelos *baselines* do capítulo 7.

Considerando que **TP** é a quantidade de exemplos positivos corretamente classificados, **FP** é quantidade de exemplos negativos classificadas como positivos e **FN** representa a quantidade de exemplos positivos classificados como negativos, as métricas propostas para avaliar os métodos apresentados nesse trabalho são as seguintes (ALPAYDIN, 2014):

2.6.1 Acurácia

Defininda como a razão representada pela seguinte fórmula:

$$Acuracia = \frac{TP + FP}{TP + FP + TN + FN} \tag{2.5}$$

A acurácia mede o desempenho geral do modelo, porém através dela não é possível determinar aspectos específicos desse desempenho como a habilidade de classificar corretamente os exemplos positivos ou os exemplos negativos. Existem métricas especializadas que melhor definem a qualidade do modelo com relação a tais características, sendo elas a precisão, cobertura e medida-F, que serão apresentadas nas subseções a seguir.

2.6.2 Precisão

A precisão é definida como:

$$Precisao = \frac{TP}{FP + TP} \tag{2.6}$$

A precisão, intuitivamente, é uma métrica que mede a capacidade do modelo de não classificar exemplos negativos como positivos. Quanto maior o seu valor, maior a quantidade de exemplos corretamente classificados como positivos. Ela não necessariamente mede quantos exemplos do total de positivos foram classificados corretamente.

2.6.3 Cobertura

A cobertura é definida como:

$$Cobertura = \frac{TP}{TP + FN} \tag{2.7}$$

Em outras palavras, a cobertura mede a eficiência do modelo em "encontrar"todos os exemplos positivos presentes no *dataset*.

2.6.4 Medida-F

A equação 2.8 apresenta a fórmula de calcular a medida-F através da precisão e cobertura. Ela consiste da média harmônica da precisão e cobertura. Esta medida é aproximadamente a média de ambas quando seus valores estão próximos.

$$Medida\text{-}F = 2 * \frac{Precisao * Cobertura}{Precisao + Cobertura} \tag{2.8}$$

Um modelo de alta cobertura e baixa precisão tem a capacidade de classificar muitos exemplos como positivos, porém poucos deles serão de fato positivos, ou seja, serão Falsos

Positivos (FP). Já um modelo com baixa cobertura e alta precisão, é capaz de classificar poucos exemplos como positivos, porém há uma alta probabilidade dos rótulos positivos estarem corretos. Em outras palavras, o modelo é bom em encontrar Verdadeiros Positivos (TP). Altas taxas de precisão e cobertura resultam na maioria dos exemplos positivos sendo classificados corretamente.

2.6.5 Correlação de *Spearman* (ρ)

A correlação de *Spearman* tem como objetivo calcular o grau de associação entre duas variáveis, podendo os valores delas estarem na mesma escala ou não. Por exemplo, supomos que desejamos calcular a correlação de *Spearman* entre as notas de português e história de um aluno.

Unidades	Português	História	*Rank* (Português)	*Rank* (História)	Distância
Unidade1	7.0	3.0	3	1	2
Unidade2	8.0	9.0	1	2	1
Unidade3	6.0	9.0	2	3	1

Tabela 2 – Notas do Aluno.

Após construir a tabela das notas e criarmos o *rank* das notas por unidade, é necessário calcular a distância entre as notas das duas variáveis (história e português) de acordo com as unidades, por exemplo, na primeira unidade a nota de história foi a primeira do *rank* e a nota de português a terceira, então a distância entre elas é 2. Em seguida, é necessário aplicarmos a equação 2.9 para obtermos o valor da correlação.

$$\rho = 1 - 6\sum \frac{d^2}{n(n^2-1)} \tag{2.9}$$

O valor ρ pode variar de -1 a $+1$. Quanto mais próximo de 1, indica uma forte associação entre as variáveis. Caso o valor de ρ seja 0, implica que as variáveis não tem associação. Na situação em que o valor ρ apresente resultado negativo, significa que as variáveis tem uma associação contrária.

3

Trabalhos Relacionados

Este capítulo tem como objetivo apresentar os trabalhos relacionados mais importantes para o nosso trabalho. Nós agrupamos eles em três categorias: (i) Métodos baseados em janela de contexto; (ii) Métodos baseados em relações semânticas; (iii) Métodos baseados em distância no grafo. Esta categorização é necessária devido a alguns modelos terem características em comum no processo de aprendizagem.

3.1 Métodos baseados em janela de contexto

Todos os métodos presentes nessa categoria utilizam um corpus de texto para obter as informações da janela de contexto das palavras. Exemplos de corpus de texto são: conteúdo textual de páginas da wikipédia, conteúdo textual de notícias, textos presentes em romances e livros, conjunto de *tweets*, entre outros. Após obtermos o corpus textual, um pré-processamento comum entre estes modelos é selecionar o vocabulário do corpus e designar um identificador para cada palavra. Em seguida, cada palavra é identificada apenas por seu identificador. Por exemplo, considere o seguinte corpus:

1. Eu gosto de *Deep Learning*.
2. Eu gosto de PLN.
3. Eu gosto de dormir.

O vocabulário desse corpus é o conjunto V = $\{Eu, gosto, de, Deep, Learning, PLN, .\}$ e os identificadores de cada palavra são:

- Eu = 1

- gosto = 2
- de = 3
- Deep = 4
- Learning = 5
- PLN = 6
- . = 7

Após obtermos os identificadores de cada palavra do vocabulário, nós inicializamos aleatoriamente um vetor $v \, \varepsilon \, R^n$ para cada palavra e prosseguimos o treinamento destes vetores de acordo com os métodos que serão explicados adiante.

Uma abordagem para aprender representações de palavras é através da janela de contexto de uma palavra dentro de uma sentença. Por exemplo, COLLOBERT et al. implementou um *Neural Language Model* (NLM) onde cada palavra i do vocabulário está relacionada com um vetor $v_i \, \varepsilon \, R^n$ de dimensão n, o *word embedding* de i. Uma sentença $s = (s_1, s_2, s_3, ..., s_l)$ de tamanho l é representada por um vetor x que é igual ao vetor resultante da concatenação dos *word embeddings* das palavras de s. Matematicamente, o vetor x é: $x = [v_{s_1} ; v_{s_2} ; ... ; v_{s_l}]$, onde $x \, \varepsilon \, R^{ln}$. Após obter x, ele é propagado em uma rede neural de duas camadas para obter um *score* indicando o quanto real é esta sentença s:

$$Score(x) = u^T(\sigma(Ax + b)) \tag{3.1}$$

A é uma matriz de pesos tal que $A \varepsilon R^{h \times ln}$ e $b \varepsilon R^h$ é o bias da primeira camada. O parâmetro h indica quantas unidades existem na camada f. $u^T \varepsilon R^{1 \times h}$ é o vetor de pesos da camada de saída. As matrizes de pesos e os *word embeddings* desse modelo são treinados usando *noise contrastive estimation* (nce) (GUTMANN; HYVÄRINEN, 2010), no qual, para cada sequência de treino s, é criada uma sequência com ruído s_c. Para introduzir o ruído em s_c nós escolhemos uma palavra da sequência s e trocamos ela por uma palavra do vocabulário selecionada aleatoriamente. Assim, nós temos um vetor x para s e um vetor x_c para s_c. Para treinar a rede neural para que ela obtenha um *score* alto nas sequências verdadeiras, é preciso minimizar a seguinte função de erro:

$$erro = max(0, 1 - Score(x) + Score(x_c)) \tag{3.2}$$

Os *word embeddings* **v** e os parâmetros **A**, **b**, e **u** são treinados com o *backpropagation* usando a descida do gradiente estocástica (SGD) sob um *corpus* de treino.

MIKOLOV et al. apresentam duas arquiteturas para aprender *word embeddings* baseado no contexto de uma palavra dentro de uma sentença: skip-gram e bag-of-words (CBOW). O

objetivo do skip-gram é: dado uma sentença s e a palavra central c de s, prever as palavras do contexto de c. Já o CBOW tem como objetivo prever a palavra central c baseado no contexto dela. Dado uma sequência de palavas $w_1, w_2, w_3, \cdots, w_T$, o objetivo do Skip-gram é maximizar a seguinte função:

$$E_{sk} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\geq c, j\neq 0} \mathbf{log} p(w_{t+j}|w_t) \tag{3.3}$$

Na qual c é o tamanho da janela de contexto utilizada. A fórmula mais simples do Skip-gram define $p(w_{t+j}|w_t)$ como:

$$p(w_{t+j}|w_t) = \frac{\mathbf{exp}({v'_{w_{t+1}}}^T v_{w_t})}{\sum_{n=1}^{N} \mathbf{exp}({v'_{w_n}}^T v_{w_t})} \tag{3.4}$$

Uma outra alternativa para aprender representações de palavras é através do método GloVe (PENNINGTON; SOCHER; MANNING, 2014b). Seja X uma matriz de coocorrência entre palavras, onde X_{ij} indica quantas vezes a palavra j estava presente no contexto da palavra i e $X_i = \sum_k X_{ik}$. O objetivo do GloVe é minimizar a seguinte função de custo:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{3.5}$$

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x \leq x_{max} \\ 1 & \text{if } otherwise \end{cases} \tag{3.6}$$

Sendo V o tamanho do vocabulário, w_i o *word embedding* da palavra central i e $\tilde{w}_j$ o *word embedding* da palavra de contexto j. Assim, no final nós teremos duas matrizes de *word embeddings*, W e $\tilde{W}$. A função $f(x)$ tem como objetivo ponderar a diferença entre o produto internos dos *word embeddings* e o *log* da sua frequência de coocorrência (X_{ij}).

Uma nova abordagem baseada no modelo Skip-Gram é o trabalho presente em (BOJANOWSKI et al., 2016). Eles propõem o FastText como um novo método para aprender representações de palavras. Neste método, cada palavra é representada por um conjunto de n-grams dos caracteres. Todo n-gram está associado com uma representação vetorial. Em seguida, cada palavra é representada a partir da soma de todos os vetores dos seus n-grams.

Os trabalhos apresentados até então não levam em consideração o conhecimento morfológico das palavras. Já o trabalho apresentado em (LUONG; SOCHER; MANNING, 2013) incorpora explicitamente o conhecimento morfológico nos *word embeddings*. Nele, cada palavra i é representada através seu vetor v_i e pelos vetores de seus morfemas. Após obtido as novas representações de cada palavra, ele utiliza o modelo NLM (COLLOBERT et al., 2011) para treinar os *word embeddings*.

3.2 Métodos baseados em relações semânticas

Existem bases de conhecimento na literatura que apresentam informações semânticas sobre palavras, por exemplo Freebase (BOLLACKER et al., 2008), WordNet (MILLER, 1995) Dbpedia (AUER et al., 2007), NELL (CARLSON et al., 2010). Esta informação está no seguinte formato: $t = (w_i, R, w_j)$, na qual R indica uma relação semântica entre as palavras w_i e w_j.

Alguns modelos tentam aprender as representações de palavras através dessas informações semânticas. Eles têm como entrada a tupla t e a saída é um *score* indicado o quão real é a relação R entre as palavras w_i e w_j.

O modelo TransE (BORDES et al., 2013) representa uma relação R como uma translação. Se a relação R entre as palavras w_i e w_j é verdadeira, w_i e w_j devem ser próximos após transladar w_i por R. Assim, matematicamente a sua função *score* é:

$$S_{TransE}(v_i, R, v_j) = -||v_i + R - v_j||^2 \tag{3.7}$$

Onde v_i e v_j são os *word embeddings* das palavras w_i e w_j, respectivamente.

O modelo *Neural Tensor Network* (NTN) proposto por SOCHER et al. também modela as relações semânticas entre duas palavras. Ele é uma rede neural de duas camadas com h unidades e um tensor bilinear que relaciona as duas palavras diretamente. A sua função *score* está presente na equação 3.8.

$$S_{NTN}(v_i, R, v_j) = \mathbf{U^T} f(v_i^T \mathbf{W_T} v_j + V_R \begin{bmatrix} v_i \\ v_j \end{bmatrix} + b_R) \tag{3.8}$$

Onde f é a função *sigmoid* logística, $U \varepsilon R^h$ é o vetor da camada de saída, $W_R \varepsilon R^{d \times d \times h}$, $V_R \varepsilon R^{h \times 2d}$ e $b_R \varepsilon R^h$ são o tensor bilinear, matriz de pesos e o vetor bias da relação R, respectivamente.

