UNIDADE DE APRENDIZAGEM: Representação vetorial de textos – word embeddings – word2vec

Apresentação

O processamento de linguagem natural busca a interação entre pessoas e máquinas na linguagem utilizada pelos seres humanos. Diferentemente de outros problemas, as diversas linguagens existentes não podem ser facilmente aproximadas por meio de relações lógicas diretas, o que por si só dificulta essa tentativa de comunicação.

Na maior parte, os algoritmos de processamento de linguagem natural lidam com representações esparsas e discretas de termos, palavras, caracteres ou outros elementos presentes nos textos, documentos e até mesmo áudios.

As técnicas de representação vetorial de textos, como o *word embeddings*, incorporam mais informações a respeito dos elementos presentes em um texto do que a simples frequência destes. Por meio delas, podem ser relacionadas características semânticas e sintáticas das palavras, que estabelecem conexões mais complexas entre estas e se aproximam muito da forma como os humanos compreendem e utilizam a linguagem. Além disso, graças à representação vetorial e contínua de palavras e caracteres na forma de números, podem-se empregar funções matemáticas para analisar e relacionar esses elementos.

Nesta Unidade de Aprendizagem, você vai conhecer as técnicas mais comuns utilizadas até o desenvolvimento de *word embeddings*, como TF-IDF e *bag-of-words*. Estas serão, então, comparadas com técnicas de representação vetorial, que incluem *word2vec*, *GloVe* e *fastText*. Por fim, você verá de que forma um algoritmo de *word2vec* pode ser implementado.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Definir métodos de vetorização de palavras do tipo *word embeddings*.
- Descrever algoritmos de *word embeddings*.
- Analisar a implementação de algoritmos de *word embeddings*.

Desafio

Os vetores contínuos apresentam outras vantagens para além de incorporar características mais complexas do texto, como as relações semânticas. Eles também podem ser utilizados para comparar a similaridade entre duas palavras.

Ciente disso, na qualidade de profissional, considere o seguinte cenário:

Você está desenvolvendo um aplicativo de tradução, porém, somente um dos dois idiomas conta com vasto vocabulário. Para ampliar a capacidade de tradução, sempre que uma palavra tiver tradução desconhecida, seu aplicativo poderá recorrer a palavras sinônimas a ela, que tenham tradução cadastrada.

Para tanto, você precisará identificar quais palavras são sinônimas, dispondo apenas dos vetores e da matriz de frequência dessas palavras.

As tabelas a seguir apresentam quatro palavras, sendo dois pares de sinônimos:

WORD EMBEDDINGS

	DOMINÂNCIA	VALÊNCIA	INTENSIDADE
PALAVRA 1	2	4	3
PALAVRA 2	3	5	1
PALAVRA 3	7	2	2
PALAVRA 4	4	1	2

MATRIZ TERMO-FREQUÊNCIA

	DOC 1	DOC 2	DOC 3
PALAVRA 1	5	1	20
PALAVRA 2	25	30	15
PALAVRA 3	18	2	0
PALAVRA 4	6	21	2

Diante do exposto, responda:

Como os dados apresentados podem ajudar a determinar a similaridade entre os vetores e definir quais palavras são sinônimas?

Infográfico

Quase 60 anos se passaram desde uma famosa frase de John Rupert Firth, que definiu a importância do contexto no processamento da linguagem natural, até o modelo *word2vec*, que possibilitou que aplicações práticas desse conceito fossem desenvolvidas.

No Infográfico, veja a sequência de eventos que determinaram o surgimento do *word embeddings* até a criação do *word2vec*.

WORD EMBEDDINGS:

COMO SURGIU O MÉTODO QUE REVOLUCIONOU O PROCESSAMENTO DE LINGUAGEM NATURAL

Muito tempo se passou desde a definição da importância do contexto no processamento da linguagem natural até o surgimento do modelo *word2vec*.

