UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Assistência bibliográfica durante a escrita de textos científicos: uma abordagem com modelos de linguagem pré-treinados

Trabalho de Conclusão de Curso

Demetrius Silva de Santana

São Cristóvão – Sergipe

2020

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Demetrius Silva de Santana

Assistência bibliográfica durante a escrita de textos científicos: uma abordagem com modelos de linguagem pré-treinados

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Hendrik Teixeira Macedo
Coorientador: Flávio Arthur Oliveira Santos

São Cristóvão – Sergipe

2020

Resumo

A produção científica inclui a geração de artefatos textuais, como artigos científicos e projetos de pesquisa. A escrita científica, por sua vez, apresenta desafios próprios. Um deles é lidar com a sobrecarga de informação na literatura, que dificulta a contextualização de um novo documento com as informações mais relevantes da área de pesquisa. Diante desse desafio, neste trabalho são investigadas formas de se representar vetorialmente sentenças e parágrafos de textos científicos, com o propósito de recomendar conteúdo relevante durante sua escrita. Por varredura automatizada das bases de dados das editoras Springer e Elsevier, foi construído um espelho para acesso local de artigos científicos, do qual foi possível ser extraído corpus com 40 mil artigos de periódicos de Ciência da Computação. Modelos de linguagem pré-treinados foram usados para se obter vetorizações de palavras ou fragmentos de palavras para uma amostra de 1605 artigos. A partir dessas representações, métodos de agregação para gerar representações de sentenças e de parágrafos foram investigados. Estratégias de vetorização para os elementos textuais foram avaliadas em dois aspectos. Primeiramente, na capacidade de refletir a agregação de parágrafos dentro das seções dos artigos já publicados. Em seguida, foram usadas fixadas para treinar uma rede recorrente de longa memória de curto prazo bidirecional (BiLSTM) na tarefa de determinar se um fragmento de texto e um resumo pertenciam a um mesmo artigo científico. Nas duas situações, atingiu melhor desempenho a vetorização com representações de codificador bidirecional por transformadores (BERT), na variante pré-treinada em corpus de textos científicos (SciBERT). Também nos dois cenários, a representação de sentenças pela média dos vetores pré-treinados, ponderada a partir da frequência do elemento, mostrou desempenho inferior quando comparada à média simples dos vetores após remoção das palavras de parada (*stopwords*). A representação de parágrafos a partir da codificação de uma sequência de sentenças por uma BiLSTM se mostrou superior quando comparada à simples média dos vetores de sentenças e, quando aplicada à introdução de um artigo científico no subconjunto de teste, foi capaz de retornar o resumo do próprio artigo dentre os 5% mais prováveis, em média. Uma demonstração qualitativa da recomendação de conteúdo que integra o resultado das duas abordagens é apresentada. A partir das estratégias investigadas, a assistência bibliográfica automatizada durante a produção de textos científicos se mostrou viável, podendo ser melhorada com a otimização de redes com BiLSTMs hierárquicas.

Palavras-chave: processamento de linguagem natural, redação científica, mineração de texto, vetorização de palavras, recomendação de conteúdo; assistência bibliográfica.

Abstract

Scientific output includes the production of textual artifacts, as scientific papers and research projects. Scientific writing, in turn, presents its own challenges. One of them is to face information overload in literature, which hinders contextualization of a new document with the most relevant information of its research area. In face of this difficulty, in this work we will investigate ways of embedding sentences and paragraphs of scientific texts, with the purpose of content recomendation along scientific writing. Through automated scanning of databases of publishers Springer and Elsevier, a local mirror was built for access of research papers, from which it was possible to obtain a corpus with 40 thousand papers from Computer Science journals. Pre-trained language models were used to obtain embedding for words or wordpieces for a sample of 1605 papers. Using these representations, aggregation methods for generating embeddings for sentences and paragraphs were investigated. Embedding strategies for these textual elements were evaluated in two aspects. Firstly, in their capacity of reflecting the aggregation of paragraphs within sections of already published papers. Secondly, they were used frozen to train a bidirectional long short-term memory (BiLSTM) neural network for the task of classifying wether a text fragment and an abstract belonged to the same paper. In both situations, a better performance was obtained with embeddings from bidirectional encoder representations from Transformers (BERT), specifically with a version pre-trained on a scientific corpus (SciBERT). Also in both circumstances, embeddings of sentences obtained by taking the mean of pre-trained word embeddings, weighted considering the word frequency, showed worse performance when compared to the mean of vectors after removing stopwords. Paragraph embeddings using the encoding of a sequence of sentences with a BiLSTM presented superior performance when compared to simple mean of sentence vectors and, when applied to the introduction of a scientific paper in test set, was able to return its own original paper within the 5% most likely ones, on average. A qualitative demonstration of content recommendation which integrates the result of both analyses is presented. Considering the studied strategies, automated bibligraphical assistance during production of scientific texts was feasible, with potential for improvement with more optimized hierarchical BiLSTMs.

Keywords: natural language processing, scientific writing, text mining, word embeddings, content recommendation; bibliographical assistance.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BiLSTM	*Bidirectional long short-term memory*
BERT	*Bidirectional Encoder Representations from Transformers*
BOW	*Bag of Words*
CBOW	*Continuous Bag of Words*
ELMo	*Embeddings from a Language Model*
GloVe	*Global Vectors*
LSTM	*Long short-term memory*
PCA	Análise de Componentes Principais – de *Principal Component Analysis*
PLN	Processamento de Linguagem Natural
SDAE	Sequential Denoising Autoencoder

Sumário

1

Introdução

Muitos são os motivos que dificultam a publicação de um artigo científico. Por exemplo, a escrita na língua inglesa pode ser uma barreira para os pesquisadores que não a tem como língua nativa (MARTÍN et al., 2014). Outras razões editoriais também podem ser impeditivas ao aceite de manuscritos, tais como inadequação ao escopo do periódico ou à estrutura editorial e, em particular, a ausência de referências atualizadas que contextualizem o trabalho produzido (DERNTL, 2014), o que é agravado pela crescente literatura disponível (EVANS, 2008).

A publicação científica bate recordes de produção anualmente. Estima-se que o total de artigos científicos ultrapassou 50 milhões em 2009 (JINHA, 2010), mas a a coleção aumenta aceleradamente. Em 2018, periódicos revisados por pares em língua inglesa ultrapassaram os 33 mil e, em conjunto com os mais de 9 mil periódicos em outras línguas, publicam anualmente mais de 3 milhões de artigos. Esse último número cresce 4% ao ano, refletindo o acúmulo de pesquisadores, que agora estão entre 7 e 8 milhões a depender da definição (JOHNSON; WATKINSON; MABE, 2018).

Os números são estimulados pela cultura do "publicar ou perecer", nascida com as medidas de produtividade de cientistas usadas para determinar qual será seu acesso ao financiamento de pesquisa (FRITH, 2020). Essas métricas acabam influenciando adversamente o que está sendo avaliado, por exemplo com impactos negativos na reprodutibilidade dos achados publicados (SMALDINO; MCELREATH, 2016). A sobrecarga de informação afeta, além da qualidade dos trabalhos científicos, a saúde mental dos pesquisadores, que devem gerenciá-la (BAWDEN; ROBINSON, 2009).

Essa fartura de fontes dificulta a contextualização de um novo artigo ou projeto de pesquisa com as informações mais relevantes da área de pesquisa na qual esteja inserido. Evans (2008) associou à maior oferta de artigos a produção de material cientificamente raso. Segundo a análise desse autor, artigos de áreas de pesquisa com periódicos mais disponíveis citam menor variedade de periódicos e deixam de se contextualizar com os resultados clássicos da área.

Diante do problema de sobrecarga de informação na literatura apresentado acima, pode-se justificar a necessidade de uma ferramenta que analise uma quantidade massiva de textos científicos para a recomendação de conteúdo. Uma possibilidade de realizar tal recomendação é através de modelos de aprendizagem de máquina aplicados ao processamento de linguagem natural. A disponibilidade de modelos de linguagem pré-treinados de alta qualidade tem o potencial de auxiliar pesquisadores a enfrentar os problemas expostos, ao permitir buscas semânticas mais flexíveis em bases de dados (HASSAN et al., 2019).

Neste trabalho serão investigadas formas de se representar vetorialmente sentenças e parágrafos de textos científicos, com o propósito de recomendar conteúdo relevante durante sua escrita. Para isso, é construída uma base de artigos científicos publicados em periódicos de alto impacto. São avaliadas estratégias de recomendação em níveis variáveis de complexidade, considerando que, para ser colocado em prática, um serviço de recomendação de conteúdo deverá atentar ao balanço entre custo computacional e acurácia para permitir recomendações relevantes com um tempo de resposta aceitável.

O fato de que o espaço vetorial das representações pré-treinadas reflete a semântica das palavras permite abordages mais simples que verificam a proximidade entre elementos textuais através de métricas matemáticas computacionalmente pouco custosas. Por outro lado, dado que os artigos possuem resumos que descrevem todo seu conteúdo, torna-se possível uma abordagem mais elaborada com o treinamento de redes neurais recorrentes que associem trechos de texto científico (do corpo do artigo) a um conteúdo de referência (o resumo). Essa investigação permite que futuramente possam ser desenvolvidos serviços que auxiliem pesquisadores durante a escrita de textos científicos, avaliando manuscritos iniciais e sugerindo conteúdo relacionado.

1.1 Objetivos

1.1.1 Objetivo geral

Investigar modelos de vetorização de sentenças e de parágrafos e aplicá-los em base de textos de artigos científicos.

1.1.2 Objetivos específicos

Tomando como padrão de referência a estrutura de artigos científicos já publicados em periódicos revisados por pares:

a) Avaliar o desempenho de modelos de vetorização de sentenças e de parágrafos em medidas de agrupamento desses vetores;

b) Avaliar o desempenho de modelos de vetorização de sentenças e de parágrafos no treinamento de redes neurais para indicar trabalhos relacionados a trecho de texto científico.

2

Fundamentação Teórica

A área de Processamento de Linguagem Natural (PLN) demanda técnicas que representem componentes de um texto vetorialmente. A depender da técnica, a unidade a ser representada por esses vetores varia em escala; cada um deles pode representar desde um fragmento de palavra até um texto inteiro. Diversos modelos de linguagem foram desenvolvidos ao longo dos anos, fazendo uso de redução de dimensionalidade, de modelos probabilísticos ou de redes neurais (MARTIN; JURAFSKY, 2009). Para os modelos baseados em redes neurais, uma seleção dos considerados mais relevantes serão apresentados nas seções deste capítulo. O foco deste capítulo é a contribuição teórica dos modelos às técnicas de representação vetorial de unidades de texto.

Os modelos de interesse diferem entre si por variações na arquitetura de sua rede neural e na base de dados usada para treinamento. A base de treinamento é um fator importante para determinar o escopo de aplicação de um modelo pré-treinado. De forma geral, o conhecimento armazenado na rede será preferencialmente representativo do domínio dos dados de treino. Ainda assim, pode haver transferência de aprendizagem entre domínios distintos porém próximos (YOSINSKI et al., 2014). O conceito de transferência de aprendizagem será exposto em sua seção própria no Capítulo 3.

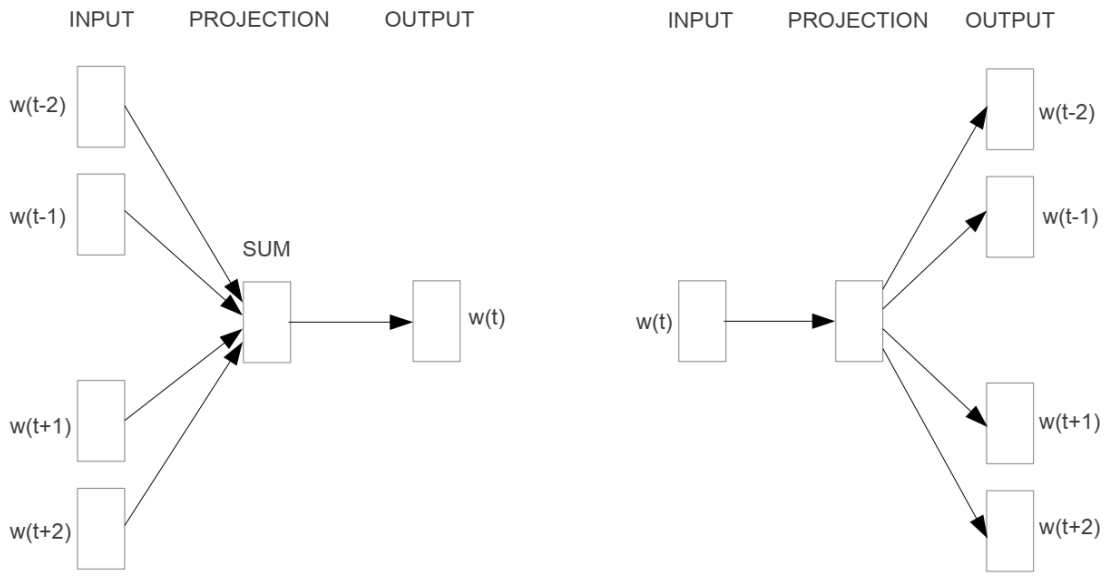
2.1 Word2vec

Desenvolvido em 2013 por pesquisadores da Google, o Word2vec foi projetado como uma extensão aperfeiçoada do modelo de base Skip-gram (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b). Nessa família de modelos, palavras são as unidades textuais representadas em um espaço vetorial. Os vetores provêm dos coeficientes resultantes após treinamento da rede neural associada ao modelo.

A arquitetura Skip-gram, que era uma de duas arquiteturas que haviam sido propostas por essa equipe anteriormente no mesmo ano, pode ser esquematizada como na Figura 1. À esquerda

está a arquitetura *Continuous Bag of Words* (CBOW), que é treinada com a tarefa de predizer a representação vetorial de uma palavra central a partir das representações das palavras ao seu redor. À direita está a arquitetura Skip-gram, que é treinada com a tarefa de predizer as representações de palavras vizinhas a partir da representação uma palavra central.