Assim como no NLM, o TransE e o NTN são treinados usando *noise constrative estimation* (GUTMANN; HYVÄRINEN, 2010) e descida do gradiente estocástica (SGD) (BOTTOU, 2010) usando a função de erro definida em 3.2, trocando apenas $Score(x)$ por $S_{TransE}(v_i, R, v_j)$ e $S_{NTN}(v_i, R, v_j)$.

YU; DREDZE apresentam o modelo RCM. O objetivo dele é bastante semelhante ao do word2vec (MIKOLOV et al., 2013b), a única mudança é que ao invés de considerar as palavras do contexto dentro de uma sentença como vizinho, ele considera todas as palavras que tem alguma relação semântica com a palavra atual. Dessa forma, o seu objetivo é maximizar a função

3.9.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w \varepsilon R_{w_i}} \mathbf{log}(p(w|w_i)) \tag{3.9}$$

Onde R_{w_i} é o conjunto de todas as palavras que tenha alguma relação semântica com w e $p(w|w_i)$ é estimado de acordo com a equação 3.4. WANG; MA apresentam uma versão modificada do RCM, a arquitetura do modelo continua a mesma, entretanto, eles adicionam funções para ajustar a taxa de aprendizado do modelo de acordo com a frequência que as palavras ocorrem no *corpus* de treinamento. Eles argumentam que essa modificação é necessária porque *word embeddings* de palavras com baixa frequência podem não ter diversidade de contexto e assim não terem informação suficiente para capturar o significado da palavra.

XU et al. apresentaram o *framework* RC-NET cujo objetivo é incorporar conhecimento nos *word embeddings*. Ele é composto por três modelos, C-NET, R-NET e RC-NET. Dado um conjunto S contendo triplas da forma $t = (w_i, r, w_j)$, onde que w_i e w_j pertencem ao vocabulário W e r é uma relação semântica entre elas. O princípio do R-NET é que se a relação $t = (w_i, r, w_j)$ se mantém, então a distância entre a representação de w_j e a representação $w_i + r$ deve ser próxima. Matematicamente, o objetivo do R-NET é minimizar a função de erro E_r.

$$E_r = \sum_{(w_i,r,w_j)\, \varepsilon S} \sum_{(w_i^{'},r,w_j^{'})\, \varepsilon S^{'}_{(w_i,r,w_j)}} [\gamma + d(w_i + r, w_j) - d(w_i^{'} + r, w_j^{'})]_+ \tag{3.10}$$

Onde $[x]_+$ representa a parte positiva de x, γ é parâmetro, e $d(x,y)$ é a distância euclidiana entre os vetores x e y. $S^{'}_{(w_i,r,w_j)}$ é definido da seguinte forma:

$$S^{'}_{(w_i,r,w_j)} = \{(w_i^{'}, r, w_j) | \, w_i^{'} \varepsilon W\} \cup \{(w_i, r, w_j^{'}) | \, w_j^{'} \varepsilon W\} \tag{3.11}$$

Onde $S^{'}_{(w_i,r,w_j)}$ é o conjunto de tuplas ruídos usado para treinar no NCE. Ele é obtido a partir do conjunto S de forma que $S^{'}_{(w_i,r,w_j)} \cap S = \emptyset$. Os autores do RC-NET também propõem treinar a C-NET em conjunto com o Skip-gram, assim o objetivo dessa combinação é minimizar a seguinte função de erro:

$$E_{r-join} = E_r - E_{sk} \tag{3.12}$$

O modelo R-NET é responsável por codificar atributos de palavras, fazendo com que palavras com os mesmos atributos estejam próximas. Eles definem uma função s que retorna o quanto duas palavras são semelhantes de acordo com seus atributos, de forma que dado uma

palavra w_i:

$$\sum_{j=1}^{V} s(w_i, w_j) = 1 \tag{3.13}$$

Onde que V é o conjunto de palavras do vocabulário. Para codificar os atributos, o C-NET minimiza a seguinte função de erro:

$$E_c = \sum_{i=1}^{V} \sum_{j=1}^{V} s(w_i, w_j) d(w_i, w_j) \tag{3.14}$$

Onde d(x, y) é a distância euclidiana entre os vetores x e y. Já o modelo RC-NET é a combinação do C-NET, R-NET e o modelo Skip-gram, assim o seu objetivo é minimizar a seguinte função de erro:

$$J_{rc} = E_c + E_r - E_{sk} \tag{3.15}$$

3.3 Métodos baseados em distância no grafo

O WordNet (MILLER, 1995) é uma base de conhecimento que incorpora taxonomias contendo a informação da relação *is-a* (hiperônimos/hipônimos) entre palavras. Ela permite que nós possamos consultar a similaridade semântica entre duas palavras baseada nas informações dessa taxonomia. Por exemplo, a distância Leacock-Chodorow (LCH) (BUDANITSKY; HIRST, 2001) indica o quanto duas palavras são similares baseado no menor caminho entre elas na taxonomia e na profundidade máxima da taxonomia. A distância LCH é calculada da seguinte forma:

$$LCH(w_i, w_j) = -log(\frac{p}{2d}) \tag{3.16}$$

Onde p é a menor distância entre w_i e w_j na taxonomia, e d é a profundidade máxima da taxonomia.

FRIED; DUH proporam o modelo *Graph Distance* (GD). O objetivo do modelo GD é treinar os *word embeddings* de forma que a similaridade entre eles seja igual a distância LCH entre as suas respectivas palavras no WordNet. Matemáticamente, o seu objetivo é minimizar a função 3.17.

$$L_{GD}(v_i, v_j) = (\frac{v_i v_j}{||v_i||^2 ||v_j||^2} - [a \times LCH(w_i, w_j) + b])^2 \tag{3.17}$$

Onde v_i e v_j são os *word embeddings* das palavras w_i e w_j, respectivamente. **a** e **b** são parâmetros para deixar a distância LCH na mesma escala que a similaridade do cosseno entre v_i e v_j

4
Método

Neste capítulo nós discutiremos sobre Paráfrase e Análise morfológica, que são as fontes de informações utilizadas em nossos modelos. Também explicaremos detalhadamente os métodos desenvolvidos neste trabalho, são eles: (i) GloVe Paráfrase e (ii) Skip-Gram Morfológico (SGM). Além disso, discorremos minuciosamente sobre o método FastText, que é o *baseline* de comparação do SGM.

4.1 GloVe Paráfrase

Paráfrase é a tarefa de reescrever uma sentença p utilizando palavras diferentes, mas mantendo o mesmo significado de p . Paráfrase a nível de palavra é quando reescrevemos uma palavra w com caracteres diferentes mas mantemos o significado de w . GANITKEVITCH; Van Durme; CALLISON-BURCH apresenta uma base de dados para Paráfrase, chamada *The Paraphrase Database* (PPDB). Esta base de dados contém aproximadamente 73 milhões de paráfrases a nível de sentença e 8 milhões a nível de palavra. Ela é dividida em seis tamanhos: S, M, L, XL, XXL, XXXL, em ordem crescente. As paráfrases presentes na subparte S, menor parte, tem maior precisão. Neste trabalho, nós selecionamos a subparte S do PPDB devido a suas paráfrases serem de melhor qualidade. Após selecionarmos apenas as paráfrases cujas palavras estavam presentes nosso vocabulário de treino, obtivemos por volta de 473 mil paráfrases a nível de palavra. Nós também implementamos o método $getparaphrase(word = w)$, cujo objetivo é selecionar aleatoriamente uma paráfrase da base S.

Neste modelo, nós utilizamos o método GloVe original para treinar os *word embeddings*. Ele é um método baseado em janela de contexto e utiliza a informação global do contexto das palavras contida no *corpus* de treinamento. Durante o treino, ele utiliza uma matriz de coocorrência X, onde X_{ij} indica quantas vezes a palavra j está presente no contexto de i no *corpus* de treinamento.

input : matrix X, int V

for $i \leftarrow 1$ **to** V **do**
    $p = getparaphrase(i)$;
    for $j \leftarrow 1$ **to** V **do**
        if $X_{ij} == 0 and X_{pj} != 0$ **then**
            $X_{ij} := X_{pj}$
        end
    end
end

Algorithm 1: Algoritmo utilizado para adicionar informação a matriz X.

Nós utilizamos o algoritmo 1 para adicionar informação a nossa matriz de coocorrência X e em seguida treinamos os *word embeddings* usando o GloVe. Para cada palavra v do vocabulário, o algoritmo 1 seleciona aleatoriamente uma paráfrase x e utiliza o contexto dela para preencher posições vazias do v na matriz X. Esta ideia é válida porque as palavras x e v têm o mesmo significado, então a palavra v pode ser utilizada nos mesmos contextos que a palavra x.

4.2 Análise Morfológica

Análise morfológica é a tarefa de encontrar os morfemas de uma palavra. Morfema é a menor unidade portadora de significado de uma palavra. Os morfemas obtidos pela análise morfológica estão classificados em:

- Radical/base: parte comum a um determinado conjunto de palavras, onde a partir dele outras palavras serão formadas.
- Desinência de gênero e número: tem a função de indicar se a palavra está no masculino ou feminino, plural ou singular.
- Vogal temática: liga o radical às desinências (elementos terminais indicativos das flexões das palavras) formadores das palavras, constituindo o tema.
- Afixos (prefixos e sufixos): prefixos são as partículas que estão localizadas antes do radical e sufixos aparecem depois.

Nós utilizamos a biblioteca Polyglot para realizar a analise morfológica nos modelos deste trabalho. Esta biblioteca é composta por modelos treinados do Morfessor 2.0 (SMIT et al., 2014), biblioteca composta por métodos probabilísticos cujo objetivo principal é definir os morfemas de uma palavra. Um exemplo de análise morfológica realizada pela Polyglot é: dado a palavra federativas, ela retorna o conjunto de tokens morfemas = $\{federa, tiva, s\}$ como saída.

4.3 FastText - Modelo *Baseline*

Na subseção 3.1 nós apresentamos uma breve descrição do FastText, mas nesta seção nós explicaremos detalhadamente este modelo devido ao mesmo ser o *baseline* do nosso trabalho.

Usando um vetor diferente para cada palavra, o modelo Skip-Gram ignora a informação da estrutura interna das palavras. O modelo FastText propõe uma função **score** diferente, com o objetivo de aproveitar esta informação.

Cada palavra *w* é representada como uma *bag* de caracteres n-gram. Inicialmente são adicionados os caracteres especiais $<$ e $>$ ao início e fim das palavras, permitindo ao modelo diferenciar entre sequências de caracteres que começam e terminam uma palavra. Também é adicionado a palavra completa w na *bag* de n-grams, permitindo ao modelo aprender as representações vetoriais das palavras e dos n-grams do seus caracteres. Tomando como exemplo a palavra *federativas* e usando os conjuntos 4-grams e 5-grams dos caracteres, nós obtemos:

<fede, eder, dera, erat, rati, ativ, tiva, ivas, feder, edera, derat, erati, rativ, ativa, tivas>

e a sequência especial:

<federativas>

Na prática, o modelo FastText utiliza os n-grams dos caracteres de tamanhos 3, 4, 5 e 6.

Supondo que nós tenhamos um dicionário G de caracteres n-gram de tamanho K. Dado uma palavra w, denotamos $G_w \subset G$ como o conjunto de n-gram dos caracteres que aparecem em w. Para cada n-gram contido em G, é associado uma representação vetorial z_g. Uma palavra w é representada pela soma das representações vetoriais de todos os seus z_g, assim obtemos a seguinte função *score*:

$$s(w,c) = \sum_{z_g \in G_w} z_g^T v_c \tag{4.1}$$

Reescrevendo a função objetivo do Skip-Gram utilizando a função *score* do FastText, nós obtemos:

$$E_{fasttext} = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \mathbf{log}(p(w_{t+j}|w_t)) \tag{4.2}$$

Onde $\mathbf{log} p(w_{t+j}|w_t)$ é calculado através de:

$$\mathbf{log}(p(w_{t+j}|w_t)) = log(\sigma(s(w_{t+j}, w_t))) + \sum_{i=1}^{k} E_{k_t \sim P_n(w)}[log(\sigma(-s(w_i, w_t)))] \tag{4.3}$$

Assim, a nova função score do FastText também incorpora a informação contextual nas representações vetoriais dos n-grams dos caracteres e consequentemente nas palavras que tenham alguns deles em comum na sua estrutura interna.