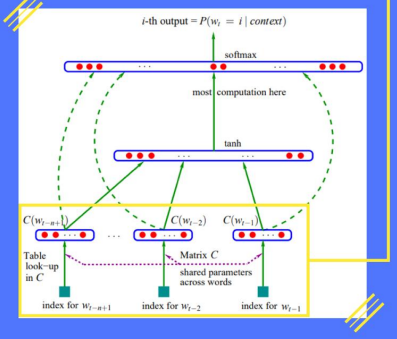
1957 - HIPÓTESE DISTRIBUTIVA

"Você deve conhecer uma palavra segundo o que a acompanha." Foi com essa frase que John Rupert Firth popularizou a hipótese distributiva que serviria de base, mais tarde, para o desenvolvimento dos vetores semânticos. No entanto, as cinco décadas seguintes foram marcadas pelo uso quase exclusivo de vetores discretos em técnicas como ***bag-of-words*** (saco de palavras), muito pela dificuldade de representar as várias características de uma palavra.

2003 - APRENDIZADO SEMISSUPERVISIONADO

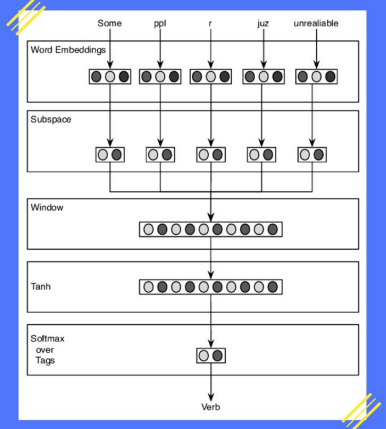
Neste ano, foi criado o primeiro modelo neural de linguagem de aprendizado semissupervisionado, por Yoshua Bengio *et al.* Trata-se de uma representação mais eficiente, porém com alto custo computacional devido ao tipo de função (*softmax*) utilizado para percorrer todo o vocabulário:

$$P(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$



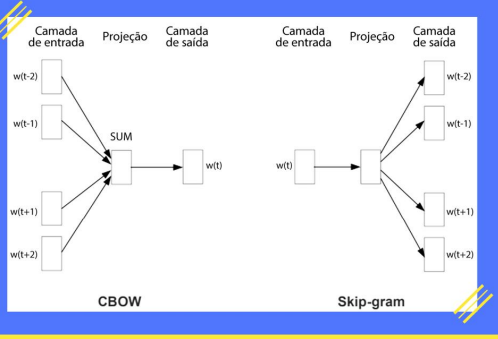
2008 - CONCEITO DE JANELA

Ronan Collobert utilizou o conceito de janela (*window*) para reduzir a varredura, antes do vocabulário inteiro, para apenas as palavras mais próximas, ou seja, que fazem parte do contexto mais próximo da palavra. Apesar de tudo, o custo computacional ainda era um empecilho.



2013 - WORD2VEC

Desenvolvido por Tomas Mikolov, surge o *word2vec*, simplificando o processo de treinamento, com *performance* e agilidade superiores, possibilitando o uso prático de *word embeddings* ao substituir as camadas ocultas por uma camada de projeção das entradas que diminui os cálculos necessários.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

O *word embeddings* é um caso interessante de aprendizado por representação, já que, lidando com valores contínuos, é capaz de incorporar, em um espaço vetorial, características que denotam semântica, sintaxe e outros elementos do texto e das palavras.

No capítulo Representação vetorial de textos – *word embeddings* – *word2vec*, da obra *Processamento de Linguagem Natural*, você verá uma breve revisão dos métodos clássicos de representação e conhecerá os benefícios do *word embeddings*, bem como os três algoritmos mais comuns: *word2vec*, *GloVe* e *fastText*.

Boa leitura.

PROCESSAMENTO DE LINGUAGEM NATURAL

Maikon Lucian Lenz

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Representação vetorial de textos — *word embeddings* — word2vec

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir métodos de vetorização de palavras do tipo *word embeddings*.
- Descrever algoritmos de *word embeddings*.
- Analisar a implementação de algoritmos de *word embeddings*.