Figura 1 – Arquiteturas de modelos de linguagem CBOW e Skip-gram



Fonte: Mikolov et al. (2013a, p. 5)

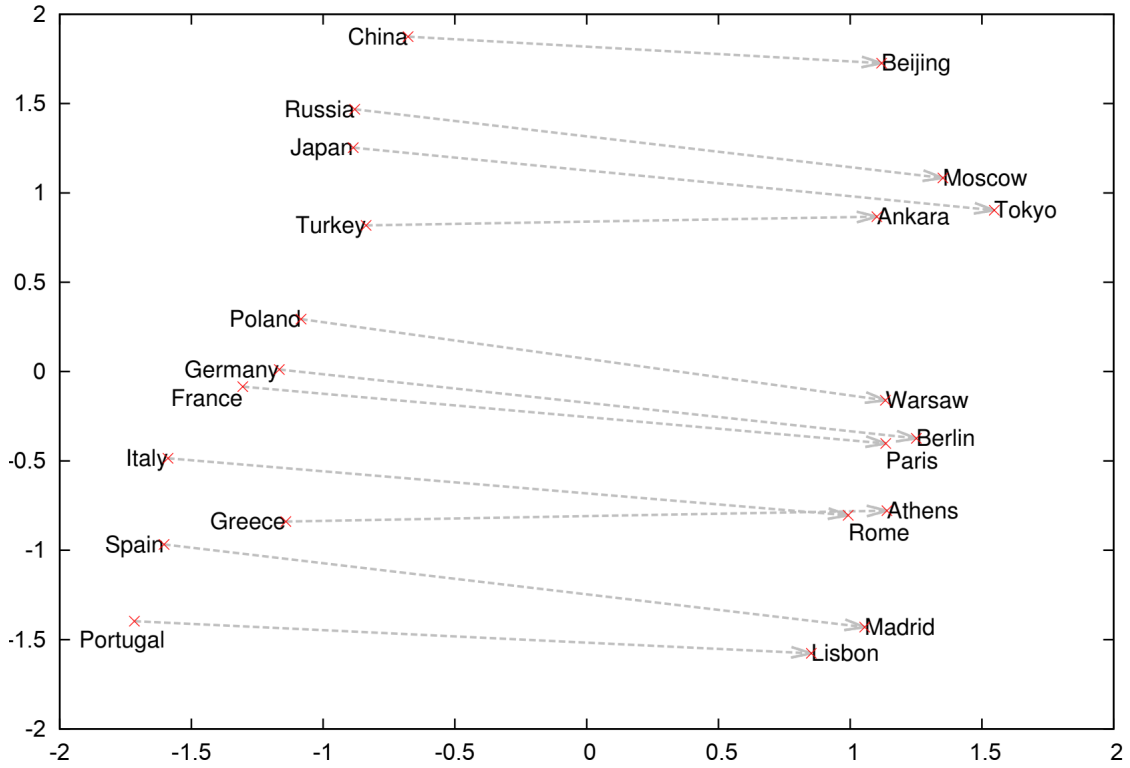
A ideia subjacente é fazer com que fossem próximas as representações vetoriais de palavras cujas vizinhanças nos textos fossem similares. Obteve-se como resultado para essa família de modelos, além dessa proximidade, a preservação de outros padrões linguísticos nas representações. Como exemplo, a adição de vetores conseguiu preservar alguns aspectos semânticos das palavras.

Um exemplo mais específico da preservação desses "eixos semânticos" é dado na Figura 2, reproduzida do trabalho original. Nela, são representados alguns dos vetores para países e suas respectivas capitais, após redução dimensional, com análise de componentes principais (PCA), de um espaço com 1000 dimensões para uma projeção com 2 componentes principais. Observa-se que o vetor que precisa ser somado à representação vetorial do nome de um país para se obter a representação vetorial do nome de sua capital é aproximadamente o mesmo: um vetor com cerca de 3 unidades de componente principal do eixo horizontal da figura[1].

O objetivo de treinamento para o modelo Skip-gram (e Word2vec) é encontrar representações de palavras que sejam úteis para predizer as palavras que cercam uma palavra central em uma sentença. Em linguagem matemática, dada uma sentença composta pela sequência de

[1] Há outra informação preservada no exemplo: a ordenação geográfica dos países é vagamente conservada enquanto se percorre o gráfico na direção do componente principal vertical.

Figura 2 – Alguns países e suas capitais projetados por PCA



Fonte: Mikolov et al. (2013b, p. 4)

palavras $w_1, w_2, \ldots, w_T$, o objetivo do modelo é maximizar o valor médio da log-probabilidade:

$$\frac{1}{T}\sum_{t=1}^{N}\sum_{\substack{-c\leq j\leq c\\ j\neq 0}} \log p\left(w_{t+j}|w_t\right) \tag{2.1}$$

onde c é o tamanho do contexto de treinamento. Valores grandes para c podem melhorar a acurácia do modelo, pois fazem com que o modelo considere um número maior de palavras como contexto, entretanto ao custo de aumentar o tempo de treinamento [2].

A probabilidade à qual o modelo Skip-gram se refere é definida como uma função *softmax*:

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})} \tag{2.2}$$

em que v_w e v'_w são as representações vetoriais de "entrada" e de "saída" de alguma palavra w, e W é o número de palavras do vocabulário. Essa formulação foi alterada por não ser prática em termos de custo computacional, que é proporcional ao tamanho do vocabulário W.

[2] Alguns modelos estendem a janela de contexto para toda a sentença ou todo o documento, como o modelo Sent2Vec, que é apresentado na seção 3.1 do próximo capítulo.

Seguindo a proposta de Morin e Bengio (2005), esse cálculo de probabilidade foi aproximado por uma versão hierárquica do *softmax*. A principal vantagem é avaliar apenas cerca de $\log_2(W)$ dos W nós de saída da rede neural. Para isso, é usada uma representação da camada de saída da rede como uma árvore binária com W palavras em suas folhas e, para cada um dos demais nós, uma representação explícita de probabilidades para se ir aos nós filhos. Ou seja, é possível percorrer a árvore aleatoriamente do nó raiz até as folhas, atribuindo-se assim probabilidades a cada uma das palavras.

Definida a estrutura da árvore binária, a fórmula para o *softmax* hierárquico é:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([n(w, j+1) = ch(n(w, j))] \cdot {v'_{n(w,j)}}^T v_{w_I}\right) \tag{2.3}$$

em que: $n(w, j)$ é uma função que retorna o j-ésimo nó do caminho que liga a raiz à palavra w; $L(w)$ é o comprimento desse caminho; $ch(n)$ é um filho arbitrário de um nó n; σ é a função sigmoide $\sigma(x) := 1/(1 + \exp(-x))$; e $[b]$ é uma função que retorna 1 quando a variável boleana b tem valor $true$ e -1, quando $false$.

Com essa função, o custo computacional de se calcular $\log p(w_O|w_I)$ e $\nabla \log p(w_O|w_I)$ torna-se proporcional ao comprimento $L(w_O)$ do caminho da árvore binária que liga a raiz a w_O.

Uma alternativa ao *softmax* hierárquico como função a ser maximizada é a amostragem negativa, uma variação de "estimação contrastiva de ruído" como aplicado em modelos de linguagem anteriormente por Mnih e Teh (2012). Substituem-se as ocorrências de $\log p(w_O|w_I)$ no objetivo do Skip-gram por:

$$\log \sigma\left({v'_{w_O}}^T v_{w_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma\left(-{v'_{w_i}}^T v_{w_I}\right)\right] \tag{2.4}$$

em que $\mathbb{E}_{w_i \sim P_n(w)}$ representa a probabilidade de se encontrar a palavra w_i numa distribuição "ruidosa" P_n para a palavra w. Com isso a tarefa se torna distinguir uma palavra alvo w_O de palavras sorteadas da distribuição ruidosa $P_n(w)$ usando regressão logística, onde existem k "amostras negativas" para cada exemplo de treino. Nesse cenário, a distribuição $P_n(w)$ passa a ser parâmetro livre do modelo Word2vec.

A última melhoria, a ser mencionada, trazida pelos modelos Word2vec é a realização do treinamento com subamostragem de palavras frequentes. Geralmente palavras frequentes, como artigos e pronomes, carregam menos da informação semântica de um texto que palavras mais raras. Essa subamostragem é especificada ao se descartar uma palavra w_i com a probabilidade

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.5}$$

em que $f(w_i)$ é a frequência da palavra w_i e t é uma variável de limiar arbitrária pequena.

2.2 GloVe

Pennington, Socher e Manning (2014) propuseram um modelo de linguagem que visava capturar diretamente as propriedades lineares obtidas indiretamente pelo modelo Word2Vec. Isso foi feito através de um modelo de mínimos quadrados que treinava considerando as entradas da matriz global de coocorrência de palavras, sendo portanto nomeado de *Global Vectors* (GloVe).

Os autores do modelo GloVe argumentaram que os modelos baseados em janelas de contexto, como o Word2Vec, tinham a desvantagem de não aproveitar toda a informação armazenada nas estatísticas de coocorrência das palavras.

A função de custo para o modelo foi proposta como:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \tag{2.6}$$

em que V é o tamanho do vocabulário; f é a função de pesos para os mínimos quadrados; X_{ij} são as entradas da matriz de coocorrências; w_i e $\tilde{w}_j$ são vetores de entrada e de saída determinados pelo modelo para as i-ésima e j-ésima palavras; e b_i e $\tilde{b}_j$ são as respectivas constantes de viés.

A função de peso f usada pelos autores desse modelo foi:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{se } x < x_{\max} \\ 1 & \text{caso contrário} \end{cases} \tag{2.7}$$

parametrizada com $x_{\max} = 100$ e $\alpha = 3/4$, determinados empiricamente. O propósito dessa função foi reduzir a influência das coocorrências de palavras muito frequentes no vocabulário, permitindo que as coocorrências menos comuns fossem melhor capturadas pelo modelo.

Dado que os vetores w e $\tilde{w}$ diferem apenas por conta das suas diferenças de inicialização aleatória, os dois vetores devem apresentar desempenho similar para a representação vetorial da palavra. Os autores do modelo então tomaram a representação da palavra como $w + \tilde{w}$, reduzindo sobreajuste e ruído capturados pelos vetores separadamente.

Os autores compararam os modelos Word2Vec e GloVe no desempenho de tarefas de analogia, observando para o último uma melhora de acurácia de 4,5% e 2,3% para as analogias semânticas e sintáticas, respectivamente, quando comparado ao primeiro.

2.3 fastText

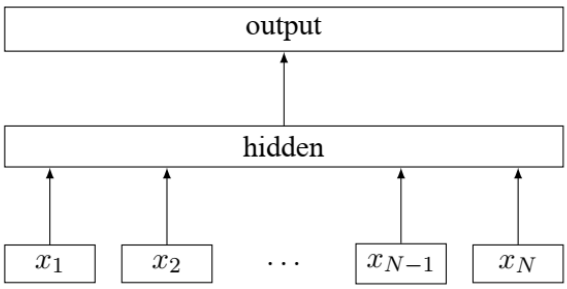
Em 2016, a equipe do Facebook AI Research propôs um novo modelo para classificação de textos, o fastText (JOULIN et al., 2016; BOJANOWSKI et al., 2017). Uma simplificação do Word2vec que pode ter impacto no modelo é não considerar a ordem das palavras da sentença. Ainda que alguma melhora possa ser obtida realizando-se uma maior amostragem ou maior ponderação das palavras mais próximas à central durante o treinamento (MIKOLOV et al., 2013a), grande parte da informação sobre a ordem das palavras na sentença é perdida.

Técnicas anteriores foram preservadas nessa família de modelos, como o já mencionado *softmax* hierárquico. Uma ideia agregada aos modelos fastText foi substituir a vetorização de palavras por vetorização de n-gramas de palavras para preservar alguma da informação de ordem nas sentenças (JOULIN et al., 2016), motivados por resultados de Wang e Manning (2012).

Os n-gramas são subsequências contíguas de n elementos de uma sequência dada (MARTIN; JURAFSKY, 2009) . A depender do valor n, recebem um nome próprio: bigramas são n-gramas com $n = 2$. Por exemplo, na frase "bigramas têm dois elementos", o conjunto de bigramas de palavras é {(bigramas, têm), (têm, dois), (dois, elementos)}.

O modelo a ser treinado é esquematizado na Figura 3. Nessa representação, durante o processo de treinamento em alguma tarefa, a sentença contém N características n-grama, das quais é tomada a média de suas vetorizações para ser usada como a variável de entrada na camada seguinte da rede. Esse valor médio também representa uma vetorização para a sentença.

Figura 3 – Arquitetura do modelo fastText



Fonte: Joulin et al. (2016, p. 2)

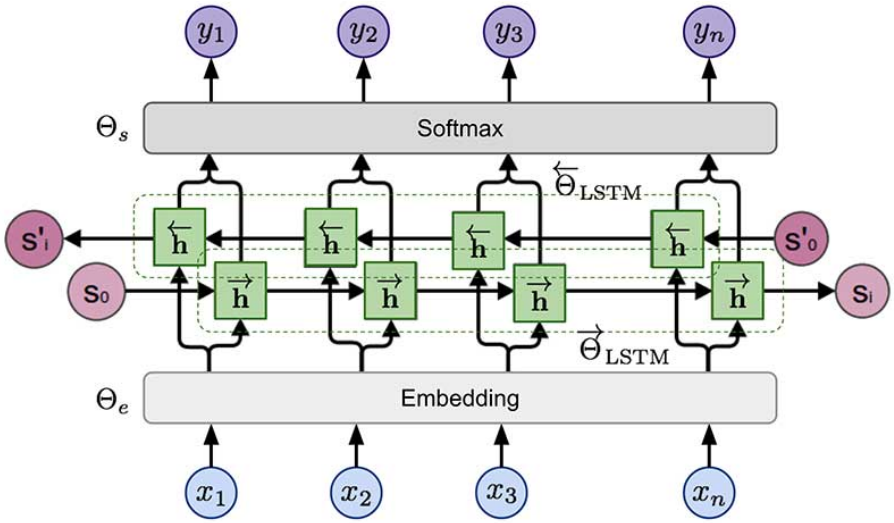
Inicialmente os modelos fastText fizeram apenas uso de n-gramas de palavras, mas logo em seguida passou-se a usar n-gramas de caracteres para se obter informação da estrutura morfológica de palavras (BOJANOWSKI et al., 2017). Uma característica adicional desse subconjunto de modelos é ser capaz de vetorizar palavras que não estavam presentes na base de treinamento original, mas são a combinação de morfemas conhecidos.