4.4 Skip-Gram Morfológico

O objetivo da tarefa de segmentação morfológica é segmentar palavras em seus morfemas, a menor unidade provedora de significado de uma palavra. Morfessor (SMIT et al., 2014) é um *toolkit* desenvolvido na linguagem de programação Python para segmentação morfológica estatística. Ele é composto de uma família de métodos utilizando aprendizado não supervisionado.

Para cada palavra v em nosso vocabulário V, nós definimos um conjunto m_v, onde:

$$m_v = \{x \mid x \text{ é um morfema de v}\} \tag{4.4}$$

Por exemplo, o conjunto m_v da palavra federativas é:

$$m_v = \{federa, tiva, s\} \tag{4.5}$$

Nós construímos os conjuntos m_v utilizando o Morfessor (SMIT et al., 2014). No Skip-Gram original, cada palavra v é representada por um vetor w_v. No Skip-Gram Morfológico (SGM), cada palavra v é representada da seguinte forma:

$$k_v = w_v + \sum_{x}^{m_v} z_x \tag{4.6}$$

Onde z_x é a representação vetorial de cada morfema x de v. Assim como no Skip-gram original, o SGM também tem duas representações para cada palavra, uma quando ela é o centro e outra quando ela está no contexto.

Assim, o objetivo do SGM é maximizar a função E_{sgm}:

$$E_{sgm} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \mathbf{log}(p(k_{t+j}|k_t)) \tag{4.7}$$

Onde $\mathbf{log}p(k_{t+j}|k_t)$ é calculado através de:

$$\mathbf{log}(p(k_{t+j}|k_t)) = log(\sigma(k_{t+j}^T k_t)) + \sum_{i=1}^{k} E_{k_t\sim P_n(w)}[log(\sigma(-k_i^T k_i))] \tag{4.8}$$

A partir da equação 4.8, podemos ver que ao maximizarmos $\mathbf{log}p(k_{t+j}|k_t)$ nós estaremos aprendendo que k_t está presente no contexto de $k_(t+j)$ e k_i não está no contexto de $k_(t+j)$.

4.4.1 Comparação visual com o modelo FastText

As figuras 6 e 7 apresentam uma representação visual das arquiteturas dos modelos Skip-Gram Morfológico e FastText, respectivamente. A fim de simplificar a imagem de representação dos modelos, nós utilizamos um contexto de tamanho 2, por isso as saídas tem apenas quatro palavras: $w(t-2)$, $w(t-1)$, $w(t+1)$ e $w(t+2)$, duas anteriores e posteriores a palavra central $w(t)$. Na figura 6, os vetores $m_1, m_2, ..., m_n$ representam os morfemas da palavra $w(t)$, já na figura 7, os vetores $n_1, n_2, ..., n_t$ correspondem aos n-grams dos caracteres da palavra $w(t)$.

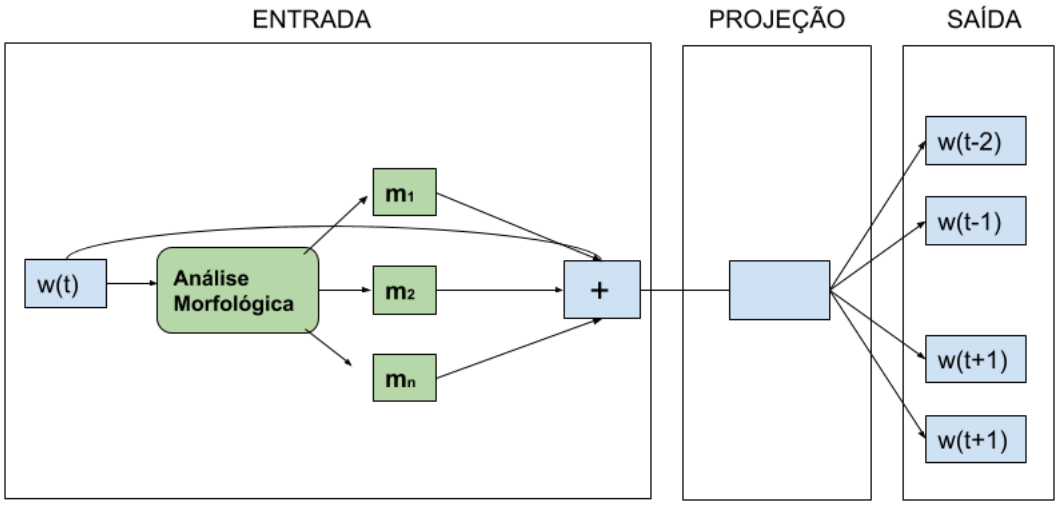

Figura 6 – Arquitetura do Skip-Gram Morfológico

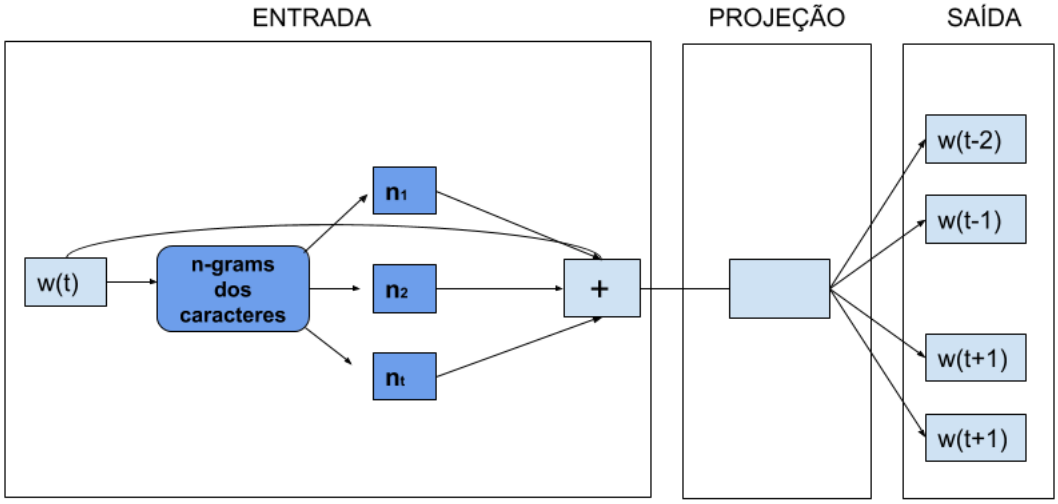

Figura 7 – Arquitetura do FastText

A partir das duas arquiteturas, fica evidente que a diferença entre os dois modelos está na fonte de informação adicionada na arquitetura base Skip-Gram. Um ponto importante a se destacar é a complexidade dos modelos, como também será aprendido as representações

vetoriais dos morfemas no Skip-Gram Morfológico e dos n-grams dos caracteres no FastText, a complexidade de ambos os modelos dependem da quantidade de *tokens* que será adicionado, pois na etapa de treinamento de ambos os modelos serão calculadas as derivadas com relação a cada um destes vetores. Por exemplo, no caso da palavra $federativas$, utilizando os conjuntos 4-gram e 5-gram dos caracteres nós precisaremos aprender 16 representações, enquanto que utilizando o conjunto $\{federa, tiva, s\}$ obtido através da análise morfológica, nós precisaremos aprender apenas 4 representações.

Baseado nas representações visuais das arquiteturas, nós também podemos analisar ambos os modelos como uma rede neural artificial composta de três camadas: entrada, projeção (intermediária) e saída. A camada de projeção é onde encontra-se os *embeddings* das palavras, morfemas ou n-grams, assim, quanto maior a dimensão desta camada, mais parâmetros serão aprendidos e consequentemente nós teremos um espaço maior para armazenar informações.

5

Experimentos

Este capítulo tem como objetivo apresentar uma descrição detalhada de todos os métodos de avaliação que foram utilizados nos experimentos e *datasets* utilizados para teste e treinamento. Os detalhes sobre a escolha dos parâmetros de treinamento de cada modelo serão apresentados nas seções dos seus respectivos resultados.

5.1 Métodos de Avaliação

Existem três categorias de métodos de avaliação de *word embeddings*: (i) Similaridade e Associação; (ii) Analogia; (iii) Categorização. Esta seção descreve cada uma dessas categorias e os seus respectivos objetivos. É importante destacar que nós utilizamos o projeto disponível em (JASTRZEBSKI; LEŚNIAK; CZARNECKI, 2017) para realizar todos os experimentos do modelo Skip-Gram Morfológico, enquanto que os experimentos do modelo GloVe Paráfrase foram implementados por nós.

5.1.1 Similaridade e Associação

Antes de explicar os *datasets* de similaridade, é preciso destacar a diferença entre similaridade e associação. Usando como exemplo as tuplas (carro, bicicleta) e (carro, petróleo), carro é semanticamente similar a bicicleta e associado (mas não similar) a petróleo. Carro e bicicleta podem ser intuitivamente entendidos como similares porque eles têm características físicas em comum, por exemplo pneus, utilidade comum (transportar). Em contraste, carro e petróleo estão associados porque frequentemente eles aparecem juntos em um espaço de contexto, devido a relação funcional entre eles.

Os *datasets* de avaliação de similaridade têm como objetivo analisar se os *word embeddings* aprendidos têm o conhecimento da similaridade semântica entre as palavras. Cada *dataset*

é composto por uma lista de triplas da forma (a, b, score), onde score é um valor indicando o quanto similar são as palavras *a* e *b*. O valor *score* dos *datasets* é definido através da média dos *scores* definidos por um conjunto de humanos julgadores.

Dado um modelo treinado, a similaridade do cosseno entre as representações vetoriais de a e b indica o quão similar elas são. Assim, para nós sabermos o quão bom é um *word embedding* neste experimento, calculamos o índice ρ da correlação de *Spearman* entre a lista de similaridades do *dataset* (definida pela média dos julgadores) e as indicadas pela similaridade do cosseno dos *word embeddings* das palavras. O quanto mais próximo de 1 o valor ρ estiver, implica que o modelo é melhor e está correlacionado com os valores de similaridade do *dataset*.

A partir da explicação acima, podemos agrupar os *datasets* desta classe em dois tipos: Similaridade e Associação. O *dataset* SimLex999 tem o objetivo de avaliar a qualidade de similaridade de um *word embedding*, já o MEN (BRUNI; TRAN; BARONI, 2014a), MTurk (RADINSKY et al., 2011), RG65 (RUBENSTEIN; GOODENOUGH, 1965), RW e WS353 (PLACING..., 2002) avaliam o grau de associação.

- **MEN**: Este *dataset* é composto por 3 grupos de pares de palavras: teste, treino e desenvolvimento. Nós utilizamos os três grupos para testar nossos modelos, totalizando 3.000 pares de palavras.
- **MTurk**: Os pares de palavras deste *dataset* foram gerados utilizando um algoritmo descrito em (RADINSKY et al., 2011). O grau de relação de cada par de palavras foi avaliado por 10 pessoas em uma escala de 1-5, em seguida foi multiplicado por 2. No total, ele consiste de 287 pares de palavras.
- **RG65**: O *dataset* RG65 contém 65 pares de palavras cujo grau de similaridade entre elas foi definido pela média de 51 julgadores. O grau de relação foi definido numa escala de 0,0 a 4,0.
- **RW**: Este *dataset* contém 2.034 pares de palavras que são relativamente raras. Para escolher as palavras deste *dataset*, eles selecionaram aleatoriamente palavras que têm frequências nos seguintes intervalos: (5, 10], (10, 100], (100, 1000], (1000, 10000].
- **SimLex999**: Este *dataset* provê uma forma de medir o quanto os *word embeddings* capturam a similaridade semântica entre palavras ao invés da relação ou associação entre elas. A proposta dele é diferente dos *datasets* WS353 e MEN. Por exemplo, *"clothes"* não é similar a *"closets"*, pois são construídos com materiais diferentes e têm funções diferentes, assim no SimLex999 eles têm um grau de similaridade igual a 1,96, enquanto que no WS353 é 9,0 porque eles têm um grau de relação alto. No total, o SimLex999 é composto de 999 pares de palavras.
- **WS353**: Este *dataset* consiste de 353 pares de palavras, onde que o grau de relação final entre elas foi definido pela média de 13-16 julgadores.