Introdução

A incorporação de palavras a um espaço vetorial abriu um leque de oportunidades para o processamento de linguagem natural. Embora a maneira como o homem se comunica não obedeça a regras fáceis de determinar, com o surgimento da *word embeddings*, que mapeia desde palavras até vetores numéricos e algoritmos que utilizam esse conceito, como word2vec e GloVe, puderam ser observados modelos bastante eficazes em operações de analogia.

Neste capítulo, você compreenderá os conceitos de *word embeddings* e como podem representar melhor as palavras em comparação aos métodos clássicos. Ainda, verá os algoritmos de word2vec, GloVe e fastText, e o modelo pré-treinado mais recente denominado BERT. E, ao final, observará detalhadamente a implementação do word2vec, que se divide nos tipos CBOW (*continuous bag-of-words*) e Skip-Gram.

1 *Word embeddings*

O contexto em que as palavras estão inseridas representa um dos elementos que devemos observar para determinar o significado de um texto e até mesmo eliminar ambiguidades na definição de palavras, já que o seu significado também pode ser compreendido a partir do seu contexto de aplicação. Isso é o que sugere a hipótese distributiva: sinônimos aparecem com frequência no mesmo ambiente e a diferença entre duas palavras pode ser medida a partir da diferença entre os ambientes em que estão inseridas (HELLRICH, 2019).

Ao utilizarmos as características de distribuição, são criados vetores semânticos, nos quais se mapeiam os elementos de um texto para vetores, o que permite o emprego de ferramentas matemáticas na manipulação desses dados (JURAFSKY; MARTIN, 2019).

Com a *word embeddings* (incorporação de palavras), introduz-se o conceito de aprendizado por representação, já que nos métodos clássicos de processamento da linguagem natural (PLN) as palavras são representadas simplesmente por uma sequência de caracteres (p. ex., modelos do tipo *bag-of-words*) ou como um índice de um vocabulário (p. ex., modelo n-gram) (JURAFSKY; MARTIN, 2019).

No BoW (do inglês, *bag-of-words*, ou "saco de palavras"), os vetores se restringem a representar a frequência das palavras em um documento, ou seja, não há qualquer informação a respeito da semântica, da similaridade sintática, da posição ou do contexto das palavras, mas apenas a lista não ordenada delas associada à quantidade com que aparecem no documento.

A representação do significado de uma palavra (semântica lexical) por um modelo deve conseguir extrair as informações necessárias para executar as tarefas demandadas pela aplicação. Frequentemente, as palavras apresentam um sentimento associado, diferentes conotações e relações de semelhança e diferença ou de sentido, podendo ser sinônimas ou antônimas de outras palavras (HELLRICH, 2019).

Duas palavras são sinônimas se podem se substituir mutuamente na sentença sem alterar o sentido, tendo, portanto, o mesmo significado proposicional, ainda que o significado de nenhuma palavra seja idêntico ao de outra. Por simetria, um antônimo deverá inverter o sentido dado à sentença (JURAFSKY; MARTIN, 2019).

Já relações de semelhança são mais flexíveis e podem se apresentar em diferentes graduações, por exemplo, a palavra "uva" é similar à palavra "maçã", já que ambas se referem a frutas.

A maioria das palavras não nutre uma relação de sinônimo ou antônimo entre si, logo podem ser mais bem representadas por algum nível de semelhança, que permitirá a execução de tarefas de PLN de modo mais flexível e abrangente. São possíveis, ainda, outras relações, como entre palavras que compartilham alguma associação mesmo sem ser similares, como as palavras "copo" e "suco" que, além de estarem ambas relacionadas ao ato de beber algum líquido, podem estar ligadas pelo mesmo campo semântico (p. ex., a cozinha de casa ou uma lanchonete) e por um mesmo quadro semântico (em que o copo exerce um papel e o suco outro) (JURAFSKY; MARTIN, 2019).