2.4 ELMo

A família de modelos fastText permite que sentenças sejam vetorizadas preservando alguma informação determinada pela ordem das palavras, ou que palavras sejam vetorizadas considerando-se sua morfologia representada indiretamente por *n*-gramas. Ainda assim, uma vez treinado o modelo, cada palavra possui uma representação independente do seu contexto. Diante dessas restrições de modelos anteriores e se baseando em um modelo supervisionado recente que treinava com a tarefa de tradução de textos (MCCANN et al., 2017), o modelo *Embeddings from a Language Model* (ELMo) buscou determinar um tipo de vetorização contextualizada refletindo: as características sintáticas e semânticas de uma palavra; e como essas características variam em diferentes contextos (PETERS et al., 2018).

A arquitetura dos modelos ELMo são baseadas em redes LSTM bidirecionais. Redes LSTM são redes recorrentes e, em contraste a redes *feedforward*, têm conexões de retroalimentação. Isso permite que a rede seja treinada com entradas de tamanho variável e reduz a quantidade de coeficientes que terão de ser aprendidos pela rede. Os detalhes da estrutura interna de uma célula de rede LSTM são variados, mas a estrutura comum permite à rede aprender mecanismos de "esquecimento" durante a avaliação de uma sequência de entrada, o que torna o seu treinamento mais rápido (HOCHREITER; SCHMIDHUBER, 1997).

Figura 4 – Arquitetura de uma camada do modelo ELMo



Fonte: Weng (2019)

Um esquema simplificado do modelo ELMo é apresentado na Figura 4 (WENG, 2019). Nele, as variáveis x_i representam as palavras de uma sentença que servem de entrada para a camada de vetorização (*Embedding*). O resultado da vetorização passa então por uma camada com duas redes LSTM em paralelo, $\overrightarrow{h}$ e $\overleftarrow{h}$, cada uma em um sentido do texto. As variáveis de saída das duas redes LSTM são concatenadas em uma variável de saída, y_i, que passa de

maneira similar por mais três camadas análogas, produzindo quatro valores de saída para cada x_i da entrada, um para cada camada de LSTM bidirecional.

Os valores de saída das quatro camadas são combinados em uma soma ponderada comum, e então esse valor é usado no treinamento de uma tarefa específica. Esses quatro vetores de coeficientes podem ser treinados conjuntamente com toda a rede, aprendendo-se também as vetorizações das palavras nesse processo, ou uma rede pré-treinada pode ser usada e apenas os quatro vetores de peso serem ajustados (PETERS et al., 2018).

2.5 BERT

BERT (*Bidirectional Encoder Representations from Transformers*) é o estado da arte para modelos de representação de linguagem (DEVLIN et al., 2018). Ele se destaca em relação aos modelos anteriores por combinar em sua arquitetura duas características importantes, antes disponíveis apenas separadamente.

Primeiramente, em contraste ao modelo ELMo já exposto, BERT é capaz de combinar o contexto de uma palavra nas duas direções em uma mesma representação profunda. ELMo traz representações calculadas nas duas direções, mas Devlin et al. (2018) argumentam que é uma representação rasa pois é a mera concatenação de representações unidirecionais.

Em segundo lugar, BERT faz uso de treinamento profundo bidirecional, ao contrário do trabalho apresentado por Radford et al. (2018). Como no modelo BERT, esses pesquisadores também pré-treinaram um modelo com arquitetura Transformer, porém considerando apenas a sequência unidirecional de palavras no sentido esquerda-direita.

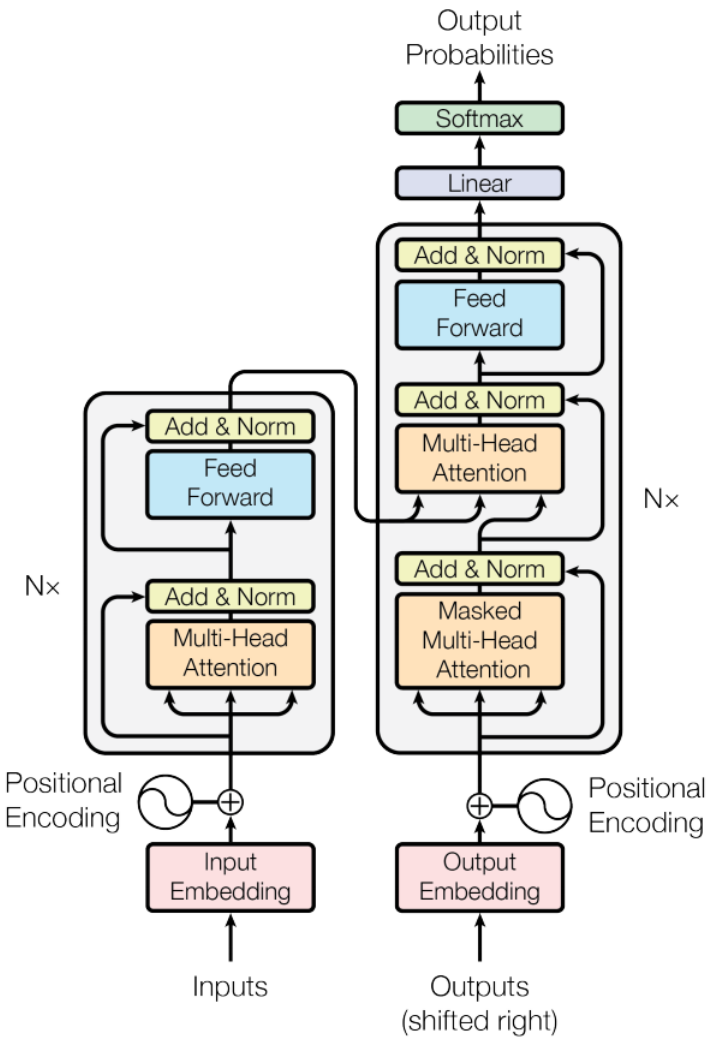
A arquitetura Transformer será apresentada como descrito por Vaswani et al. (2017).

Há duas pilhas de camadas no modelo: o codificador e o decodificador. O primeiro mapeia uma sequência de entrada $(x_1, ..., x_n)$ para uma sequência de representações $z = (z_1, ..., z_n)$. Então, dado z, o decodificador gera uma sequência $(y_1, ..., y_n)$ de símbolos de um elemento por vez. Em cada passo o modelo é auto-regressivo, ou seja, consome a sequência previamente gerada de símbolos como uma entrada adicional enquanto gera a próxima.

O codificador é composto de N camadas idênticas. Cada camada tem duas subcamadas. A primeira delas é um mecanismo de auto-atenção multifoco e a segunda é uma simples rede *feedforward* completamente conectada. O decodificador também é composto por N camadas idênticas. Porém, além das duas camadas do codificador, existe uma terceira camada que realiza o mecanismo de atenção multifoco diretamente sobre a saída do codificador. Esse esquema básico para uma das N camadas é apresentado na Figura 5.

O mecanismo de atenção usado é obtido através de produtos escalares. Esse mecanismo pode ser conceitualizado como um mapeamento de cada entrada x_i para um conjunto de três vetores, nomeados de vetor questão q, vetor chave k e vetor valor v. Os vetores questão e chave

Figura 5 – Arquitetura de uma camada Transformer



Fonte: Vaswani et al. (2017, p. 3)

devem ter a mesma dimensão d_k, enquanto que a dimensão d_v do vetor valor pode diferir. Então, dada uma sequência de n ternas de vetores (q_i, k_i, v_i) associada a uma sequência de entrada, o vetor de de atenção a_j para a j-ésima entrada pode ser calculador como a soma dos v_i ponderados pelo *softmax* do produto escalar $q_j^T k_i$ normalizado com $\sqrt{d_k}$.

Na prática, todos os vetores de atenção são obtidos numa matriz A calculada com operações matriciais a partir das respectivas matrizes Q, K, V:

$$A = A(Q, K, V) := \text{softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right) V \tag{2.8}$$

Na descrição do parágrafo anterior, cada um dos q_i, k_i, v_i é obtido a partir de projeções lineares de x_i, sejam elas $F^{(q)}(x_i), F^{(k)}(x_i), F^{(v)}(x_i)$, respectivamente, cujas matrizes associadas são aprendidas durante o treinamento do modelo. Então $F = \left(F^{(q)}, F^{(k)}, F^{(v)}\right)$ determina um foco

de atenção A:

$$A_F(X) := \text{softmax}\left(\frac{\left(F^{(q)}(X)\right)^T F^{(k)}(X)}{\sqrt{d_k}}\right) F^{(v)}(X) \tag{2.9}$$

O vetor final de atenção AMF é denominado de multifoco pois faz uso de vários focos de atenção F_m, aprendidos com variações dos valores das dimensões d_k, d_v associadas às transformações lineares dos focos. Então a atenção com h focos pode ser expressa como a concatenação:

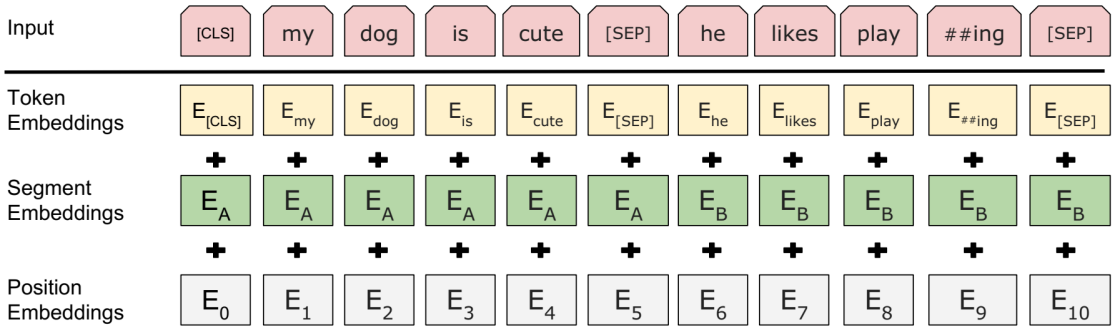
$$AMF := \left(A_{F_1}^T \; A_{F_2}^T \; \ldots \; A_{F_h}^T\right)^T \tag{2.10}$$

Além do mecanismo de atenção multifoco, a arquitetura Transformer faz uso de codificação posicional explícita após a camada de vetorização do elemento de entrada (Figura 5). Isso adiciona na entrada informação sobre a ordem dos elementos da sequência, além daquela obtida indiretamente pelo restante do Transformer. Na descrição original de Vaswani et al. (2017), a informação posicional IP foi adicionada por:

$$IP(pos,i) = \begin{cases} sin\left(\frac{pos}{10^{8i/d}}\right), & \text{se } i \text{ par} \\ cos\left(\frac{pos}{10^{8(i-1)/d}}\right), & \text{se } i \text{ ímpar} \end{cases} \tag{2.11}$$

em que pos é a posição da palavra na sequência, i é a entrada do vetor e d é a dimensão do modelo.

Figura 6 – Entrada da primeira camada Transformer da arquitetura BERT



Fonte: Devlin et al. (2018, p. 5)

O modelo BERT foi pré-treinado com duas tarefas não-supervisionadas de propósito geral. A ideia é que o modelo obtido desse treinamento inicial pudesse ser refinado em outras tarefas específicas. As duas tarefas de base são: predizer um subconjunto de palavras aleatoriamente mascaradas em uma sentença; predizer se, dadas duas sentenças, a segunda é consecutiva à primeira em aparição no texto.

A primeira tarefa, com mascaramento, não seria possível caso apenas um elemento fosse removido da sentença. Devlin et al. (2018) argumentam que nessa situação o treinamento do modelo seria trivializado, pois a palavra seria "capaz de se ver" no fluxo de entrada. Então a solução adotada é mascarar uma porcentagem das palavras da sentença aleatoriamente.

Para a segunda tarefa, adiciona-se à vetorização de entrada nas camadas Transformers uma marcação a cada palavra da sequência, que indica se ela pertence à sentença *A* ou à sentença *B*, como ilustrado na Figura 6. Então o modelo é treinado para a tarefa de determinar se a sentença *B* segue a senteça *A* ou não. Em 50% dos casos de treino, a senteça *B* de fato sucede diretamente a senteça *A*. No restante, é escolhida uma senteça aleatória da base de dados. Mesmo que simples, foi observado que esse treino é capaz de melhorar o desempenho do modelo em outras tarefas.

3

Trabalhos Relacionados

Sistemas de recomendação de artigos científicos visam indicar documentos que estejam dentro dos interesses de pesquisadores. Na sua forma mais simples, isso é feito através de palavras-chave. Muitas formas de recomendação já foram exploradas, em geral após a construção de um perfil de interesse do pesquisador. Os algoritmos clássicos disponíveis para realizar a recomendação são variados, incluindo filtragem baseada em conteúdo, filtragem colaborativa, métodos baseados em grafos e métodos híbridos (BAI et al., 2019).

Independente do método específico empregado para a recomendação, são necessárias formas de se representar os itens a serem buscados e o perfil do pesquisador. As representações de conteúdo derivadas de palavras-chave nem sempre são adequadas, pois o pesquisador pode não ser capaz de sumarizar bem suas necessidades, buscando por palavras-chave inadequadas.