5.1.2 Analogia

Os *datasets* de analogia tem como objetivo testar a capacidade de relação lógica entre os *word embeddings* aprendidos. Estes *datasets* são compostos por uma lista de perguntas, onde cada pergunta é composta por uma tupla contendo dois pares de palavras (a, b) e (c, d). A partir de cada tupla, a pergunta é feita da seguinte forma: "a está para b, assim como c está para _", denotado como a : b -> c : ?. Supondo que w' seja a representação vetorial da palavra w e está normalizada para a forma unitária. Seguindo Mikolov (MIKOLOV et al., 2013a), nós respondemos a essas perguntas encontrando o vetor d* cuja representação vetorial é mais próxima ao vetor b - a + c de acordo com a similaridade do cosseno.

$$d^* = \underset{x \in V, x \neq a, x \neq c}{argmax} \; (b - a + c)^T x \tag{5.1}$$

A resposta é julgada como certa apenas se o vetor d* for exatamente o vetor d esperado na tupla da pergunta. Há dois tipos de *datasets* de analogia: semântico e sintático. Uma exemplo de pergunta semântica é *Englang:London -> China:Beijing*, e sintático é *amazing:amazingly -> unfortunate:unfortunately*. A tabela 3 apresenta uma lista de exemplos para outros tipos de relações semânticas e sintáticas.

Tabela 3 – Tipos de relações

Relação	**Par1 (a, b)**	**Par2 (c, d)**
Commom capital city	(Athens, Greece)	(Oslo, Norway)
All capital cities	(Astana, Kazakhstan)	(Harare, Zimbabwe)
Currency	(Angola, Kwazan)	(Iran, rial)
City-in-state	(Chicago, Illinois)	(Stockton, California)
Man,woman	(Brother, Sister)	(Grandson, Granddaughter)
Adjective-to-verb	(Apparent, Apparently)	(Rapid, Rapidly)
Opposite	(Possibly, Impossibly)	(Ethical, Unethical)
Comparative	(Great, Greater)	(Tough, Tougher)
Superlative	(Easy, Easiest)	(Lucky, Luckiest)
Plural nouns	(Mouse, Mice)	(Dollar, Dollars)

Para realizar os experimentos dessa categoria, nós utilizamos os *datasets* MSR (MIKOLOV; YIH; ZWEIG, 2013), Google (MIKOLOV et al., 2013a) e SemEval-2012 (JURGENS et al., 2012). A seguir, nós descreveremos cada um deles.

- **MSR**: Para construir este *dataset*, os autores classificaram 267 milhões de palavras de textos de notícias utilizando um *Treebank POS tag*. Após essa etapa, eles selecionaram os 100 mais frequentes adjetivos de comparação, palavras no plural, nomes possessivos e verbos na forma base. Finalmente, geraram questões de analogia agrupando aleatoriamente cada palavra desses grupos de 100 com 5 outras do mesmo grupo. No total, este *dataset* contém 8.000 perguntas de analogia.

- **Google**: Este *dataset* consiste de 8.869 analogias semânticas e 10.675 sintáticas, totalizando 19.544 perguntas de analogia. De acordo com os autores, ele foi construído em dois passos: primeiro, uma lista de pares de palavras similares foi selecionada manualmente; segundo, uma lista de perguntas é formada a partir da conexão de dois pares de palavras. Por exemplo, eles selecionaram uma lista de 68 cidades americanas e os estados aos quais elas pentencem, depois eles formaram uma lista com 2.500 perguntas selecionando pares de palavras aleatoriamente.

- **SemEval-2012**: A proposta do SemEval é diferente dos outros *datasets*. Ele também contém uma lista de dois pares de palavras (A:B) e (C:D), porém, nele o modelo deve quantificar o quanto a palavra A está relacionada a B da mesma forma que C está relacionada a D. Assim, o objetivo é medir o grau de similaridade entre as relações das palavras.

5.1.3 Categorização

Este tipo de avaliação tem como objetivo verificar a qualidade dos *word embeddings* como *features* de classificação. Todos os *datasets* deste tipo são compostos por um problema de classificação envolvendo uma única palavra ou uma sentença. Caso o dado de entrada do problema seja uma palavra w, ela é exclusivamente representada pela sua representação vetorial v_w. Por outro lado, se a entrada for uma sentença P contendo uma sequência de palavras $(p_1, p_2, p_3, ..., p_n)$, ela é representada como a média das representações vetoriais de cada palavra.

No total, esse tipo de avaliação é composto por 3 *datasets*: AP (ALMUHAREB; POESIO, 2005), BATTIG (BATTIG; MONTAGUE, 1969) e BLESS (BARONI; LENCI, 2011). Todos os problemas de classificação são resolvidos de forma não supervisionada, utilizando o algoritmo K-Means (ALSABTI; RANKA; SINGH, 1997). Nós utilizamos o K-means porque era a opção padrão presente no projeto de avaliação. O resultado da avaliação é calculado utilizando a métrica *Purity* (SCHÜTZE; MANNING; RAGHAVAN, 2008) dos *clusters* resultantes.

- **AP**: *dataset* balanceado com respeito a três fatores: frequência, classe e ambiguidadade. No total, ele contém 21 classes, cada classe contendo de 13 a 21 palavras. Por exemplo, as palavras: cerimônia, festa e graduação pertencem à classe *social occasion*;

- **BATTIG**: *dataset* composto de 200 nomes pertencentes a diferentes classes (ferramentas, roupas, animais etc);

- **BLESS**: este *dataset* é composto de 5.231 palavras, sendo cada palavra categorizada com classes do tipo "é uma pedra preciosa", "unidade de tempo", "uma fruta", "uma cor", entre outras.

5.2 *Corpus* utilizado no treinamento

Nós utilizamos o *corpus* 1 *billion word language model benchmark* (CHELBA et al., 2013) para treinar todos os *word embeddings* utilizados nos experimentos deste trabalho. Este *corpus* é composto por aproximadamente 1 bilhão de palavras.

Para construir este dataset, os autores seguiram os seguintes passos:

1. Selecionaram os textos em inglês para o *dataset*;
2. Normalizaram e tokenizaram o texto utilizando o *script* do WMT11 site;
3. Removeram sentenças duplicadas, diminuindo o número de palavras de 2,9 bilhões para aproximadamente 0,8 bilhão (82.925.094, exatamente);
4. Construíram um vocabulário composto por 793.471 palavras, incluindo os limites das sentenças $< s >$ e $< \backslash s >$;
5. Palavras fora do vocabulário foram mapeadas para o *token* <UNK>;
6. Uma partição de 1% do *dataset* foi utilizada como conjunto *held-out*;
7. O conjunto *held-out* foi embaralhado e dividido em 50 partições para ser utilizado como *dataset* de desenvolvimento e teste;
8. A taxa de palavras fora do vocabulário (*out-of-vocabulary* (OOV)) do *dataset* é de 0,28%;

6

Resultados e Discussões

Este capítulo apresenta os resultados e discussões de todos os experimentos realizados neste trabalho. Os experimentos foram divididos em dois cenários: (i) GloVe Paráfrase; (ii) Skip-Gram Morfológico. No primeiro cenário nós avaliamos a qualidade dos *word embeddings* aprendidos após utilizarmos a base de dados PPDB para enriquecer a matriz de coocorrência do GloVe. O segundo cenário avalia o uso da informação da estrutura interna das palavras (morfologia) no aprendizado dos *word embeddings*.

6.1 Cenário 1 - GloVe Paráfrase

O *baseline* deste cenário de experimentação é o método GloVe, devido ao método aqui avaliado ser uma alteração explícita da sua matriz de coocorrência. Para avaliar estes *word embeddings*, nós utilizamos três *benchmarks* diferentes: (1) SimLex999 (HILL; REICHART; KORHONEN, 2016), (2) MEN (BRUNI; TRAN; BARONI, 2014b) e (3) WordSimilarity-353 (WS353) (AGIRRE et al., 2009).

Para todos os experimentos deste cenário, nós escolhemos um $x_{max} = 100$, $\alpha = 0.75$ e treinamos o GloVe com o método Adagrad (DUCHI; HAZAN; SINGER, 2011). A taxa de aprendizado inicial foi $0,05$. Executamos 100 iterações de treinamento para cada *word embedding* dos experimentos. Como *word embedding* final, usamos $W + \tilde{W}$. É importante ressaltar que nós utilizamos a implementação original do GloVe para treinar todos os *word embeddings* deste cenário. A seguir, apresentaremos e discutiremos sobre os resultados obtidos neste cenário.

6.1.1 Resultados e discussões

Na tabela 4 nós apresentamos os melhores resultados de avaliação encontrados. Podemos observar que utilizando a base de dados PPDB nós conseguimos atingir melhores resultados em

todos os cenários. Outro aspecto importante a ser notado é que a qualidade dos *word embeddings* também melhorou de acordo com o aumento das suas dimensões. Todos os três *benchmarks*, SimLex999, WS353 e MEN utilizam a correlação de Spearman como métrica de avaliação.

Tabela 4 – Melhores resultados de cada modelo. P = Paráfrase.

Model	Size	SimLex999	WS353	MEN
GloVe	100	21,85	25,95	44,00
GloVe-P	100	21,89	27,75	44,85
GloVe	200	22,74	27,60	46,87
GloVe-P	200	**23,66**	**27,93**	**47,92**

Os valores da correlação de Spearman presentes na tabela 4 estão ponderados por um fator de 100. As figuras 8-10 apresentam as curvas dos valores da correlação de Spearman para o Glove 200 e Glove-P 200 nos seguintes *benchmarks*: SimLex999, MEN e WS353, respectivamente. Além do fato de que o Glove-P 200 apresenta melhores resultados em todos os três *benchmarks* e em todas as épocas, é evidente que durante as primeiras épocas o Glove-P fica muito próximo dos seus melhores resultados. Isto é importante porque nem sempre temos um grande poder computacional disponível para executar muitas épocas de treinamento.