Tendo isso em vista, Hassan (2017) propôs o uso de métodos de aprendizagem profunda com redes neurais recorrentes para se representar *strings* de busca de artigos, fazendo uso de modelos de linguagem pré-treinados. O desenvolvimento dessa abordagem se mostrou promissor em um trabalho recente do grupo de pesquisa, retornando documentos que não seriam obtidos apenas com métodos probabilísticos clássicos (HASSAN et al., 2019). No presente trabalho, em vez de pequenas *strings* de busca, consideramos todo um manuscrito inicial para a recomendação de conteúdo, demandando-se formas de se obter *embeddings* para sentenças e parágrafos.

Para se representar vetorialmente senteças e parágrafos, duas abordagens são possíveis. Na primeira, usam-se modelos que nativamente realizam essa vetorização. Na segunda, as vetorizações contextualizadas de palavras dos modelos mais recentes são usadas e de alguma forma convertidas em uma vetorização para as agregações. Neste capítulo apresentamos alguns resultados de trabalhos relacionados dentro dessas duas vertentes possíveis. Optar pelos modelos de linguagem mais recentes, como ELMo e BERT, tem grande potencial de reúso de coeficientes dos modelos já treinados. Esses modelos são treinados em bases de dados não-supervisionadas extensas e potencialmente são capazes de extrair *embeddings* com qualidade alta para as palavras

do texto científico. Motivando-se em casos de sucesso de outras áreas de aprendizagem de máquina, o conceito de transferência de aprendizagem é detalhado na seção 3.3.

3.1 Vetorização de sentenças

Nesta subseção apresentaremos modelos de linguagem que nativamente visam o problema de encontrar *embeddings* para sentenças. Podem ser divididos em três tipos: (i) modelos não-supervisionados independentes da ordem das sentenças; (ii) modelos não-supervisionados dependentes da ordem das sentenças; e (iii) modelos supervisionados, ou seja, que demandam dados já classificados para treinamento.

Dentro do primeiro grupo de modelos, Hill, Cho e Korhonen (2016) propuseram um modelo autocodificador de sentenças. Dada uma sentença S, obtém-se uma versão corrompida N para a sentença, removendo-se alguma de suas palavras w com probabilidade p_o ou invertendo-se a ordem de bigramas com probabilidade p_x. Com isso é treinada uma rede codificadora-decodificadora baseada em arquitetura LSTM com o objetivo de predizer a sentença original S a partir da versão ruidosa N. O modelo treinado pode então codificar novas sequências em representações vetoriais. Esse modelo foi nomeado de SDAE (*Sequential Denoising Autoencoder*).

Dentro do grupo de modelos não-supervisionados que consideram a ordem das sentenças, está o FastSent (HILL; CHO; KORHONEN, 2016). O FastSent é baseado no SkipThought (KIROS et al., 2015), um modelo derivado do Skip-gram, porém orientado para a tarefa de predição, a partir de uma sentença central, da sentença imediatamente anterior e da imediatamente posterior. No SkipThought, a sentença é primeiramente codificada com uma variante de LSTM e então decodificada em duas sentenças-alvo.

Uma limitação do SkipThought é demandar muito tempo para o treinamento do modelo. Nesse sentido, o FastSent abandona a codificação das sentenças com redes recorrentes e adota a abordagem BOW (*bag of words*), isto é, desconsiderando a ordem das palavras. Mais formalmente, o modelo aprende codificações de entrada e de saída para para uma palavra w, respectivamente v_w e v'_w. Então, para um exemplo de treino com três sentenças consecutivas S_{i-1}, S_i, S_{i+1}, a senteça S_i é representada como a soma dos *embeddings* de entrada de suas palavras, $s_i = \sum_{w \in S_i} v_w$. A função alvo do treinamento então é dada por:

$$\sum_{w \in S_{i-1} \cup S_{i+1}} \phi(s_i, v'_w) \tag{3.1}$$

em que ϕ é a função *softmax*.

Uma variação desse modelo é o FastSent+AE (HILL; CHO; KORHONEN, 2016), em

que o modelo deve predizer as próprias palavras, alterando o objetivo para:

$$\sum_{w \in S_{i-1} \cup S_i \cup S_{i+1}} \phi(s_i, v'_w) \tag{3.2}$$

Um exemplo de modelo que visa embeddings de sentenças e faz uso de dados estruturados é o DictRep (HILL et al., 2016). Busca-se mapear definições de dicionários de palavras para *embeddings* pré-treinados dessas palavras. São usadas duas arquiteturas, BOW e LSTM com a possibilidade de se escolher usar um modelo de *embedding* pré-treinado para as palavras ou também aprendê-lo durante o treinamento do modelo.

Um modelo mais recente é o Sent2Vec, que visa aprender vetorizações universais para sentenças (PAGLIARDINI; GUPTA; JAGGI, 2017). Conceitualmente, o modelo é uma extensão do modelo CBOW, já mencionado na seção 2.1 (MIKOLOV et al., 2013a). No modelo CBOW, k é a quantidade de palavras usadas como contexto para a palavra central. Já o modelo Sent2Vec usa toda a sentença como contexto. Formalmente, aprende-se duas vetorizações para cada palavra w, de entrada v_w e de saída v'_w, como nos trabalhos anteriores. O *embedding* para sentenças é definido com uma estratégia BOW para unigramas e bigramas de palavras. De resto, estratégias usadas no Word2vec são usadas para acelerar o treinamento, como amostragem negativa no lugar da função objetivo e subamostragem de palavras frequentes. Os resultados obtidos pelos autores em termos de acurácia superam o FastSent e são similares ao SkipThought, com a vantagem de tempo reduzido para treino do modelo.

3.2 Vetorização de um conjunto de vetores

O sucesso de redes neurais para se obter *embeddings* de palavras motivou o desenvolvimento de modelos complexos e especializados, visando alcançar representações para segmentos de texto mais longos, como sentenças e parágrafos. Ainda assim, Wieting et al. (2015) demonstraram, em uma tarefa supervisionada, que métodos complicados podem ser facilmente superados por regressões lineares simples ou por retreinamento breve de vetorizações de palavras, especialmente em situações fora do domínio original de treino dos modelos originais, ou seja, em cenários de transferência de aprendizagem.

Arora, Liang e Ma (2016) fornecem a seguinte receita para se obter um ótimo referencial de comparação para *embeddings* de sentenças: escolha uma método de vetorização de palavras popular obtido de maneira não-supervisionada numa base de dados grande; vetorize a sentença como uma média ponderada das representações das palavras; e então a modifique levemente usando, por exemplo, PCA para remover a projeção do vetor médio sobre o primeiro componente principal do conjunto. Esse método foi capaz de superar os resultados de Wieting et al. (2015).

Além disso, o método de Arora, Liang e Ma (2016) é capaz de ser aplicado recursivamente como, por exemplo, transformando representações de sentenças em representações de parágrafos.

Ou seja, serve de protótipo para um método genérico que determina representações vetoriais para conjuntos de vetores linguísticos em diferentes escalas.

3.3 Transferência de aprendizagem

Um fenômeno comum foi observado em redes convolucionais profundas treinadas com imagens: as características aprendidas pelas primeiras camadas são consistentemente equivalentes a filtros que extraem a direcionalidade de bordos nas imagens. Isso é tão comum que a obtenção de outro resultado num conjunto de imagens naturais levanta suspeita para escolha inadequada de parâmetros do modelo (YOSINSKI et al., 2014). Esse fenômeno ocorre não apenas com bases de dados diferentes, mas também com objetivos de treinamento distintos e em tarefas supervisionadas ou não-supervisionadas (LE et al., 2011; KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Observou-se que essas características das camadas iniciais aceleravam o treinamento de um modelo para uma nova tarefa se ele já estivesse pré-treinado com outra tarefa ou base de dados. Esse fenômeno, de reaproveitamento de características aprendidas, é chamado de transferência de aprendizagem. Quando a base de dados alvo é significativamente menor que uma base de referência, a transferência de aprendizagem pode ser uma ferramenta relevante para se treinar uma rede grande sem se arriscar sobreajuste (*overfitting*) do modelo (YOSINSKI et al., 2014).

Como exemplo no caso de vetorizações em PLN, Zhang, Wilson e Mihalcea (2018) investigaram a transferência de aprendizagem na tarefa de determinar a similaridade semântica entre sentenças. Os resultados obtidos fixando-se os *embeddings* de palavras (vetores congelados, *frozen embeddings*) ou permitindo-se que fossem atualizados durante o treino (ajuste fino, *fine-tuning*) foram similares nos diversos codificadores de sentenças e modelos de base avaliados.

Houlsby et al. (2019) compararam os dois cenários de transferência de aprendizagem, isto é, fixando-se ou não os vetores de palavras, em 26 tarefas de classificação de texto. O reúso total dos *embeddings* de fragmentos de palavras da arquitetura BERT manteve-se a no máximo 0,4% do desempenho de quando feito o ajuste fino de toda a rede, mesmo adicionando apenas 3,6% de parâmetros à rede. Os autores argumentam que manter os parâmetros originais do codificador de palavras é vantajoso pois permite que a mesma rede neural seja compartilhada em diversas tarefas distintas, que diferem na arquitetura apenas nas camadas finais do classificador.

4

Material e Métodos

Para cumprir os objetivos específicos deste trabalho, três elementos são necessários: (i) uma base de textos de artigos científicos; (ii) uma medida para a avaliação da qualidade de agrupamentos de vetores; e (iii) uma arquitetura de rede neural com uma tarefa para avaliar a qualidade da representação vetorial fixada do elemento textual. Cada um deles é descrito a seguir.

4.1 Coleta e processamento de artigos científicos

Construiu-se uma ferramenta para varredura automatizada das bases de dados de artigos de editoras de textos científicos, conforme descrito nos parágrafos seguintes. A ferramenta é capaz de obter artigos em formato XHTML e de convertê-los a um JSON com as sentenças pré-processadas. Foi desenvolvida em linguagem Python com *framework* Django.

Inicialmente, foram coletados periódicos acessando-se a plataforma Sucupira da CAPES (<https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/veiculoPublicacaoQualis/listaConsultaGeralPeriodicos.jsf>). Foram selecionados os periódicos com classificação A1 ou A2 no evento de classificação do último período disponível (quadriênio 2013-2016) para a área de avaliação de Ciência da Computação. Foram mantidos apenas periódicos associados às editoras Springer ou Elsevier, por conveniência de acesso às edições de forma automatizada.

Para cada periódico da lista, através do portal Periódicos CAPES (<https://www.periodicos.capes.gov.br/>) foi descoberta a URL que continha a lista de edições já publicadas e o localizador para acesso. Daí, para cada edição, a lista de artigos foi descoberta e, para cada artigo, o XHTML foi obtido e armazenado localmente.

Para cada XHTML coletado, foi extraído um JSON com o artigo já processado como uma lista de seções. Cada seção foi obtida como uma lista de parágrafos e cada parágrafo, como uma lista de sentenças. Caso uma seção do artigo contivesse subseções, estas foram desconsideradas, e os seus parágrafos foram adicionados mantendo-se a ordem original do artigo.

Durante o processamento do XHTML, foram inicialmente removidas todas as tabelas, figuras e lista de referências bibliográficas. Quando foi possível identificar as seções de agradecimetos, conflito de interesse e apêndices, estes também foram removidos.

O resumo do artigo foi sempre mantido como o primeiro elemento na lista de seções, ou seja, como uma seção com um único parágrafo correspondendo ao próprio resumo. O resumo foi determinado como o primeiro parágrafo seguindo o texto *Abstract* em *tags div* ou *section* com atributo *class* valorado como *abstract* ou *Abstract*.

Os parágrafos foram processados quando identificada a *tag p*, ou *div* com atributo *class* valorado como *Para*. Equações sinalizadas com XHTML que ocupavam todo um parágrafo foram inteiramente removidas, e equações *inline* foram transformadas na *string "Equation"*. Dos parágrafos buscou-se remover:

- textos entre parênteses;
- referências a seções, capítulos e figuras dentro do próprio texto;
- pontuação de valores numéricos;
- as abreviações *e.g., i.e., s.t., i.i.d, et al*;
- vírgulas, aspas e colchetes.

Por fim, após essas manipulações, as sentenças foram definidas quebrando-se o texto restante do parágrafo nos pontos, pontos e vírgulas, dois pontos, e interrogações.

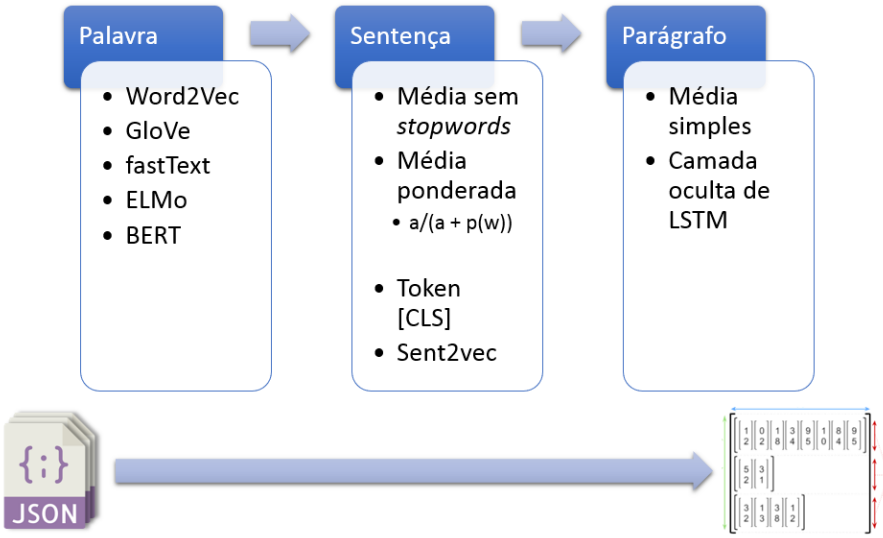
4.2 Estratégias de vetorização de elementos textuais

Um esquema geral do desenho experimental usado para se determinar vetorizações para os elementos textuais é apresentado na Figura 7 e descrito nos próximos parágrafos.

Inicialmente, foram usados modelos de linguagem pré-treinados para se obter as *word embeddings*. Os modelos de linguagem usados são apresentados na Tabela 1. Na mesma tabela, para cada modelo são apresentadas a origem das implementações e quais as fontes dos parâmetros pré-treinados. A obtenção dos *embeddings* foi realizada no ambiente interativo Jupyter, executado na nuvem pelo serviço Google Colaboratory.

A partir dos vetores pré-treinados de palavras ou fragmentos de palavras, foram obtidos os vetores para as senteças. Três estratégias foram consideradas: (i) o vetor referente ao primeiro *token* ([CLS]) quando na arquitetura BERT; (ii) a média simples dos vetores após a remoção dos referentes às *stopwords*; (iii) a média ponderada dos vetores tomando como peso para cada vetor v_w de uma palavra ou fragmento w: $a/(a+p(w))$, com $a = 10^{-3.5}$ e $p(w)$ estimado pela frequência

Figura 7 – Desenho experimental usado para se obter as vetorizações dos elementos textuais



Fonte: Próprio autor.

do *token* no documento (ARORA; LIANG; MA, 2016). Além disso, o modelo Sent2Vec também foi usado para se obter *embeddings* para sentenças.

A partir dos vetores de sentenças, duas estratégais foram consideradas para se obter os vetores de parágrafos. Na primeira, a simples média dos vetores de sentença foi tomada como representação do vetor de parágrafo. Na segunda, a representação a partir de uma BiLSTM foi usada, conforme melhor detalhado na seção 4.5.

4.3 Escore Silhouette e índice Rand adaptado

Artigos científicos são divididos em seções. Uma das nossas hipóteses de trabalho é que boas vetorizações para sentenças e parágrafos do artigo científico serão capazes de agrupar os elementos de uma mesma seção eficientemente. Para avaliar esse agrupamento entre diferentes estratégias de vetorização, podem ser usados o escore Silhouette e um índice Rand adaptado (SHRESTHA; JACQUIN; DAILLE, 2012; LAYTON; WATTERS; DAZELEY, 2013).

O escore Silhouette é uma medida específica para tarefas de agrupamento. Define-se distância média do vetor y_i aos elementos de um conjunto C do agrupamento:

$$D(y_i, C) \;:=\; (|C \setminus \{y_i\}|)^{-1} \sum_{y_j \in C} d(y_i, y_j) \tag{4.1}$$

Tabela 1 – Modelos de linguagem pré-treinados avaliados

Arquitetura	**Implementação e parâmetros pré-treinados**
Word2Vec	gensim.models.keyedvectors
	https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
GloVe	gensim.models.keyedvectors
	http://nlp.stanford.edu/data/glove.6B.zip
fastText	gensim.models.fasttext
	https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.en.300.bin.gz
Sent2vec	https://github.com/epfml/sent2vec
	https://drive.google.com/uc?confirm=ucAa&id=0B6VhzidiLvjSaER5YkJUdWdPWU0
ELMo	tensorflow_hub.Module
	https://tfhub.dev/google/elmo/2
BERT	https://github.com/google-research/bert
	https://s3-us-west-2.amazonaws.com/ai2-s2-research/scibert/tensorflow_models/scibert_scivocab_uncased.tar.gz

Fonte: Próprio autor.

que permite obter dois valores intermediários para o cálculo do escore:

$$\begin{aligned} a_i &:= D(y_i, C), C \ni y_i \\ b_i &:= \min_{C \not\ni y_i} D(y_i, C) \end{aligned} \tag{4.2}$$

Com esses valores, o escore Silhouete para o vetor y_i é definido como:

$$s_i := \begin{cases} \frac{b_i - a_i}{max\{a_i, b_i\}}, & \text{se } |C| > 1, C \ni y_i, \\ 0, & \text{caso contrário.} \end{cases} \tag{4.3}$$

Nessas últimas equações, $|C|$ é quantidade de elementos de um conjunto C. Escores Silhouette próximos a -1 sinalizam agrupamento insatisfatório e próximos a 1, satisfatório.

Para o caso de o escore Silhouette não se apresentar em faixa de valores satisfatória, a investigação da existência de outros agrupamentos nos conjuntos de vetores obtidos foi avaliada

por reamostragem estatística (*bootstrap*), com a geração aleatória por artigo de 100 novos conjuntos de agrupamentos, mantendo agrupamentos com o tamanho original das seções.

Outra medida da qualidade dos agrupamentos é o índice Rand adaptado. Na versão usada do índice, para o vetor de cada parágrafo, foi verificado se o parágrafo cujo vetor era o mais próximo em distância ℓ_1 pertencia à mesma seção do parágrafo de referência. O índice foi então determinado pela razão entre a quantidade de pares numa mesma seção pela quantidade total de pares de parágrafos. Valores próximos a 1 refletem um bom agrupamento de parágrafos.

4.4 Definição e treinamento das redes neurais

Três arquiteturas de redes neurais foram treinadas nos experimentos. As arquiteturas foram definidas em API Keras e treinadas com *backend* Tensorflow 2.1. Para todos os casos, 50 épocas foram usadas durante o treino. A função de perda para tarefas de classificação foi a entropia binária cruzada e para as tarefas de regressão, o erro absoluto médio. O algoritmo de otimização aplicado foi o *RMSprop*. As *seeds* de randomização dos pacotes *random*, *numpy* e *tensorflow*, bem como as *seeds* de todas as camadas da rede com inicialização de parâmetros foi igualmente atribuído antes de todos os treinos.

Para cada um dos 37 periódicos, buscou-se uma amostra aleatória de 50 artigos, que totalizariam 1850 artigos. Um dos periódicos tinha apenas o texto disponível para 20 artigos, que foram todos usados, reduzindo o total esperado de artigos para 1820. Além disso, foram removidos 215 artigos da amostra comum a todos os modelos por terem falhado em gerar *embeddings* para algum dos modelos. Em geral, isso ocorreu pela presença de caracteres matemáticos que não puderam ser removidos no pré-processamento dos XHTML por não estarem sinalizados com *tags*, gerando falhas no processamento do modelo ELMo. Isso resultou em uma amostra final de 1605 artigos, que foi dividida em conjuntos de treino, de validação e de teste, contendo respectivamente 1111, 337 e 157 artigos.

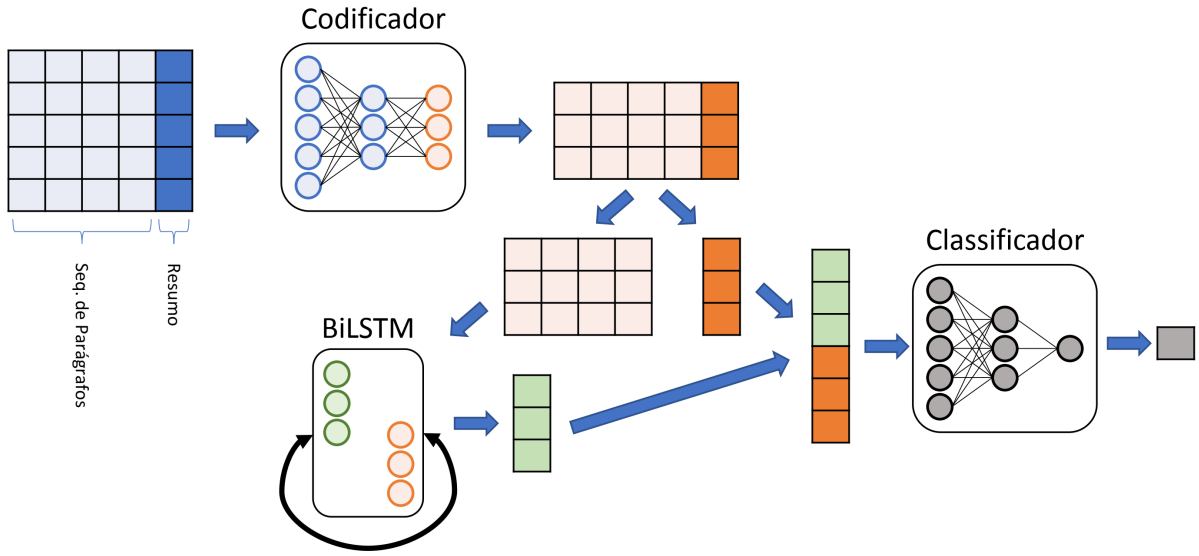
Cada *batch* de treino conteve 128 exemplos de sequência de parágrafos pareadas com o parágrafo do resumo verdadeiro do artigo. Para cada exemplo positivo, foi gerado um exemplo negativo pareando-se a sequência de parágrafos com o parágrafo de um resumo de outro artigo sorteado da base. As sequências de parágrafos foram consideradas nas seções com tamanho de 2 a 7 parágrafos, sendo passados os k primeiros parágrafos para os exemplos de treino, k sorteado uniformemente entre 2 e o tamanho da seção. No caso da arquitetura hierárquica, cada parágrafo foi passado como uma sequência de até 16 vetores de sentenças, fazendo com que cada *batch* nessa arquitetura correspondesse a um tensor de ordem 4.

4.5 Arquitetura das redes neurais treinadas

A primeira das arquiteturas, tomada como arquitetura de referência, é ilustrada na Figura 8. Essa rede neural foi composta por três submodelos: um codificador inicial, seguido por uma rede BiLSTM e, por fim, camadas *feedforward* de um classificador.

O codificador foi usado para reduzir a dimensionalidade dos vetores previamente calculados para os parágrafos (conforme as diferentes estratégias, vide seção 4.2), controlando a quantidade total de parâmetros da rede BiLSTM. A entrada para o codificador foi composta de uma sequência de vetores correspondentes ao texto científico avaliado (na figura, os vetores em azul claro) à qual foi concatenado um vetor que representou o resumo (na figura, em azul escuro). A sequência dos vetores reduzidos correspondentes ao texto (na figura, representados em laranja claro) foi codificada pela BiLSTM, que retornou uma codificação para a sequência de parágrafos (na figura, em verde claro). O classificador recebe o vetor da sequência e o vetor do resumo concatenados e deve determinar a probabilidade de pertencerem a um mesmo artigo.

Figura 8 – Esquema da arquitetura de classificação para entrada com vetores de parágrafos



Fonte: Próprio autor.

Para essa primeira arquitetura e para uma estratégia de vetorização de parágrafos com um modelo pré-treinado específico (BERT), a Tabela 2 apresenta detalhes da quantidade de parâmetros e das camadas da rede, seguindo o padrão de sumarização da API Keras. Para os demais modelos pré-treinados, há variações apenas na quantidade de parâmetros da primeira camada densa, devido à diferença de dimensionalidade entre os modelos.

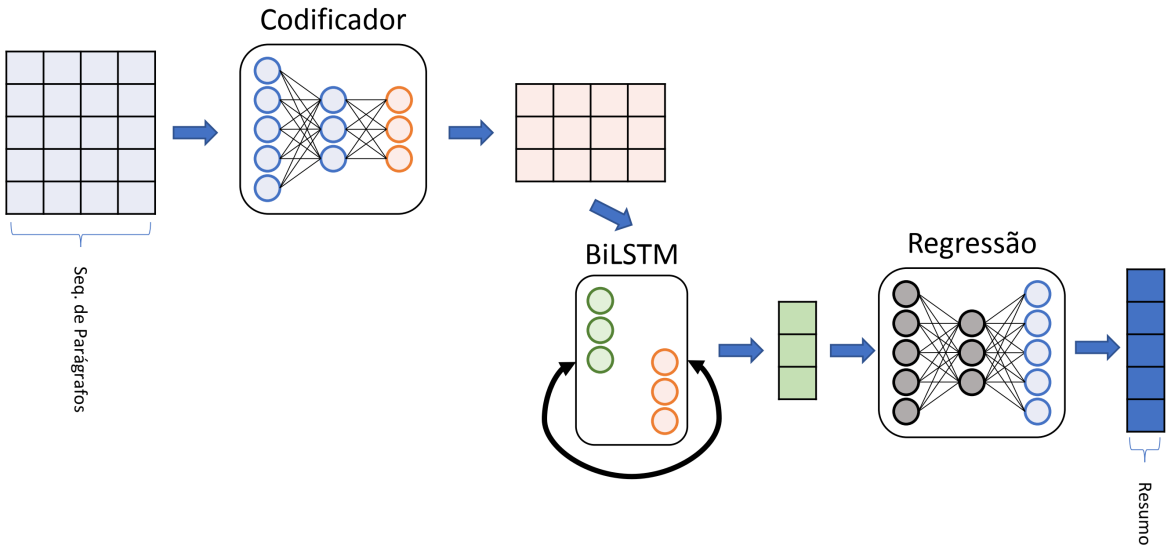
Tabela 2 – Arquitetura de referência para tarefa de classificação

Model: "Encoder"		
Layer (type)	Output Shape	Param #
Dense	(None, None, 64)	49216
BatchNormalization	(None, None, 64)	256
Dropout	(None, None, 64)	0
Dense	(None, None, 64)	4160
BatchNormalization	(None, None, 64)	256
Total params: 53,888		
Trainable params: 53,632		
Non-trainable params: 256		
Model: "LSTM"		
Layer (type)	Output Shape	Param #
Bidirectional	(None, None, 64)	24832
Bidirectional	(None, 64)	24832
BatchNormalization	(None, 64)	256
Total params: 49,920		
Trainable params: 49,792		
Non-trainable params: 128		
Model: "Classifier"		
Layer (type)	Output Shape	Param #
Dense	(None, 64)	8256
BatchNormalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 64)	4160
BatchNormalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 1)	65
Total params: 12,993		
Trainable params: 12,737		
Non-trainable params: 256		

Fonte: Próprio autor.

Na segunda arquitetura, foi feita uma adaptação da arquitetura anterior para a tarefa de regressão, o que é ilustrado na Figura 9. O funcionamento é similar ao já descrito para a tarefa de classificação, porém o vetor referente ao resumo não é passado na entrada junto à sequência de vetores de parágrafo. Além disso, a última camada densa apresentada na Tabela 2 passa a ter a quantidade de unidades igual à dimensão do *embedding* e a não ter função de ativação, transformando o classificador em um submodelo de regressão. Para as estratégias de vetorização que fizeram uso da arquitetura BERT, isso correspondeu a uma camada densa final com 49.920 parâmetros.