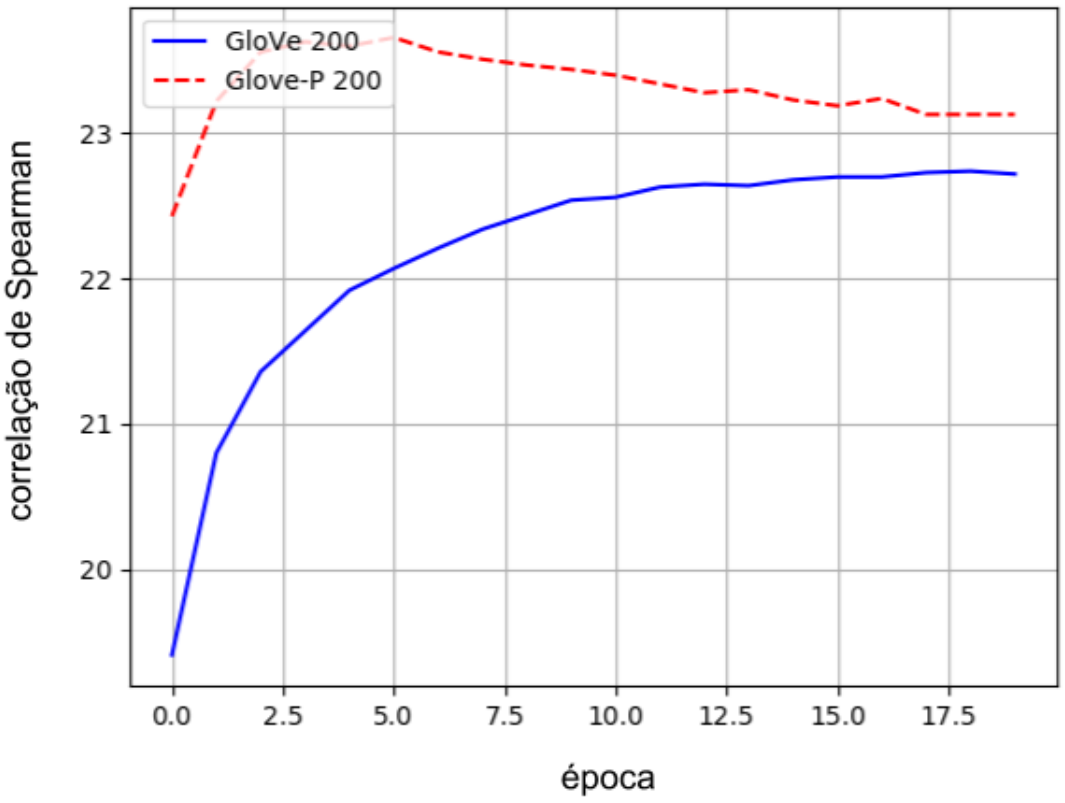

Figura 8 – Resultados do SimLex999

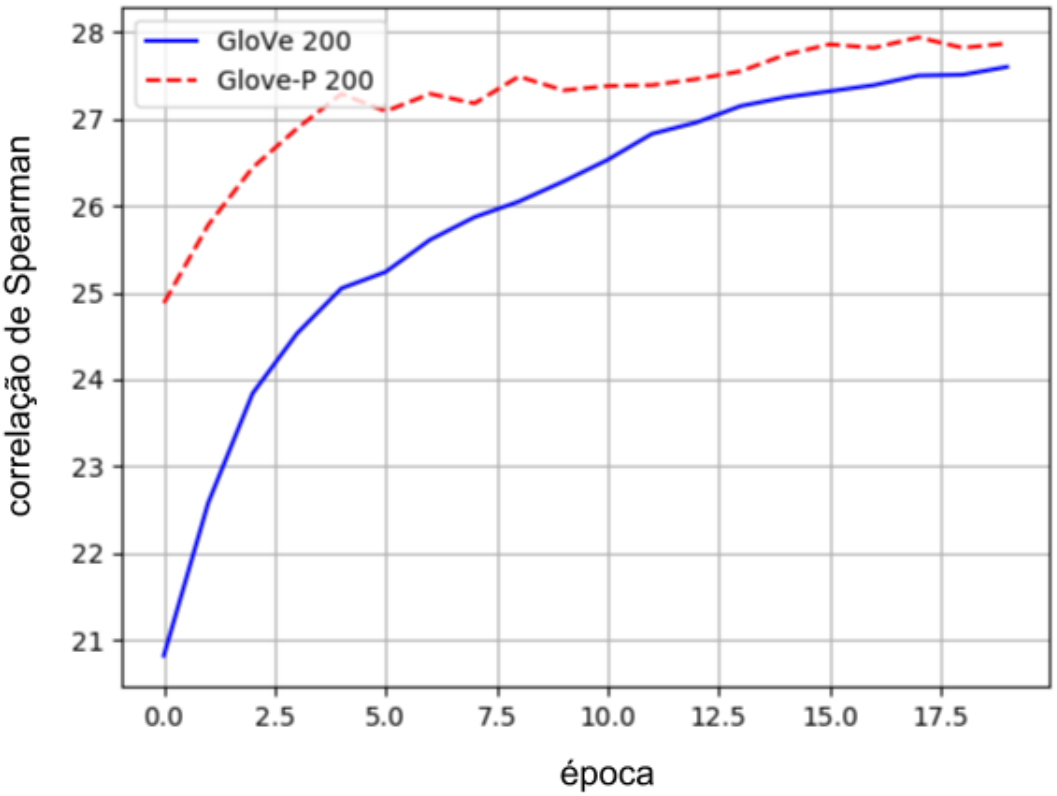

Figura 9 – Resultados do MEN

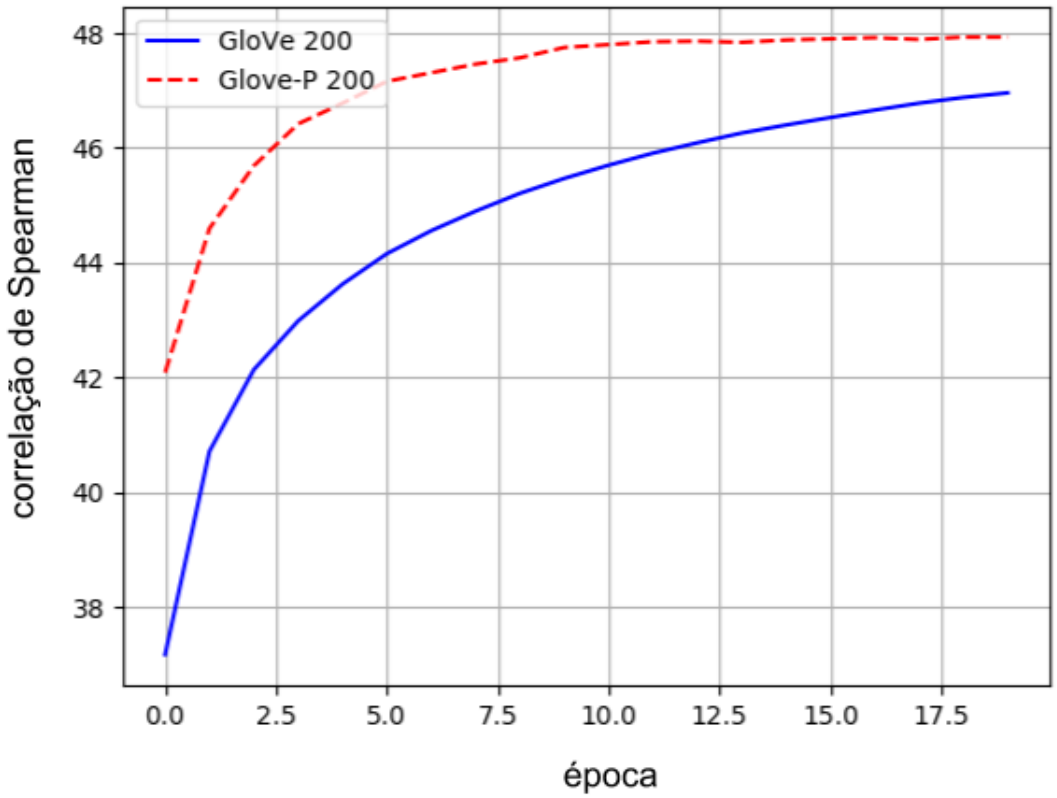

Figura 10 – Resultados do WS353

6.2 Cenário 2 - Skip-Gram Morfológico

Esta seção apresenta os resultados e discussões do Cenário 2, cujo objetivo é avaliar o uso do conhecimento morfológico das palavras ao invés do conjunto dos n-grams dos caracteres. Nós utilizamos o modelo FastText como *baseline* em razão de o mesmo ter como base a arquitetura Skip-Gram e utilizar a *bag* dos n-grams dos caracteres das palavras para representá-las. Para comparar o SGM com o FastText, nós utilizamos todos os *benchmarks* discutidos no capítulo 5.

Tabela 5 – Detalhes de treinamento

PARÂMETROS	**VALORES**
dimensões	50, 100, 200, 300
iterações	20
threads	12
negative sampling	5
context window	5
learning rate	0,1

A tabela 5 apresenta todos os valores dos parâmetros utilizados durante o treinamento dos *word embeddings* analisados nos experimentos do cenário 2. É importante destacar que os valores de *threads*, *negative sample*, *context window* e *learning rate* foram escolhidos de acordo com os valores *default* do projeto FastText. A subseção 6.2.1 apresenta e discute sobre os resultados obtidos neste cenário.

6.2.1 Resultados e discussões

Nós agrupamos os resultados desta seção em 5 tabelas. A tabela 6 apresenta os resultados dos experimentos utilizando os *datasets* de analogia, as tabelas 7 e 8 mostram os resultados de similaridade, 9 exibe os resultados de categorização e por fim, a tabela 10 apresenta a comparação dos tempos de treinamento entre os modelos. O nome "ft-d50-e20"representa o modelo FastText de dimensão 50 treinado durante 20 épocas, enquanto que "sgm-d50-e20"representa o modelo Skip-Gram Morfológico também de dimensão 50 treinado durante 20 épocas. Para treinar os modelos deste cenário, nós utilizamos um *desktop* Dell com o processador i5 da oitava geração, 8 GB de memória RAM, 1 TB de memória secundária e executando o sistema operacional Ubuntu 16.04. É importante destacar que nós asseguramos que ambos os modelos fossem executados em uma mesma situação ambiente para obter os valores da tabela 10.

Tabela 6 – Resultados de Analogia

Name	Google	MSR	SE-2012
ft-d50-e20.vec	0,0	0,000625	0,12747
sgm-d50-e20.vec	**0,004912**	**0,001125**	**0,128099**
ft-d100-e20.vec	0,008391	0,00425	0,142526
sgm-d100-e20.vec	**0,063549**	**0,012625**	**0,143793**
ft-d200-e20.vec	0,08683	0,05075	0,154959
sgm-d200-e20.vec	**0,242325**	**0,104**	**0,164312**
ft-d300-e20.vec	0,128223	0,08225	0,167914
sgm-d300-e20.vec	**0,33202**	**0,210875**	**0,179591**

De acordo com os resultados apresentados na tabela 6, podemos ver que o modelo SGM teve resultados superiores ao FT em todos os *datasets* e todas as dimensões dos *embeddings*. Entretanto, os *embeddings* de tamanho 50 não tiveram resultados bons nos três *datasets*. Também é importante destacar a variação dos resultados com a variação do tamanho dos *embeddings*, por exemplo os resultados utilizando o *dataset* Google, podemos ver que o modelo sgm-d50 teve resultado $0,005$ enquanto que o resultado do sgm-d300 foi $0,332$, ou seja, o tamanho dos *embeddings* também influenciaram bastante nos resultados desta tarefa.

Tabela 7 – Resultados de similaridade - Parte 1.

Name	MEN	MTurk	RG65	RW
ft-d50-e20.vec	**0,643736**	0,633283	**0,582444**	**0,24987**
sgm-d50-e20.vec	0,640213	**0,633706**	0,555911	0,224664
ft-d100-e20.vec	**0,691263**	**0,661416**	**0,648381**	**0,276416**
sgm-d100-e20.vec	0,682842	0,626963	0,639836	0,250898
ft-d200-e20.vec	**0,714662**	**0,673302**	**0,705861**	**0,304246**
sgm-d200-e20.vec	0,71146	0,643192	0,674454	0,285549
ft-d300-e20.vec	**0,729393**	**0,67404**	**0,729661**	**0,312796**
sgm-d300-e20.vec	0,711628	0,662214	0,677973	0,290232

Tabela 8 – Resultados de similaridade - Parte 2

Name	SimLex999	WS353	WS353R	WS353S
ft-d50-e20.vec	**0,251962**	**0,608528**	0,550006	**0,67745**
sgm-d50-e20.vec	0,249763	0,606456	**0,555385**	0,670089
ft-d100-e20.vec	0,29213	**0,642096**	**0,585298**	0,700297
sgm-d100-e20.vec	**0,294585**	0,641834	0,5822	**0,703131**
ft-d200-e20.vec	0,328424	**0,652081**	0,581129	**0,730383**
sgm-d200-e20.vec	**0,329708**	0,650738	**0,586298**	0,725935
ft-d300-e20.vec	0,342441	**0,663631**	0,587315	0,748731
sgm-d300-e20.vec	**0,347435**	0,662206	**0,590457**	**0,755688**

Os resultados apresentados nas tabelas 7 e 8 mostram que o tamanho dos *embeddings* não influenciam tanto na tarefa de similaridade quanto na tarefa de analogia. Na tarefa de similaridade, o modelo SGM teve resultados competitivos comparado ao FT, tendo uma diferença de desempenho de no máximo $0,05$ no caso do *dataset RG65* utilizando os modelos ft-d300 e sgm-d300. É importante salientar que o FT apresentou melhores resultados em todos os casos do *dataset Rare Words* (RW), assim mostrando que ele generaliza melhor no cenário de palavras desconhecidas. A nossa suposição do porque o FastText generaliza melhor para palavras desconhecidas é que mesmo essas palavras tendo pouca informação sobre o contexto dela (devido a aparecem poucas vezes no corpus), mas alguns dos seus n-grams dos caracteres podem estar presentes em palavras que são abundantes em informação do contexto (aparecem muitas vezes no corpus), assim fazendo com que as palavras raras também utiliza a informação do contexto dessa palavras abundantes em informação do contexto.