Figura 9 – Esquema da arquitetura de regressão para entrada com vetores de parágrafos



Fonte: Próprio autor.

Por último, na terceira arquitetura, as duas camadas BiLSTM, antes empilhadas em um único submodelo recorrente, são separadas em hierarquias, vide Figura 10. Na entrada da arquitetura, cada uma das sequências de vetores de sentença correspondeu a um parágrafo de texto (na figura, em azul claro), às quais foi adicionada uma sequência de vetores de sentença referentes ao resumo (na figura, em azul escuro).

De forma similar ao feito para os vetores de parágrafo nas arquiteturas anteriores, os vetores de sentença iniciais foram codificados para uma dimensão reduzida (na figura, em laranja). Em seguida, a primeira BiLSTM é aplicada a cada sequência de vetores de sentença, codificando cada parágrafo como um único vetor (na figura, representado em verde claro).

A segunda BiLSTM é aplicada à sequência de vetores de parágrafo, obtidos da primeira camada, representando o texto científico como um único vetor (na figura, em amarelo). Os vetores do texto e do resumo são então concatenados e passam por um submodelo classificador, como feito para a primeira arquitetura.

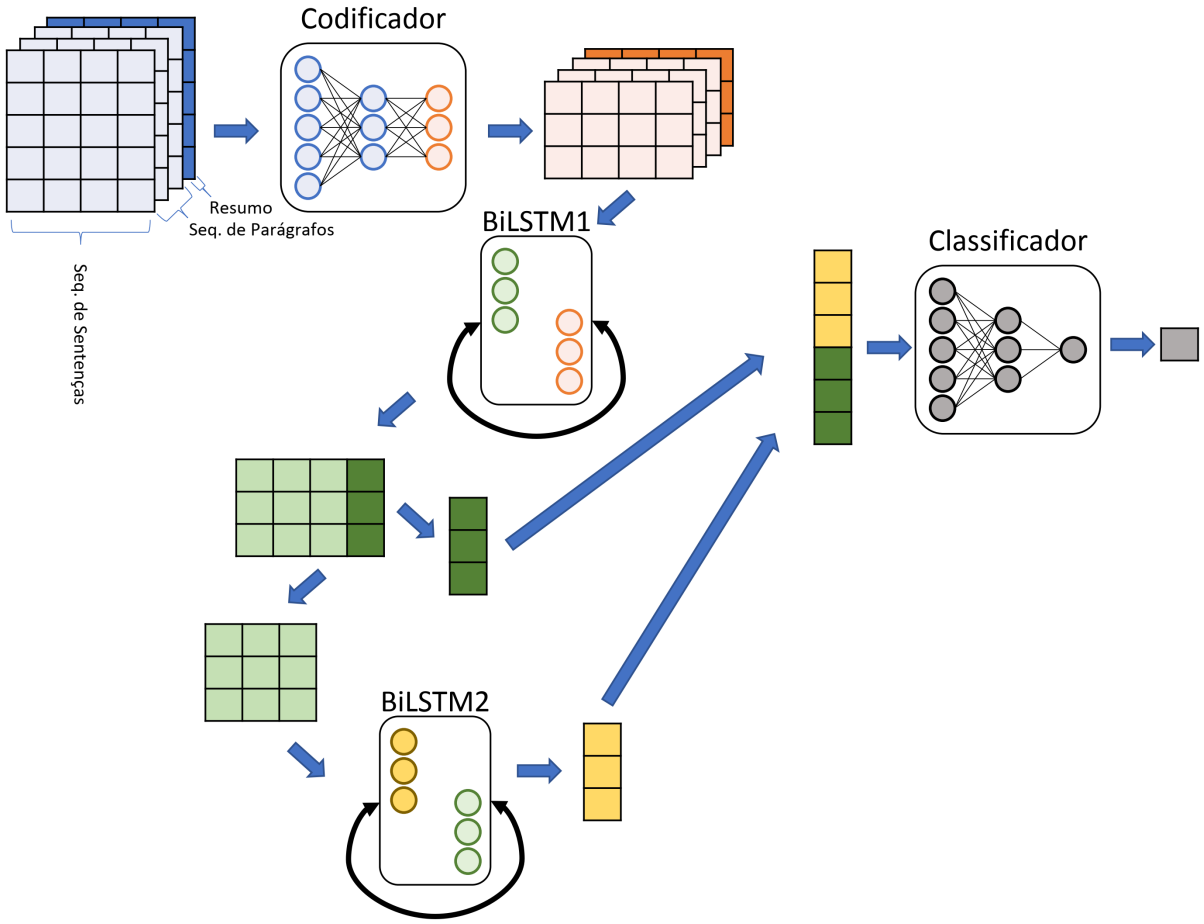
Detalhes das camadas da terceira arquitetura são apresentados na Tabela 3.

Tabela 3 – Arquitetura hierárquica para tarefa de classificação

Model: "Encoder+LSTM1"		
Layer (type)	Output Shape	Param #
Dense	(None, None, None, 64)	49216
BatchNormalization	(None, None, None, 64)	256
Dropout	(None, None, None, 64)	0
Dense	(None, None, None, 64)	4160
BatchNormalization	(None, None, None, 64)	256
Bidirectional	(None, None, 64)	24832
BatchNormalization	(None, None, 64)	256
Total params: 78,976		
Trainable params: 78,592		
Non-trainable params: 384		
Model: "LSTM2"		
Layer (type)	Output Shape	Param #
Bidirectional	(None, 64)	24832
BatchNormalization	(None, 64)	256
Total params: 25,088		
Trainable params: 24,960		
Non-trainable params: 128		
Model: "Classifier"		
Layer (type)	Output Shape	Param #
Dense	(None, 64)	8256
BatchNormalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 64)	4160
BatchNormalization	(None, 64)	256
Dropout	(None, 64)	0
Dense	(None, 1)	65
Total params: 12,993		
Trainable params: 12,737		
Non-trainable params: 256		

Fonte: Próprio autor.

Figura 10 – Esquema da arquitetura de classificação para entrada com vetores de sentenças



Fonte: Próprio autor.

4.6 Fração mais provável que o resumo original

Uma avaliação final das arquiteturas de rede neural, mais próxima da aplicação de um sistema de recomendação que faça uso das estratégias investigadas, foi obtida da seguinte forma, no conjunto de 157 artigos reservados para teste. O valor obtido dessa avaliação será em geral referenciado como "fração mais provável".

Para a arquitetura de regressão, as introduções dos artigos, consideradas como primeira seção do artigo após o resumo, foram processadas pela rede neural gerando um vetor de resumo estimado. A distância ℓ_1 desse vetor aos vetores de resumo foi obtida; e foi registrada qual a fração de resumos teve distância menor que a distância ao resumo original.

Para as arquiteturas de classificação, as introduções dos artigos foram processadas até a camada que antecede a entrada no Classificador da rede. Os resumos também foram processados pela rede até a camada que antecede o Classificador. Então se determinou o resultado da função de ativação sigmoide ao fim da rede para todos os pares introdução-resumo. Para cada introdução, foi registrada a fração de resumos mais prováveis que o resumo original.

5
Resultados

Neste capítulo, serão apresentados os resultados obtidos com os experimentos delineados no capítulo anterior.

Na primeira seção, são apresentados os resultados referentes à capacidade de agrupamento dos vetores de parágrafos das diversas estratégias. Essa análise busca identificar o potencial de recomendação de conteúdo feita pela simples proximidade entre vetores de parágrafos, dado que é o modo de recomendação de conteúdo que apresenta menor custo computacional para uma futura implementação em ambiente de produção.

Na segunda seção, modos de recomendação de conteúdo mais sofisticados são analisados, com o potencial de maior acurácia com contrapartida de maior custo computacional, como argumentado nos próximos dois parágrafos.

A arquitetura de referência é construída levando-se em conta que parágrafos estão previamente vetorizados e que a tarefa de classificação tenderia a apresentar melhores resultados que a tarefa de regressão. Isso seria o esperado pois determinar a proximidade de dois vetores diretamente a partir de uma métrica matemática é menos custoso que passá-lo por uma rede neural para determinar sua proximidade.

Adicionalmente, considerando-se que a estratégia de tomar alguma média dos vetores de sentença para se obter os vetores de parágrafos poderia ser muito simples e levar à perda considerável de precisão, uma terceira arquitetura é investigada, obtendo-se os vetores de parágrafos como o resultado da passagem da sequência de vetores de sentença por uma BiLSTM.

Ao todo, seriam então quatro modos de recomendação possíveis, listados a seguir, ordenados do mais simples ao mais complexo:

- verificar quais vetores de parágrafos da base de artigos são mais próximos à média dos vetores de parágrafos de entrada **com uma métrica matemática pré-definida** (viabilidade investigada na primeira subseção);
- verificar quais vetores de parágrafos para resumos da base de artigos são mais próximos à **saída de uma arquitetura de regressão** para uma sequência de vetores de parágrafo de entrada (variação que simplifica a arquitetura seguinte, tomada como referência);
- verificar quais vetores de parágrafos para resumos da base de artigos são classificados como próximos a partir da saída de uma arquitetura de **classificação** para uma sequência de vetores de parágrafo de entrada (é a arquitetura tomada como referência);
- verificar quais vetores de parágrafos para resumos da base de artigos são classificados como próximos a partir dà saída uma arquitetura de classificação para uma sequência de **vetores de sentença de entrada agrupados em parágrafos** (abordagem mais complexa).

5.1 Agrupamento dos vetores de parágrafos

Duas medidas foram usadas como estimadores da capacidade de agregação semântica de modelos de linguagem pré-treinados: o índice Rand adaptado e o escore Silhouette.

O resultado para o índice Rand adaptado é apresentado na Tabela 4. O melhor desempenho foi obtido com a estratégia que parte do modelo BERT e usa o vetor do seu *token* [CLS] como vetorização para a sentença, alcançando uma mediana de 52,3% dos parágrafos em artigo tiveram o parágrafo mais próximo na mesma seção. O segundo e terceiro melhores desempenhos também partiram do modelo pré-treinado BERT, fazendo uso da média sem *stopwords* e média ponderada, respectivamente, como forma de se obter o vetor de sentença. De forma geral, os piores desempenhos foram observados nas estratégias de vetorização que usaram a média ponderada para a obtenção dos vetores de sentenças, quando comparadas às que usaram média sem *stopwords*.

O resultado para o escore Silhouette é apresentado na Tabela 5. O escore foi inferior a zero em todas as estratégias de vetorização investigadas. Considerando-se que: o percentil obtido por *bootstrap* chegou a 90,5% no melhor dos casos e que o índice Rand indicou uma proximidade maior entre vetores dentro das seções que fora das seções; é possível especular que a distribuição dos vetores no hiperespaço não apresenta uma estrutura de agrupamentos distinta, com os agrupamentos de parágrafos em seções sobrepondo-se entre si nas suas margens.

Tabela 4 – Índice Rand adaptado para diferentes estratégias de vetorização de parágrafos

Modelo	**Vetorização de sentenças**	**p25**	**p50**	**p75**
BERT	Primeiro token	0,416	0,523	0,622
BERT	Média sem stopwords	0,405	0,508	0,605
BERT	Média ponderada	0,373	0,476	0,567
GloVe	Média sem stopwords	0,368	0,465	0,556
ELMo	Média sem stopwords	0,362	0,463	0,554
Word2Vec	Média sem stopwords	0,367	0,462	0,548
Sent2vec	(direto do próprio modelo)	0,349	0,443	0,531
fastText	Média sem stopwords	0,347	0,443	0,535
ELMo	Média ponderada	0,317	0,413	0,514
GloVe	Média ponderada	0,303	0,391	0,478
fastText	Média ponderada	0,295	0,375	0,471
Word2Vec	Média ponderada	0,282	0,368	0,462

Fonte: Próprio autor.

Nota: Calculado com amostra de 1605 artigos; p25, p50, p75 – percentis da amostra para o índice Rand adaptado. Vetor de parágrafo obtido como média simples de vetores de sentenças.

5.2 Desempenho das redes neurais

Para a arquitetura de rede neural tomada como referência para a tarefa de classificação (vide Tabela 2 do capítulo anterior), o resultado dos valores de função de perda e acurácia para o conjunto de validação são apresentados nas Figura 11 e Figura 12, respectivamente.

De maneira geral para todos os modelos, as estratégias de vetorização que aplicaram média ponderada para a obtenção de vetores de sentenças apresentaram pior desempenho quando comparadas às estratégias que envolveram a média sem *stopwords*; esse padrão também esteve presente para a acurácia de validação, com valores maiores para as estratégias com média ponderada. É possível argumentar que o hiperparâmetro a usado na geração de pesos para os vetores de palavras pudesse ser ajustado para a obtenção de melhores resultados, mas a otimização desse hiperparâmetro não foi realizada.

Conforme já esperado, as vetorizações de parágrafos derivadas do modelo BERT atingiram resultados superiores quando comparadas às demais estratégias. Em particular, o vetor de senteças obtido com a média sem *stopwords*, apresentou maior acurácia ao fim do treino quando comparada à média ponderada e à extração do primeiro *token* [CLS], usado como vetor para a classificação de sentenças no pré-treino de modelos BERT.

Tabela 5 – Escore Silhouette para diferentes estratégias de vetorização de parágrafos

Modelo	**Vetorização de sentenças**	**p25**	**p50**	**p75**	**bootstrap**
BERT	Primeiro token	-0,124	-0,087	-0,053	90,5
BERT	Média sem stopwords	-0,137	-0,104	-0,071	78,0
ELMo	Média sem stopwords	-0,145	-0,111	-0,081	68,5
BERT	Média ponderada	-0,152	-0,119	-0,084	59,0
ELMo	Média ponderada	-0,165	-0,131	-0,096	42,0
Word2Vec	Média sem stopwords	-0,171	-0,134	-0,101	73,0
GloVe	Média sem stopwords	-0,170	-0,134	-0,101	78,0
GloVe	Média ponderada	-0,190	-0,152	-0,110	46,0
Word2Vec	Média ponderada	-0,194	-0,152	-0,111	42,5
Sent2vec	(resultado do próprio modelo)	-0,209	-0,164	-0,121	35,0
fastText	Média sem stopwords	-0,209	-0,176	-0,142	61,0
fastText	Média ponderada	-0,243	-0,210	-0,174	33,5

Fonte: Próprio autor.