Tabela 9 – Resultados de categorização

Name	AP	BLESS	BATTING
ft-d50-e20.vec	0,609453	**0,75**	**0,429937**
sgm-d50-e20.vec	**0,621891**	0,735	0,418276
ft-d100-e20.vec	**0,594527**	0,765	0,429363
sgm-d100-e20.vec	0,589552	0,765	**0,44676**
ft-d200-e20.vec	**0,606965**	0,795	**0,432231**
sgm-d200-e20.vec	0,594527	**0,805**	0,42554
ft-d300-e20.vec	0,587065	0,805	**0,426114**
sgm-d300-e20.vec	**0,597015**	**0,815**	0,422864

A partir dos resultados apresentados na tabela 9, podemos perceber que os modelos têm resultados semelhantes, com no máximo $0,01$ de diferença entre eles. Além do mais, os

resultados nos três *datasets* não variam muito com relação a dimensão dos *embeddings*.

Tabela 10 – Tempo de treinamento

Name	**Tempo (minutos)**
ft-d50	307m27
sgm-d50	**196m11**
ft-d100	409m42
sgm-d100	**268m50**
ft-d200	647m32
sgm-d200	**414m30**
ft-d300	853m58
sgm-d300	**541m35**

De acordo com os resultados presentes nas tabelas 6-9, ficou evidente que o modelo SGM é melhor que o FT em tarefas de analogia, enquanto que nas avaliações de Similaridade e Categorização os modelos tem resultados equivalentes. Podemos destacar que o modelo FT apresentou melhores resultados no *dataset Rare Words*. Além disso, a tabela 10 mostra que o tempo de treinamento do modelo SGM é aproximadamente 40% mais rápido que o modelo FT.

7

Aplicações

Este capítulo apresenta aplicações do uso dos *word embeddings* para soluções de problemas de PLN. A finalidade deste capítulo é a de mostrar ao interessado nos modelos propostos, sua aplicabilidade para tarefas reais e importantes de PLN, independentemente da qualidade aferida pelas métricas já discutidas nos capítulos anteriores. Nós escolhemos utilizar dois problemas de PLN como exemplo, são eles: (i) Reconhecimento de entidades nomeadas; (ii) Discurso de ódio. Nós escolhemos essas duas aplicações porque a primeira trabalha a capacidade do modelo classificar uma palavra diante de um contexto e a segunda aplicação é um problema em que é necessário representar uma única sentença (ou um parágrafo) e classificá-la. Assim, nós utilizamos dois problemas com características diferentes.

7.1 Reconhecimento de entidades nomeadas

Reconhecimento de entidades nomeadas (REN) é uma tarefa de PLN cujo objetivo é identificar e classificar entidades em um texto dado. Alguns exemplos dos tipos de entidades são: pessoa, local, organização, valor e tempo. Essa tarefa é muito importante no processo de extração de informação em textos. Um modelo para resolver este problema deve receber como entrada a representação da palavra numa dada sentença e retornar qual tipo de entidade ela pertence ou se ela não é uma entidade.

7.1.1 Modelo

A figura 11 apresenta uma forma visual da arquitetura utilizada para treinar o nosso modelo de REN. Esta arquitetura está presente no trabalho de (JúNIOR et al., 2016). O vetor $w = [w_1, w_2, ..., w_n]$ representa a informação da palavra que será classificada, onde cada w_i é um índice de uma palavra. Este vetor contém três informações sobre a palavra a ser classificada: (i)

contexto esquerdo, (ii) índice da palavra e (iii) contexto direito. Por exemplo, utilizando uma janela de contexto de tamanho 3, o nosso vetor w terá 7 posições. A camada *embeddings* recebe o vetor w como entrada e retorna uma matriz M de saída, onde cada linha i de M representa o *word embedding* da palavra w_i. Os pesos dessa camada podem ser inicializados utilizando os *word embeddings* aprendidos por modelos como Skip-Gram Morfológico e GloVe Paráfrase, permitindo a rede neural artificial utilizar um conhecimento pré-treinado. Em seguida, por se tratar de um problema de classificação de sequência, a arquitetura utiliza duas camadas LSTM com uma camada *Dropout* entre elas e finaliza com uma camada densa utilizando a função *softmax* para retornar as probabilidades de classificação.

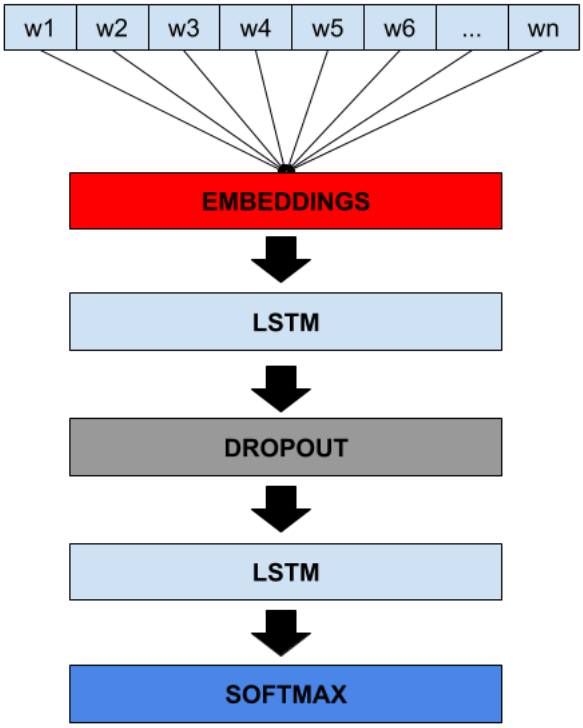

Figura 11 – Arquitetura utilizada para REN.

7.1.2 Experimentos e resultados

Para inicializar os pesos da camada *embeddings* dos modelos deste experimento, nós utilizamos os *word embeddings* treinados com o modelo Skip-Gram Morfológico. O experimento desta aplicação foi dividido em dois cenários. No primeiro cenário nós utilizamos o *dataset* PrimeiroHarem como treinamento e o MiniHarem como teste. No segundo cenário foi utilizado o *dataset* Paramopama como treinamento e os 10% finais do WikiNER como teste. Estes dois cenários foram definidos de acordo com o trabalho (JúNIOR et al., 2016).

Na tabela 11 é apresentado o número de *tokens* por classe de cada *dataset*, deixando explícito que todos os *datasets* utilizados nesse experimento são desbalanceados. Além do mais, é importante salientar que o classificador do primeiro experimento é treinado para 6 classes

(Outro, Pessoa, Local, Organização, Tempo e Valor), enquanto que o do segundo cenário é treinado para 5 classes. A classe Outro significa que a palavra não é uma entidade nomeada.

Tabela 11 – Distribuição dos *datasets*.

Dataset	Outro	Pessoa	Local	Organização	Tempo	Valor
PrimeiroHarem	86.682	2.242	2.036	2.168	1.420	1.121
MiniHarem	59.023	1.651	1.385	1.372	701	712
Paramopama	263.916	7.326	17.461	7.154	10.827	0
WikiNER	122.671	5.510	6.536	2.509	6.506	0

A arquitetura do modelo Paramopama-WNN está presente na figura 11. O modelo Paramopama-CWNN é semelhante ao da figura 11, entretanto, também utiliza uma camada convolucional para extrair informações dos caracteres das palavras (*char embeddings*), utilizando, assim, informações das palavras e dos seus caracteres. É importante ressaltar que todos os resultados dos modelos Paramopama-CWNN e Paramopama-WNN foram retirados do trabalho (JúNIOR et al., 2016).

A fim de comparar os nosso resultados com o modelo *baseline*, nós utilizamos os mesmo cenários e métricas presentes em (JúNIOR et al., 2016). As tabelas 12-13 apresentam os resultados do cenário 1. Podemos perceber que utilizar os *word embeddings* morfológicos causou uma influência importante, umas vez que o modelo Paramopama-SGM obteve 74,07% de Medida-F comparado a 71,22% do modelo Paramopama-WNN. Outro aspecto importante é o resultado da medida-F da classe Tempo, o Paramopama-SGM obteve 70,00% enquanto que o resultado do Paramopama-WNN é 60,77%. A nossa suposição para este resultado é que, como as palavras dessas entidades não são semelhantes (uma vez que representam datas, meses ou épocas do ano), os morfemas das palavras do contexto desta entidade são semelhantes, assim a informação morfológica presente implicitamente nos *word embeddings* pode ter ajudado.

Tabela 12 – Resultados - Cenário 1.

Modelo	Precisão(%)	Cobertura(%)	Medida-F(%)
Paramopama-SGM	74,10	**74,59**	**74,07**
Paramopama-CWNN	**75,13**	68,38	71,35
Paramopama-WNN	73,68	69,26	71,22

Tabela 13 – Resultados por entidades - Cenário 1.

Modelo	Pessoa(%)	Local(%)	Organização(%)	Tempo(%)	Valor(%)
Paramopama-SGM	**77,26**	**70,13**	**55,28**	**70,00**	73,69
Paramopama-CWNN	75,95	68,57	54,22	55,43	**75,81**
Paramopama-WNN	76,87	68,75	52,07	60,77	70,85

Os resultados do cenário 2 estão presentes nas tabelas 14-15. Assim como no cenário 1, o modelo utilizando os *word embeddings* morfológicos obtiveram melhores resultados, entretanto,

nesse cenário foi apenas uma leve melhora. Mais uma vez, o resultado obtido pelo Paramopama-SGM na classe Tempo foi o que mais se destacou, contribuindo para a nossa suposição de que os morfemas presentes no contexto de palavras da classe Tempo são semelhantes. Um trabalho futuro a ser feito neste experimento é avaliar os tipos de escritas dos *datasets* Paramopama, WikiNER, PrimeiroHarem e MiniHarem para que possamos tirar conclusões a respeito da contribuição do uso da informação dos morfemas.

Tabela 14 – Resultados - Cenário 2.

Modelo	Precisão(%)	Cobertura(%)	Medida-F(%)
Paramopama-SGM	**88,45**	88,68	**88,49**
Paramopama-CWNN	83,97	78,39	80,50
Paramopama-WNN	86,45	**89,77**	88,08

Tabela 15 – Resultados por entidades - Cenário 2.

Modelo	Pessoa(%)	Local(%)	Organização(%)	Tempo(%)
Paramopama-SGM	**89,06**	87,78	**75,65**	**91,33**
Paramopama-CWNN	87,45	83,57	62,73	71,00
Paramopama-WNN	87,00	**87,82**	75,41	88,00

7.2 Discurso de ódio

O discurso do ódio refere-se a palavras, frases ou expressões que insultam, intimidam ou assediam pessoas em virtude de sua raça, cor, etnicidade, nacionalidade, sexo ou religião, ou que têm a capacidade de instigar violência, ódio ou discriminação contra as pessoas (MOURA, 2016). A detecção automática desses conteúdos se propõe a determinar se há ou não conteúdo odioso dentro de textos ou mesmo determinar qual seu sub-tipo. Alguns dos seus tipos mais comuns são: misoginia, racismo, xenofobia, homofobia, discriminação por aparência, discriminação por ideologia política e etc.

Sua detecção automática representa um desafio por diversas razões: Primeiro porque ela exige um nível de compreensão avançado da estrutura e semântica dos comentários, envolvendo detecção da intenção do usuário e da presença de ironia/sarcasmo, fatores que englobam, dentre outras coisas, conhecimento de mundo. Em outras palavras, envolve que o computador tenha representados conceitos comuns que expliquem e delimitem de forma objetiva elementos do mundo real.

7.2.1 Modelo

Diversos modelos foram propostos para a tarefa de detecção de discursos de ódio. Dentre tais modelos, aqueles que utilizaram *word embeddings* ganharam notoriedade pelo desempenho,

particularmente quando usados em arquiteturas *deep learning* (SCHMIDT; WIEGAND, 2017) (BADJATIYA et al., 2017).

O modelo abordado neste trabalho é adaptado de BADJATIYA et al. (2017) e ilustrado na Figura 12. O fluxo de treinamento dos comentários, desde a camada de entrada até a sua efetiva classificação, segue a ordem dos passos discriminados a seguir:

1. **Pré-processamento e vetorização**: Os dados são pré-processados e vetorizados de forma a respeitar o tamanho máximo dentre todos os dados de treinamento, tamanho esse representado pela variável *max_sentence_length*. A vetorização consiste da criação de vetores de índices baseados na posição de cada *token* em relação ao vocabulário do *dataset*.

2. **Camada de entrada**: Uma vez vetorizados, os dados são submetidos em *batchs* de tamanho *batch_size*. A função dessa camada é criar uma representação inicial dos *embeddings* a serem aprendidos pelo modelo. Os valores de suas dimensões são calculados através de um distribuição uniforme e a quantidade delas é controlada através do parâmetro *embedding_dim*.

3. **Camada LSTM**: A função desta camada é aprender uma representação adequada para cada comentário submetido, gerando, portanto, *embeddings* específicos voltados ao domínio de discursos de ódio (BADJATIYA et al., 2017). Essas representações são usadas então para classificação do modelo, conforme detalharemos a seguir. Os parâmetros da LSTM podem ser conferidos na Tabela 16.

4. **Camada densa**: Em nosso modelo, a camada densa consiste de um Multi-Layer Perceptron, o qual é responsável por receber o vetor de dimensão *units* da camada LSTM e determinar sua classe. O erro referente à classificação nesta camada é então propagado para as camadas anteriores e a atualização de seus parâmetros realizada conforme o algoritmo de otimização escolhido. A saída desta camada é submetida à função *softmax*, responsável por calcular a probabilidade para cada classe e com isso, predizer a classe para os comentários recebidos.

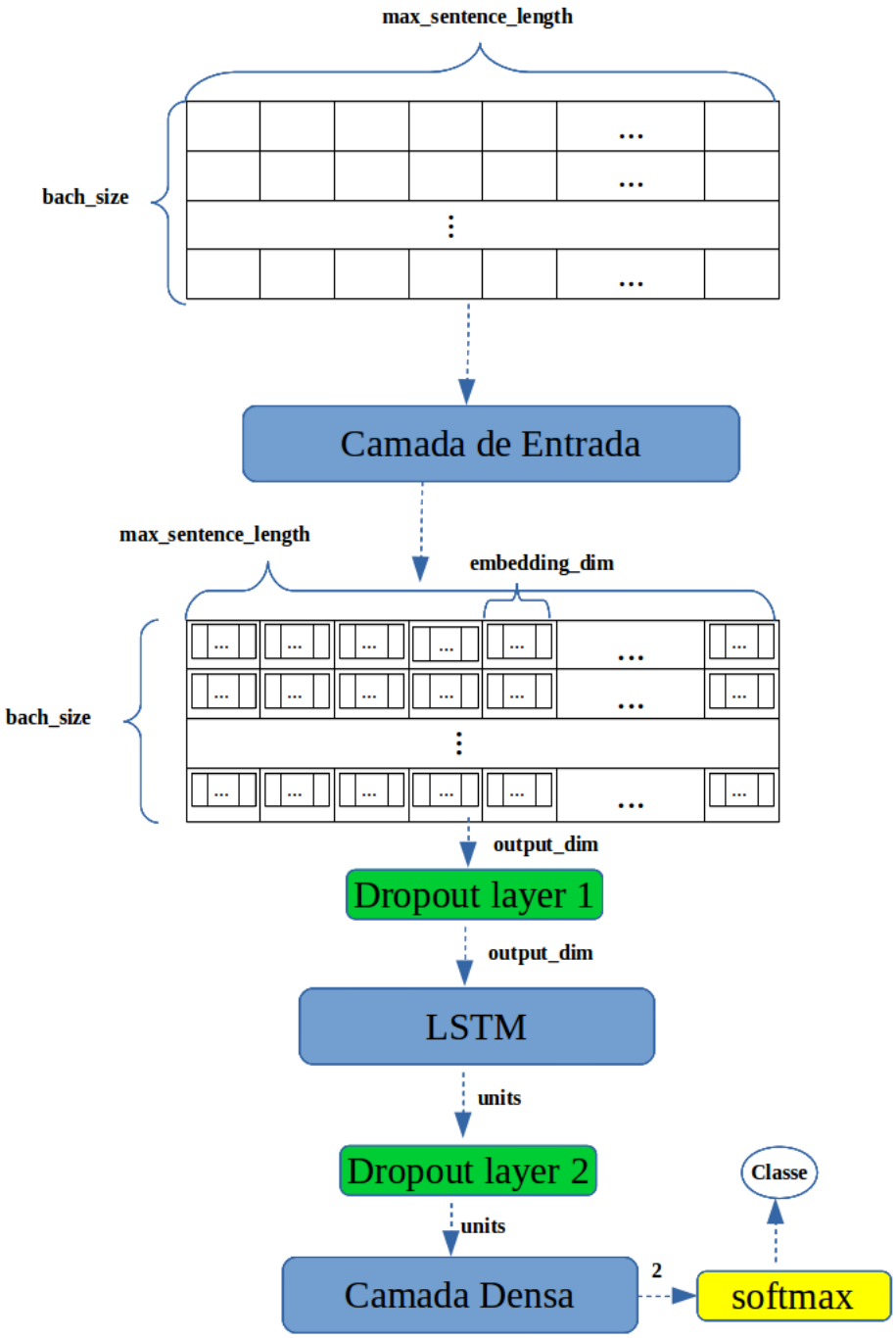

Figura 12 – Arquitetura LSTM usada para detecção de discursos de ódio.

Tabela 16 – Configuração padrão de parâmetros do modelo LSTM.

Parâmetro	**Descrição**	**Valor**
input_dim	Valor numérico máximo esperado na camada de entrada	10000
output_dim	Tamanho do *embedding* de palavra a ser gerado	200
input_length	Tamanho do vetor de sentença de entrada	Variável
units	Quantidade de células na camada LSTM	200
weights	Pesos de inicialização da camada de entrada	[]
dropout_rate1	Taxa de *dropout* da camada de entrada	0,25
dropout_rate2	Taxa de *dropout* da camada LSTM	0,50
optimizer	Otimizador da função e perda	"rmsprop"
loss_fun	Função de erro	"categorical_crossentropy"
bach_size	Tamanho do *mini bach*	128

7.2.2 Experimentos e Resultados

O *dataset* usado nos experimentos é o mesmo de BADJATIYA et al. (2017) e consiste de 16.131 *tweets* rotulados dentre uma das três classes distintas: "sexism"(Sexismo/Misoginia), "racism"(Racismo) e "none"(Nenhum dos dois). Nem todos os *tweets* estavam disponíveis no momento do *download* e, por esta razão, nosso *dataset* é menor que o considerado no trabalho referência.

Em nossos experimentos usamos a LSTM e a combinação dela com um Gradient Boosting Decision Tree (GBDT), que representa a melhor arquitetura validada por BADJATIYA et al. (2017). Salvamos o resultado da classificação usando a rede neural e a camada densa Perceptron Multi-camadas (MLP) para então, utilizando o mesmo modelo treinado, extrairmos os *embeddings* aprendidos e os submetermos à GBDT, executando, portanto, uma nova classificação. Em outras palavras, quando combinamos o modelo LSTM com o GBDT, ignoramos o resultado da classificação da rede neural usando a camada densa e executamos um novo treinamento com a GBDT através dos *embeddings* aprendidos.

O artigo referência executa diversos experimentos a fim de validar quais modelos e configurações obtêm os melhores resultados. Em alguns experimentos são usados *word embeddings* treinados utilizando os modelos GloVe ou FastText para inicializar os modelos e comparar seu desempenho com os experimentos do mesmo trabalho que não os utilizam.

Por esta razão, para cada uma dessas arquiteturas citadas anteriormente, executamos dois experimentos. No primeiro, configuramos os pesos da camada de entrada como sendo os *embeddings* GloVe do vocabulário do *dataset* utilizado, da mesma maneira que o artigo referência. No segundo, os pesos foram os *embeddings* do modelo Skip-Gram morfológico extraídos do mesmo *dataset* com o objetivo de comparar os resultados do artigo com o nosso, usando o modelo de *embeddings* desenvolvido neste trabalho.

Assim como no artigo referência, os nossos modelos foram submetidos a uma validação cruzada com 10 *folds* treinados com 10 épocas cada e as métricas calculadas através da média de seus valores em cada *fold*. Portanto, o valor de Medida-F pode estar fora do intervalo definido

pela precisão e cobertura apresentados, uma vez que aquela não foi calculada diretamente a partir destes, e sim da média de seu histórico. Os resultados são listados na Tabela 17 em termos das métricas de Precisão, Cobertura e Medida-F. Os valores entre parênteses presentes em cada uma dessas métricas nas linhas dos experimentos 1 e 4 são os resultados obtidos por BADJATIYA et al. (2017) na execução dos mesmos experimentos.

Tabela 17 – Resultado dos experimentos com *dataset* de discursos de ódio.

#	Arquitetura	Inicialização	Precisão	Cobertura	Medida-F
1	LSTM	GloVe	**0,833 (vs. 0,807)**	**0,831 (vs. 0,809)**	**0,827 (vs. 0,808)**
2	LSTM	Morfo	0,829	0,823	0,818
3	LSTM + GBDT	GloVe	0,905	0,906	0,905
4	LSTM + GBDT	Morfo	**0,910 (vs. 0,849)**	**0,911 (vs. 0,848)**	**0,910 (vs. 0,848)**

Nota: Morfo = Skip-Gram morfológico

8

Conclusão

Esta dissertação apresentou dois métodos para aprendizado de *word embeddings* utilizando conhecimento especialista: (i) GloVe Paráfrase; (ii) Skip-Gram Morfológico (SGM).

O primeiro método utiliza o conhecimento de paráfrase a nível de palavras para preencher a matriz de coocorrência do GloVe, assim tornando os *word embeddings* destas paráfrases mais semelhantes, uma vez que são palavras com o mesmo significado. Nós comparamos o método GloVe Paráfrase com o GloVe original utilizando 3 *benchmarks* bastante difundido na literatura. A partir dos resultados encontrados podemos inferir que utilizando o conhecimento de Paráfrase a nível de palavras nós precisamos de menos épocas de treinamento para obter boas representações vetoriais de palavras. Este fato ocorre porque utilizando esse conhecimento para preencher a matriz de coocorrência do GloVe é como se nós estivéssemos utilizando um *corpus* de treinamento maior, pois a matriz de coocorrência passa a ter informações que antes não tinha.

O segundo método emprega o conhecimento morfológico das palavras para substituir a *bag* de n-grams dos caracteres utilizada no modelo FastText. Utilizar essa *bag* de n-grams é uma solução força bruta, pois tenta todas as combinações possíveis dos n-grams do caracteres, limitados a um número n. Como o objetivo de utilizar essa *bag* dos n-grams é também aprender a informação da estrutura interna das palavras, nós utilizamos o conhecimento morfológico delas porque é uma informação cientificamente embasada. Nós comparamos o SGM com o FastText utilizando 12 *benchmarks* e a partir dos resultados ficou evidente que o SGM obtém qualidade competitiva com o FastText, além de que precisa de 40% menos tempo que o FastText para treinar os *words embeedings*. Manter a qualidade dos *word embeddings* e diminuir o tempo de treinamento é muito importante, porque apesar de nós termos executado os experimentos com um corpus composto de 1B *tokens*, existem *corpora* contendo 820B *tokens* como o *Commom Crawl* presente nos experimentos de (PENNINGTON; SOCHER; MANNING, 2014a).

Além dos dois métodos desenvolvidos, nós utilizamos os *word embeddings* aprendidos

com o SGM para aplicá-los a dois problemas de Processamento de Linguagem Natural: Reconhecimento de entidades nomeadas e Detecção de Discurso de ódio. Os resultados encontrados mostraram que utilizando o conhecimento morfológico implícito nos *word embeddings* nós conseguimos melhorar os resultados dos *baselines* escolhidos para resolver estes dois problemas.