Nota: Calculado com amostra de 1549 artigos; p25, p50, p75 – percentis da amostra para o escore Silhouette com métrica ℓ_1. Vetor de parágrafo obtido como média simples de vetores de sentenças. Na última coluna, é apresentada a mediana do percentil para o *bootstrap*, com reamostragem de 100 conjuntos de agrupamento por artigo, em uma subamostra de 94 artigos.

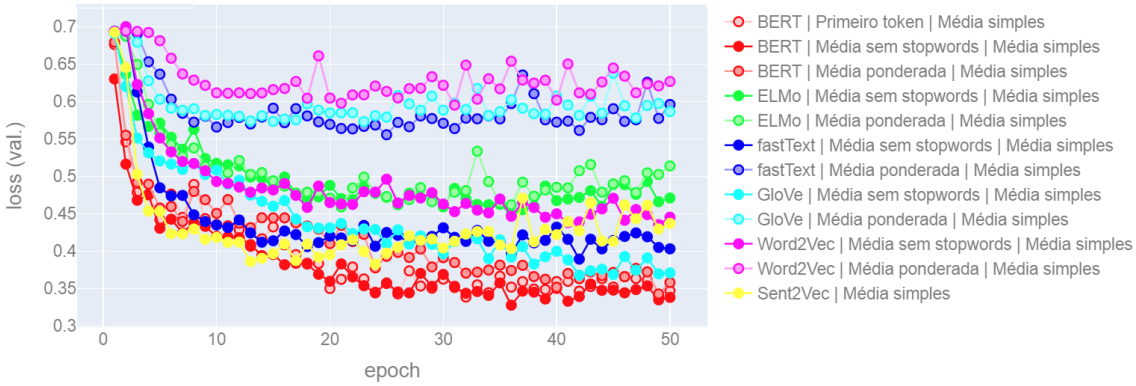
Uma avaliação mais próxima à aplicação real dessas redes neurais é apresentada na Tabela 6. São apresentados os valores da fração de resumos mais prováveis dadas as introduções do conjunto de testes, conforme explicado na seção 4.6. Para essa tabela, são apresentados os desempenhos para as arquiteturas de classificação de referência, que operaram com vetores de parágrafos obtidos por alguma média de vetores de sentenças. De forma similar ao observado para as medidas de desempenho de agrupamento, os melhores desempenhos foram observados com modelos pré-treinados BERT. Na estratégia que usa vetorização de sentenças pela média sem *stopwords*, metade das introduções alcançaram uma fração mais provável inferior a 3,18%.

O resultado do desempenho das variações de arquitetura nessa mesma medida são apresentados na Tabela 7. A abordagem com o modelo de classificação hierárquico levou a mediana da fração mais provável de 3,18% para 1,91%, e setenta e cinco porcento da introduções avaliadas tiveram uma fração mais provável melhor que 5,10%.

Para a abordagem com a arquitetura que reduz o custo computacional da recomendação, tornando-se uma regressão, a mediana elevou-se para 33,76%. Em um cenário onde a base de artigos seja maior que centenas de milhares, essa porcentagem significaria um sistema incapaz de recomendar conteúdos relacionados de maneira eficiente, sendo necessário um filtro prévio

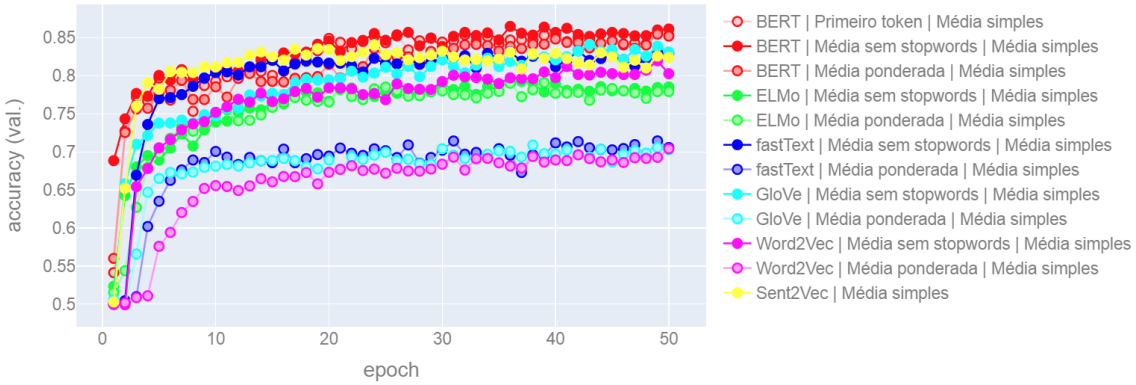
por palavras-chave para reduzir inicialmente a quantidade de artigos candidatos, por exemplo.

Figura 11 – Entropia cruzada binária de validação no treino com vetores de parágrafos



Fonte: Próprio autor.

Figura 12 – Acurácia de validação no treino com vetores de parágrafos



Fonte: Próprio autor.

5.3 Exemplos qualitativos de aplicação

A partir da Tabela 8, são apresentados alguns exemplos qualitativos de entrada e saída com textos. O procedimento de recomendação é descrito no parágrafo seguinte. Para a entrada de texto da Tabela 8, o procedimento retorna saída da Tabela 9; e para a entrada de texto da Tabela 10, a saída da Tabela 11.

Para esses exemplos, foi realizado o seguinte procedimento em duas etapas para a recomendação de conteúdo. Na primeira etapa foi usada a estratégia com mais acurácia dentre as anteriores para seleção inicial dos artigos relacionados ao texto de entrada, nomeadamente: *embedding* de fragmentos de palavras com modelo pré-treinado SciBERT; vetores de setença obtidos a partir da média sem *stopwords*; e arquitetura BiLSTM hierárquica. Como texto de entrada foram usados os dois primeiros parágrafos de uma introdução do conjunto de teste. Na segunda etapa, é apontado o parágrafo de cada artigo selecionado na primeira etapa mais próximo

Tabela 6 – Fração mais provável que o resumo original dada introdução, no conjunto de teste após treino com vetores de parágrafos

Modelo	Vetorização de sentenças	p25	p50	p75
BERT	Média sem stopwords	0,0127	0,0318	0,0637
BERT	Média ponderada	0,0127	0,0382	0,0637
GloVe	Média sem stopwords	0,0064	0,0318	0,0764
Sent2vec	(direto do próprio modelo)	0,0191	0,0382	0,0924
BERT	Primeiro token	0,0191	0,0382	0,0955
fastText	Média sem stopwords	0,0064	0,0382	0,0955
Word2vec	Média ponderada	0,0159	0,0573	0,0892
ELMo	Média ponderada	0,0127	0,0510	0,1051
ELMo	Média sem stopwords	0,0191	0,0573	0,1306
fastText	Média ponderada	0,0573	0,1465	0,2739
Word2Vec	Média ponderada	0,0446	0,1720	0,2994
GloVe	Média ponderada	0,0764	0,1656	0,3280

Fonte: Próprio autor.

Nota: Avaliado no conjunto de teste com 157 artigos; p25, p50, p75 – percentis da fração mais provável que o resumo original dada introdução.

Tabela 7 – Fração mais provável que o resumo original dada introdução, no conjunto de teste após treino, com variações no tipo de tarefa e vetorização de parágrafos

Vetorização de parágrafos	Tarefa	p25	p50	p75
BiLSTM	Classificação	0,0064	0,0191	0,0510
Média simples	Classificação	0,0127	0,0318	0,0637
Média simples	Regressão	0,1306	0,3376	0,6210

Fonte: Próprio autor.

Nota: Avaliado no conjunto de teste com 157 artigos; p25, p50, p75 – percentis da fração mais provável que o resumo original dada introdução.

à média dos parágrafos de entrada (abordagem mais simples e menos custosa). Nos exemplos, é possível notar que o tema dos artigos recomendados encontra-se dentro do texto original. Como apenas 157 artigos pertenceram ao conjunto de testes nesse trabalho, é possível que os resultados fossem ainda mais adequados se executados em uma base maior.

Tabela 8 – Entrada para primeiro exemplo qualitativo

Entrada (BEZGIN et al., 2009)
[['Brain atlases are indispensable tools for localizing specific '
'structures and making interindividual comparisons with reference to a '
'variety of markers',
'Such atlases are typically constructed from stacks of either volume '
'element imaging data or serial histological sections',
'For atlases of the macaque brain in particular non-invasive imaging '
'data have mainly provided coordinates and shape information whereas '
'histological sections have been used for micro-structural '
'delineations and partitioning',
'Whereas previous macaque atlases were mostly limited to brain stem '
'structures occasionally indicating the gyral and sulcal pattern of '
'cerebral cortex the first atlas with a complete cortical '
'parcellation was published by Paxinos et al in its first edition in '
'2000',
'To perform the task of brain parcellation coronal sections of one '
'brain were shown to experts in cytoarchitecture for partitioning and '
'a list of delineated structures and terms was compiled',
'Currently there are at least three macaque atlases with parcellations '
'of cerebral cortex and matched information from several data '
'modalities'],
['The publication of atlases such as those discussed above has much '
'potential for their interfacing with other primate data modalities '
'that exist in electronic form',
'In particular we see mutual synergistic benefits by interfacing an '
'electronic version of the Paxinos et al atlas with our database '
'collation of connectivity data in the macaque brain',
'The prime advantages are',
'Intuitive visual query interface as an alternative to not so familiar '
'or even confusing nomenclature used by specialists in anatomy '
'Mapping of coordinate-independent parcellation-based data to '
'spatially registered data Presentation of connectivity data as '
'straight-line arrows linking parcellated brain regions Overlay of '
'several data modalities for integrated data presentation']]
Resumo do trabalho original (BEZGIN et al., 2009)
['Brain atlases are widely used in experimental neuroscience as tools '
'for locating and targeting specific brain structures',
'Delineated structures in a given atlas however are often difficult to '
'interpret and to interface with database systems that supply '
'additional information using hierarchically organized vocabularies',
'Here we discuss the concept of volume-to-ontology mapping in the '
'context of macroscopical brain structures',
'We present Java tools with which we have implemented this concept for '
'retrieval of mapping and connectivity data on the macaque brain from '
'the CoCoMac database in connection with an electronic version of The '
'Rhesus Monkey Brain in Stereotaxic Coordinates authored by George '
'Paxinos and colleagues',
'The software including our manually drawn monkey brain template can '
'be downloaded freely under the GNU General Public License',
'It adds value to the printed atlas and has a wider informatics '
'application since it can read appropriately annotated data from '
'delineated sections of other species and organs and turn them into 3D '
'registered stacks',
'The tools provide additional features including visualization and '
'analysis of connectivity data volume and centre-of-mass estimates and '
'graphical manipulation of entire structures which are potentially '
'useful for a range of research and teaching applications']

Fonte: Indicadas na tabela.

Tabela 9 – Saída para primeiro exemplo qualitativo

Resumo do trabalho recomendado (GRIFFANTI et al., 2016)
['White matter hyperintensities are a hallmark of small vessel diseases', 'Yet no automated segmentation method is readily and widely used ' 'especially in patients with extensive WMH where lesions are close to the ' 'cerebral cortex', 'BIANCA is a new fully automated supervised method for WMH segmentation', 'In this study we optimized and compared BIANCA against a reference method ' 'with manual editing in a cohort of patients with extensive WMH', 'This was achieved in two datasets', 'a clinical protocol with 90 patients having 2-dimensional FLAIR and an ' 'advanced protocol with 66 patients having 3-dimensional FLAIR', 'We first determined simultaneously which input modalities and which ' 'training sets were better compared to the reference', 'Three strategies for the selection of the threshold that is applied to ' 'the probabilistic output of BIANCA were then evaluated', 'chosen at the group level based on Fazekas score or determined ' 'individually', 'Accuracy of the segmentation was assessed through measures of spatial ' 'agreement and volumetric correspondence with respect to reference ' 'segmentation', 'Based on all our tests we identified multimodal inputs mixed WMH load ' 'training set and individual threshold selection as the best conditions to ' 'automatically segment WMH in our cohort', 'A median Dice similarity index of 0 80 and an intraclass correlation ' 'coefficient of 0 97 were obtained for the clinical protocol', 'However Bland-Altman plots identified a difference with the reference ' 'method that was linearly related to the total burden of WMH', 'Our results suggest that BIANCA is a reliable and fast segmentation ' 'method to extract masks of WMH in patients with extensive lesions']
Parágrafo do trabalho recomendado (GRIFFANTI et al., 2016)
['BIANCA is a new fully automated supervised method for WMH detection ' 'based on the k-nearest neighbor algorithm', 'The output of BIANCA is a probabilistic map of WMH', 'It was optimized and validated so far on two subject categories that are ' 'representative of groups of patients where the clinical importance of WMH ' 'is being increasingly recognized', 'a predominantly neurodegenerative cohort including people with or at risk ' 'of Alzheimer's disease and a predominantly vascular cohort including ' 'patients with or at risk of vascular cognitive impairment', 'BIANCA is multimodal highly flexible so that users can adapt the tool to ' 'their protocol and specific needs and freely available', 'It also showed higher performance compared to other freely available ' 'algorithms', 'CASCADE and the toolbox LST', 'Lesion Segmentation Tool using the same dataset', 'Although BIANCA was validated on two datasets these findings were limited ' 'to patients with low to moderate burden of WMH', 'As yet no available algorithm has been validated in patients presenting ' 'with a large extent of WMH and showing a high variability in WMH spatial ' 'pattern']

Fonte: Indicadas na tabela.