De acordo com o que foi apresentado nesta dissertação, fica explícito que o objetivo geral deste trabalho foi alcançado, pois os resultados obtidos nos experimentos mostraram que utilizando conhecimento especialista nós conseguimos aprender boas representações vetoriais de palavras utilizando um menor tempo de treinamento.

Este trabalho pode ser expandido de diversas formas, entre elas podemos destacar o uso da etimologia da palavra para aprender *word embeddings* de palavras de línguas diferentes. Além do mais, podemos investigar os métodos para efetuar a análise morfológica e detecção de paráfrases, a fim de produzir novos experimentos verificando a qualidade desses modelos. Outra investigação a ser feita é verificar o impacto dessas informações em línguas como o Português, Espanhol e Alemão.

A nível de aplicação, é necessário investigar arquiteturas de aprendizado profundo para utilizar os *embeddings* dos morfemas explicitamente, uma vez que eles também são aprendidos no modelo SGM.

Referências

AGIRRE, E. et al. A study on similarity and relatedness using distributional and wordnet-based approaches. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. [S.l.], 2009. p. 19–27. Citado na página 48.

ALMUHAREB, A.; POESIO, M. Concept learning and categorization from the web. In: *proceedings of the annual meeting of the Cognitive Science society*. [S.l.: s.n.], 2005. v. 27, n. 27. Citado na página 46.

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2014. Citado na página 27.

ALSABTI, K.; RANKA, S.; SINGH, V. An efficient k-means clustering algorithm. 1997. Citado na página 46.

AUER, S. et al. Dbpedia: A nucleus for a web of open data. *The semantic web*, Springer, p. 722–735, 2007. Citado 2 vezes nas páginas 19 e 33.

BADJATIYA, P. et al. Deep learning for hate speech detection in tweets. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 26th International Conference on World Wide Web Companion*. [S.l.], 2017. p. 759–760. Citado 3 vezes nas páginas 59, 61 e 62.

BARONI, M.; LENCI, A. How we blessed distributional semantic evaluation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. [S.l.], 2011. p. 1–10. Citado na página 46.

BATTIG, W. F.; MONTAGUE, W. E. Category norms of verbal items in 56 categories a replication and extension of the connecticut category norms. *Journal of experimental Psychology*, American Psychological Association, v. 80, n. 3p2, p. 1, 1969. Citado na página 46.

BIRD, S. Nltk: the natural language toolkit. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the COLING/ACL on Interactive presentation sessions*. [S.l.], 2006. p. 69–72. Citado na página 19.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006. Citado na página 14.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. Citado 2 vezes nas páginas 15 e 32.

BOLLACKER, K. et al. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. [S.l.], 2008. p. 1247–1250. Citado 3 vezes nas páginas 15, 19 e 33.

BORDES, A. et al. Translating embeddings for modeling multi-relational data. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 2787–2795. Citado na página 33.

BOTSTEIN, D. et al. Gene ontology: tool for the unification of biology. *Nat Genet*, v. 25, n. 1, p. 25–9, 2000. Citado na página 17.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. [S.l.]: Springer, 2010. p. 177–186. Citado 2 vezes nas páginas 26 e 33.

BRUNI, E.; TRAN, N. K.; BARONI, M. Multimodal distributional semantics. *J. Artif. Int. Res.*, AI Access Foundation, USA, v. 49, n. 1, p. 1–47, jan. 2014. ISSN 1076-9757. Disponível em: <http://dl.acm.org/citation.cfm?id=2655713.2655714>. Citado na página 44.

BRUNI, E.; TRAN, N.-K.; BARONI, M. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, v. 49, n. 2014, p. 1–47, 2014. Citado na página 48.

BUDANITSKY, A.; HIRST, G. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In: *Workshop on WordNet and other lexical resources*. [S.l.: s.n.], 2001. v. 2, p. 2–2. Citado 2 vezes nas páginas 15 e 35.

CARLSON, A. et al. Toward an architecture for never-ending language learning. In: *AAAI*. [S.l.: s.n.], 2010. v. 5, p. 3. Citado 2 vezes nas páginas 19 e 33.

CHELBA, C. et al. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013. Disponível em: <http://arxiv.org/abs/1312.3005>. Citado na página 47.

COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: ACM. *Proceedings of the 25th international conference on Machine learning*. [S.l.], 2008. p. 160–167. Citado na página 21.

COLLOBERT, R. et al. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, v. 12, n. Aug, p. 2493–2537, 2011. Citado 5 vezes nas páginas 15, 18, 21, 31 e 32.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011. Citado 2 vezes nas páginas 22 e 48.

ETZIONI, O. et al. Open information extraction: The second generation. In: *IJCAI*. [S.l.: s.n.], 2011. v. 11, p. 3–10. Citado na página 17.

FADER, A.; ZETTLEMOYER, L.; ETZIONI, O. Open question answering over curated and extracted knowledge bases. In: ACM. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2014. p. 1156–1165. Citado na página 17.

FERRUCCI, D. et al. Building watson: An overview of the deepqa project. *AI magazine*, v. 31, n. 3, p. 59–79, 2010. Citado na página 18.

FRIED, D.; DUH, K. Incorporating both distributional and relational semantics in word representations. *arXiv preprint arXiv:1412.4369*, 2014. Citado na página 35.

GANITKEVITCH, J.; Van Durme, B.; CALLISON-BURCH, C. PPDB: The paraphrase database. In: *Proceedings of NAACL-HLT*. Atlanta, Georgia: Association for Computational Linguistics, 2013. p. 758–764. Citado na página 37.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 4 vezes nas páginas 9, 25, 26 e 27.

GUTMANN, M.; HYVÄRINEN, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2010. p. 297–304. Citado 2 vezes nas páginas 31 e 33.

HAGHIGHI, A.; VANDERWENDE, L. Exploring content models for multi-document summarization. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. [S.l.], 2009. p. 362–370. Citado na página 17.

HILL, F.; REICHART, R.; KORHONEN, A. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, MIT Press, 2016. Citado na página 48.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 82–97, 2012. Citado na página 18.

HINTON, G. E.; MCCLELLAND, J. L.; RUMELHART, D. E. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. In: RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). Cambridge, MA, USA: MIT Press, 1986. cap. Distributed Representations, p. 77–109. ISBN 0-262-68053-X. Disponível em: <http://dl.acm.org/citation.cfm?id=104279.104287>. Citado na página 21.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 27.

HUNTER, L.; COHEN, K. B. Biomedical language processing: what's beyond pubmed? *Molecular cell*, Elsevier, v. 21, n. 5, p. 589–594, 2006. Citado na página 17.

JASTRZEBSKI, S.; LEŚNIAK, D.; CZARNECKI, W. M. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017. Citado na página 43.

JEAN, S. et al. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014. Citado na página 18.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009. ISBN 0131873210. Citado na página 22.

JURGENS, D. A. et al. Semeval-2012 task 2: Measuring degrees of relational similarity. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. [S.l.], 2012. p. 356–364. Citado 2 vezes nas páginas 21 e 45.

JúNIOR, C. A. et al. Uma arquitetura híbrida lstm-cnn para reconhecimento de entidades nomeadas em textos naturais em língua portuguesa. 2016. Citado 4 vezes nas páginas 14, 55, 56 e 57.

LAKKARAJU, H.; SOCHER, R.; MANNING, C. Aspect specific sentiment analysis using hierarchical deep learning. In: *NIPS Workshop on Deep Learning and Representation Learning*. [S.l.: s.n.], 2014. Citado na página 14.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Research, v. 521, n. 7553, p. 436–444, 2015. Citado na página 18.

LU, J. et al. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887*, 2016. Citado na página 14.

LUONG, T.; SOCHER, R.; MANNING, C. Better word representations with recursive neural networks for morphology. In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. [S.l.: s.n.], 2013. p. 104–113. Citado na página 32.

MAAS, A. L. et al. Learning word vectors for sentiment analysis. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. [S.l.], 2011. p. 142–150. Citado na página 21.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. Citado 6 vezes nas páginas 9, 15, 18, 21, 22 e 45.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 3111–3119. Citado 3 vezes nas páginas 15, 31 e 33.

MIKOLOV, T.; YIH, W.-t.; ZWEIG, G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. [S.l.: s.n.], 2013. p. 746–751. Citado na página 45.

MILLER, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, ACM, v. 38, n. 11, p. 39–41, 1995. Citado 4 vezes nas páginas 15, 19, 33 e 35.

MOURA, M. A. *O Discurso do Ódio em Redes Sociais*. [S.l.]: Lura Editorial (Lura Editoração Eletrônica LTDA-ME), 2016. Citado na página 58.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814. Citado na página 26.

PAULUS, R.; XIONG, C.; SOCHER, R. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017. Citado na página 14.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543. Citado na página 63.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *EMNLP*. [S.l.: s.n.], 2014. v. 14, p. 1532–1543. Citado 2 vezes nas páginas 15 e 32.

PLACING Search in Context: The Concept Revisited. *ACM Trans. Inf. Syst.*, ACM, New York, NY, USA, v. 20, n. 1, p. 116–131, jan. 2002. ISSN 1046-8188. Disponível em: <http://doi.acm.org/10.1145/503104.503110>. Citado na página 44.

RADINSKY, K. et al. A word at a time: Computing word relatedness using temporal semantic analysis. In: *Proceedings of the 20th International Conference on World Wide Web*. New York, NY, USA: ACM, 2011. (WWW '11), p. 337–346. ISBN 978-1-4503-0632-4. Disponível em: <http://doi.acm.org/10.1145/1963405.1963455>. Citado na página 44.

RUBENSTEIN, H.; GOODENOUGH, J. B. Contextual correlates of synonymy. *Commun. ACM*, ACM, New York, NY, USA, v. 8, n. 10, p. 627–633, out. 1965. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/365628.365657>. Citado na página 44.

RUMELHART, D. David e. rumelhart, geoffrey e. hinton, and ronald j. williams. *Nature*, v. 323, p. 533–536, 1986. Citado na página 14.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, Springer, v. 115, n. 3, p. 211–252, 2015. Citado na página 18.

SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2010. p. 92–101. Citado na página 24.

SCHMIDT, A.; WIEGAND, M. A survey on hate speech detection using natural language processing. In: *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain*. [S.l.: s.n.], 2017. p. 1–10. Citado na página 59.

SCHÜTZE, H.; MANNING, C. D.; RAGHAVAN, P. *Introduction to information retrieval*. [S.l.]: Cambridge University Press, 2008. v. 39. Citado na página 46.

SERMANET, P.; CHINTALA, S.; LECUN, Y. Convolutional neural networks applied to house numbers digit classification. In: *International Conference on Pattern Recognition (ICPR 2012)*. [S.l.: s.n.], 2012. Citado na página 24.

SMIT, P. et al. Morfessor 2.0: Toolkit for statistical morphological segmentation. In: AALTO UNIVERSITY. *The 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 26-30, 2014*. [S.l.], 2014. Citado 2 vezes nas páginas 38 e 40.

SOCHER, R. et al. Reasoning with neural tensor networks for knowledge base completion. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 926–934. Citado na página 33.

SPANGLER, S. et al. Automated hypothesis generation based on mining scientific literature. In: ACM. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2014. p. 1877–1886. Citado na página 18.

TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 48th annual meeting of the association for computational linguistics*. [S.l.], 2010. p. 384–394. Citado na página 21.

WANG, H.-Y.; MA, W.-Y. Integrating semantic knowledge into lexical embeddings based on information content measurement. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. [S.l.: s.n.], 2017. v. 2, p. 509–515. Citado na página 34.

WU, Y. 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 14.

XIONG, C.; ZHONG, V.; SOCHER, R. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016. Citado na página 14.

XU, C. et al. Rc-net: A general framework for incorporating knowledge into word representations. In: ACM. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. [S.l.], 2014. p. 1219–1228. Citado na página 34.

YU, M.; DREDZE, M. Improving lexical embeddings with semantic knowledge. In: *ACL (2)*. [S.l.: s.n.], 2014. p. 545–550. Citado na página 33.