Tabela 10 – Entrada para segundo exemplo qualitativo

Entrada (JING et al., 2018)
[['Modular self-reconfigurable robots are systems composed of repeated ' 'robot elements that have the ability to connect together to form ' 'larger robotic structures', 'These systems distinguish themselves from traditional robot systems ' 'through their ability to self-reconfigure', 'changing the connective structure of the modules to assume different ' 'shapes that have different capabilities', 'Over the last 3 decades many kinds of modular reconfigurable robots ' 'have been built and many different approaches have been introduced ' 'for controlling and programming them'], ['Robotics research is increasingly focused on deploying robots in ' 'real-world applications such as search and rescue', 'Operating in these scenarios entails handling an enormous amount of ' 'variability in task requirements and environment conditions', 'One approach to this problem is to build complex monolithic systems ' 'such as large humanoids', 'These systems can perform a wide range of actions but are extremely ' 'complex', 'In a sense their broad range of capability comes at the cost of ' 'having to solve each individual problem in a complicated way', 'For example to pick up and move an object a humanoid must balance on ' 'two legs while using a high degree of freedom arm to manipulate the ' 'object', 'In contrast a robot that was purpose-built for that task could ' 'accomplish it with far fewer DOF and would require less complicated ' 'control algorithms']]
Resumo do trabalho original (JING et al., 2018)
['The advantage of modular self-reconfigurable robot systems is their ' 'flexibility but this advantage can only be realized if appropriate ' 'configurations and behaviors can be selected for a given task', 'In this paper we present an integrated system for addressing ' 'high-level tasks with modular robots and demonstrate that it is ' 'capable of accomplishing challenging multi-part tasks in hardware ' 'experiments', 'The system consists of four tightly integrated components', 'a high-level mission planner a large design library spanning a wide ' 'set of functionality a design and simulation tool for populating the ' 'library with new configurations and behaviors and modular robot ' 'hardware', 'This paper builds on earlier work by Jing et al extending the ' 'original system to include environmentally adaptive parametric ' 'behaviors which integrate motion planners and feedback controllers ' 'with the system']

Fonte: Indicadas na tabela.

Tabela 11 – Saída para segundo exemplo qualitativo

Resumo do trabalho recomendado (TAN; CHITRE; HOVER, 2016)
['We present a cooperative bathymetry-based localization approach for a '
'team of low-cost autonomous underwater vehicles each equipped only with '
'a single-beam altimeter a depth sensor and an acoustic modem',
'The localization of the individual AUV is achieved via fully '
'decentralized particle filtering with the local filter's measurement '
'model driven by the AUV's altimeter measurements and ranging information '
'obtained through inter-vehicle communication',
'We perform empirical analysis on the factors that affect the filter '
'performance',
'Simulation studies using randomly generated trajectories as well as '
'trajectories executed by the AUVs during field experiments successfully '
'demonstrate the feasibility of the technique',
'The proposed cooperative localization technique has the potential to '
'prolong AUV mission time and thus open the door for long-term autonomy '
'underwater']
Parágrafo do trabalho recomendado (TAN; CHITRE; HOVER, 2016)
['Often a particle filter is designed to estimate and track a large number '
'of system variables which requires a large number of particles for the '
'filter to converge',
'This poses a challenge for the AUVs' limited computational power onboard',
'In order to alleviate this a number of researchers have adopted an '
'approach called the Marginalized Particle Filter\xa0 also referred to as '
'Rao-Blackwellization',
'The idea behind the MPF is to marginalize the system states that exhibit '
'linear dynamics and to estimate the marginalized states using a Kalman '
'Filter',
'The remaining states with reduced dimension can then be estimated by the '
'PF thus lowering the number of particles required to produce comparable '
'results',
'The MPF has been employed in Nordlund and Gustafsson in an integrated '
'navigation system of an aircraft with a state vector of more than 15 '
'dimensions and simulation results showed good performance with a much '
'lower computational load',
'In the domain of underwater navigation the authors in Teixeira et\xa0al '
'have shown the feasibility of applying the MPF for an AUV with the number '
'of particles as low as 500 and manage to achieve good localization',
'The results have encouraged the application of MPF-based localization '
'techniques in low-cost limited computational-power AUVs',
'The work presented in this paper adopts the MPF localization technique '
'for its advantages']

Fonte: Indicadas na tabela.

6
Considerações Finais

6.1 Conclusão

Redes BiLSTMs hierárquicas se mostraram adequadas ao processamento de texto agrupado em parágrafos, ainda com potencial de melhoria na sua acurácia através da otimização de hiperparâmetros. No entanto, não é possível descartar que outras arquiteturas também tenham a capacidade de aumentar a acurácia e velocidade de processamento, como as baseadas em camadas *Transformers*. A partir das estratégias investigadas, a assistência bibliográfica durante a produção de textos científicos se mostrou viável, sendo possível a recomendação de conteúdo em diversos pontos do balanço entre qualidade e velocidade de recomendação.

6.2 Contribuições

O presente trabalho aborda um problema relevante à comunidade científica, desenvolvendo um método auxiliar à sobrecarga de informação através do uso de modelos de linguagem pré-treinados de alta qualidade. As aplicações de um serviço baseado nessa estratégia são diversas, e incluem a elaboração de projetos de pesquisa e escrita de artigos científicos. Além disso, é apresentada maneira de se treinar redes neurais para vetorizar elementos textuais maiores de forma não supervisionada a partir de corpus científico.

6.3 Limitações

Apesar de ter abrangido uma grande quantidade de modelos de linguagem recentes, a quantidade de artigos usados durante o treinamento da rede foi reduzida, tendo usado apenas cerca de 5% dos artigos coletados nos periódicos escolhidos, devido às limitações de disponibilidade do ambiente de testes. Com o uso de todo o corpus armazenado, a arquitetura da rede treinada

poderia ter sido dimensionada com mais parâmetros e os resultados obtidos poderiam ter sido melhores quanto à acurácia e à fração mais provável. Isso também é verdade para a otimização de hiperparâmetros, que não foi realizada para todos os casos.

A conversão de XHTML para JSON também é um ponto onde o *pipeline* pode sofrer melhorias. Como mostrado nos exemplos qualitativos, muitas sentenças sofreram quebra prematura durante esse pré-processamento. A melhor conservação das sentenças originais poderia ter aproveitado ainda mais o potencial dos *embeddings* de palavras contextualizados à sentença ELMo e BERT.

A aplicação das estratégias de recomendação em ambiente de produção podem demandar algumas etapas prévias para tratar questões de escalabilidade. Em alguns testes com o módulo *timeit* do Python, realizados no ambiente do Google Colab, a recomendação dentre os 157 artigos de teste para uma entrada de parágrafos levou $31ms$, $136ms$ e $167ms$ para as arquiteturas de regressão, de classificação simples e de classificação com BiLSTMs hierárquicas, respectivamente. Caso esse tempo aumente proporcionalmente à quantidade de artigos candidatos, uma base com centenas de milhares de artigos poderia levar mais de um minuto para gerar recomendações para um texto de entrada. Seria preciso, portanto, uma forma de realizar uma filtragem prévia dentre os artigos candidatos, possivelmente através da pré-seleção com algumas palavras-chave.

6.4 Trabalhos futuros

A implementação do serviço de recomendação propriamente dito pode ser um trabalho futuro visado a mais longo prazo. Para mais de imediato, um quantidade maior de artigos pode ser usada para o treinamento das redes neurais, com a otimização de hiperparâmetros, partindo das melhores estratégias observadas neste trabalho.

Outras áreas científicas além de Ciência da Computação podem ser exploradas. Por exemplo, a área biomédica tem o potencial de apresentar resultados superiores ao observado no presente trabalho, tendo em vista que tipicamente se usa uma quantidade menor de notação matemática em seus artigos.

Outras arquiteturas de redes neurais podem ser usadas recebendo os vetores pré-treinados, como as que fazem uso de mecanismos de atenção por terem processamento mais paralelizável. Além disso, é possível estender o conceito de BiLSTM hierárquica para mais um nível, um que envolve a vetorização de sentenças a partir dos vetores de palavras pré-treinados.

Referências Bibliográficas

ARORA, S.; LIANG, Y.; MA, T. A simple but tough-to-beat baseline for sentence embeddings. 2016. Citado 2 vezes nas páginas 24 e 28.

BAI, X. et al. Scientific paper recommendation: A survey. *IEEE Access*, IEEE, v. 7, p. 9324–9339, 2019. Citado na página 22.

BAWDEN, D.; ROBINSON, L. The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of information science*, Sage Publications Sage UK: London, England, v. 35, n. 2, p. 180–191, 2009. Citado na página 9.

BEZGIN, G. et al. Matching spatial with ontological brain regions using java tools for visualization, database access, and integrated data analysis. *Neuroinformatics*, Springer, v. 7, n. 1, p. 7–22, 2009. Citado na página 43.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, MIT Press, v. 5, p. 135–146, 2017. Citado na página 16.

DERNTL, M. Basics of research paper writing and publishing. *International Journal of Technology Enhanced Learning*, Inderscience Publishers, v. 6, n. 2, p. 105–123, 2014. Citado na página 9.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Citado 3 vezes nas páginas 18, 20 e 21.

EVANS, J. A. Electronic publication and the narrowing of science and scholarship. *science*, American Association for the Advancement of Science, v. 321, n. 5887, p. 395–399, 2008. Citado na página 9.

FRITH, U. Fast lane to slow science. *Trends in cognitive sciences*, Elsevier, v. 24, n. 1, p. 1–2, 2020. Citado na página 9.

GRIFFANTI, L. et al. Bianca (brain intensity abnormality classification algorithm): a new tool for automated segmentation of white matter hyperintensities. *Neuroimage*, Elsevier, v. 141, p. 191–205, 2016. Citado na página 44.

HASSAN, H. A. M. Personalized research paper recommendation using deep learning. In: *Proceedings of the 25th conference on user modeling, adaptation and personalization*. [S.l.: s.n.], 2017. p. 327–330. Citado na página 22.

HASSAN, H. A. M. et al. Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation? 2019. Citado 2 vezes nas páginas 10 e 22.

HILL, F.; CHO, K.; KORHONEN, A. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016. Citado na página 23.

HILL, F. et al. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, MIT Press, v. 4, p. 17–30, 2016. Citado na página 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 17.

HOULSBY, N. et al. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019. Citado na página 25.

JING, G. et al. Accomplishing high-level tasks with modular robots. *Autonomous Robots*, Springer, v. 42, n. 7, p. 1337–1354, 2018. Citado na página 45.

JINHA, A. E. Article 50 million: an estimate of the number of scholarly articles in existence. *Learned Publishing*, Wiley Online Library, v. 23, n. 3, p. 258–263, 2010. Citado na página 9.

JOHNSON, R.; WATKINSON, A.; MABE, M. The stm report: An overview of scientific and scholarly publishing. *International Association of Scientific, Technical and Medical Publishers*, 2018. Citado na página 9.

JOULIN, A. et al. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016. Citado na página 16.

KIROS, R. et al. Skip-thought vectors. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 3294–3302. Citado na página 23.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 25.

LAYTON, R.; WATTERS, P.; DAZELEY, R. Evaluating authorship distance methods using the positive silhouette coefficient. *Natural Language Engineering*, Cambridge University Press, v. 19, n. 4, p. 517–535, 2013. Citado na página 28.

LE, Q. V. et al. Ica with reconstruction cost for efficient overcomplete feature learning. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2011. p. 1017–1025. Citado na página 25.

MARTIN, J. H.; JURAFSKY, D. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. [S.l.]: Pearson/Prentice Hall Upper Saddle River, 2009. Citado 2 vezes nas páginas 11 e 16.

MARTÍN, P. et al. Publishing research in english-language journals: Attitudes, strategies and difficulties of multilingual scholars of medicine. *Journal of English for Academic Purposes*, Elsevier, v. 16, p. 57–67, 2014. Citado na página 9.

MCCANN, B. et al. Learned in translation: Contextualized word vectors. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 6294–6305. Citado na página 17.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. Citado 4 vezes nas páginas 11, 12, 16 e 24.

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 2 vezes nas páginas 11 e 13.

MNIH, A.; TEH, Y. W. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012. Citado na página 14.

MORIN, F.; BENGIO, Y. Hierarchical probabilistic neural network language model. In: CITESEER. *Aistats*. [S.l.], 2005. v. 5, p. 246–252. Citado na página 14.

PAGLIARDINI, M.; GUPTA, P.; JAGGI, M. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017. Citado na página 24.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. 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 15.

PETERS, M. E. et al. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. Citado 2 vezes nas páginas 17 e 18.

RADFORD, A. et al. *Improving language understanding with unsupervised learning*. [S.l.], 2018. Citado na página 18.

SHRESTHA, P.; JACQUIN, C.; DAILLE, B. Clustering short text and its evaluation. In: SPRINGER. *International Conference on Intelligent Text Processing and Computational Linguistics*. [S.l.], 2012. p. 169–180. Citado na página 28.

SMALDINO, P. E.; MCELREATH, R. The natural selection of bad science. *Royal Society open science*, The Royal Society, v. 3, n. 9, p. 160384, 2016. Citado na página 9.

TAN, Y. T.; CHITRE, M.; HOVER, F. S. Cooperative bathymetry-based localization using low-cost autonomous underwater vehicles. *Autonomous Robots*, Springer, v. 40, n. 7, p. 1187–1205, 2016. Citado na página 46.

VASWANI, A. et al. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008. Citado 3 vezes nas páginas 18, 19 e 20.

WANG, S.; MANNING, C. D. Baselines and bigrams: Simple, good sentiment and topic classification. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. [S.l.], 2012. p. 90–94. Citado na página 16.

WENG, L. *Generalized language models: CoVe, ELMo & cross-view training*. 2019. <https://www.topbots.com/generalized-language-models-cove-elmo/>. Acessado em: 2019-08-15. Citado na página 17.

WIETING, J. et al. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015. Citado na página 24.

YOSINSKI, J. et al. How transferable are features in deep neural networks? In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 3320–3328. Citado 2 vezes nas páginas 11 e 25.

ZHANG, L.; WILSON, S. R.; MIHALCEA, R. Direct network transfer: Transfer learning of sentence embeddings for semantic similarity. *arXiv preprint arXiv:1804.07835*, 2018. Citado na página 25.