No entanto, foi a partir de um estudo da conotação das palavras que se criou uma das primeiras representações das palavras por vetores semânticos: ao pontuarem o significado afetivo de uma palavra em três variáveis conforme o estímulo provocado — valência/prazer, excitação/intensidade e dominância/grau de controle exercido —, Osgood, Suci e Tannenbaum (1957) logo perceberam que essa sinalização poderia ser utilizada para traduzir uma palavra em um ponto em um espaço tridimensional (HELLRICH, 2019).

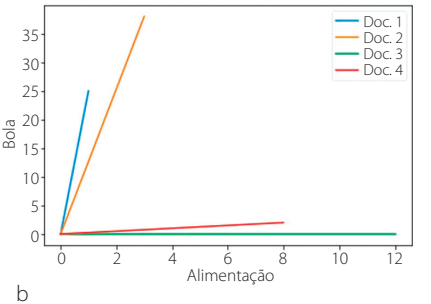
Nesse contexto, as abordagens lógicas utilizadas até então deram vez às representações que simulavam de alguma forma a maneira como as pessoas costumam utilizar o idioma, estabelecendo e determinando relações de tipos e níveis diferentes entre cada palavra.

Atualmente, modelos de semântica vetorial podem ter múltiplas dimensões e incluir características obtidas por algoritmos de aprendizado não supervisionado.

Os vetores podem ser expressos também na forma de uma tabela de ocorrência das palavras em diferentes textos, conhecida como matriz termo-documento (Figura 1a), na qual cada linha expressa uma palavra distinta presente nos documentos e determina a dimensão dos vetores, enquanto as colunas expõem os documentos do conjunto de dados. Cada vetor é uma coluna, que associa determinado documento à ocorrência das palavras de seu vocabulário. Obviamente, por se tratar de um vetor, este também pode ser expresso na forma de gráficos (Figura 1b).

	Doc. 1	Doc. 2	Doc. 3	Doc. 4
Alimentação	1	3	12	8
Bola	25	38	0	2

a



b

Figura 1. (a) Matriz termo-documento. (b) Vetores da matriz termo-documento representados graficamente.

Percebe-se, principalmente pela Figura 1b, que existem dois pares de documentos com direções bastante distintas entre si — assim, enquanto os documentos 3 e 4 poderiam representar uma coleção de artigos nutricionais, 1 e 2 poderiam fazer parte de uma coletânea de matérias esportivas.

Tal técnica poderia ser aplicada às palavras e aos seus significados, em vez de relacioná-las a um documento, em que os vetores passam a ser as linhas (palavras) e sua dimensão determinada pela quantidade de colunas (significados). Além disso, podemos modificar a matriz termo-documento para atender a outro objeto de comparação entre as palavras, utilizando tanto linhas quanto colunas como palavras e a célula de cruzamento entre elas representando a frequência com que ambas as palavras aparecem juntas em um dado contexto, como o documento ou um recorte menor de texto (p. ex., as 10 palavras mais próximas), tornando-se conhecida como matriz palavra-palavra ou termo-contexto (JURAFSKY; MARTIN, 2019).

Uma vez que os documentos (colunas da matriz) podem ter dimensões muito maiores (linhas da matriz, palavras), tanto a matriz termo-documento quanto o gráfico dos vetores se tornam métodos pouco eficientes para comparar cada um dos vetores (colunas), já que, de fato, qualquer vetor com mais de três dimensões seria impraticável de representar graficamente.

Assim, deve-se buscar formas mais eficientes de comparar cada um dos vetores para determinar a similaridade entre eles, ou seja, entre os documentos ou as palavras.

Para tanto, medidas de similaridade comumente utilizam o produto interno entre vetores para enfatizar dimensões em que ambos os vetores têm valor

elevado e suprimir dimensões em que o valor é baixo. Assim, vetores distantes tendem a um produto interno baixo; se forem ortogonais, o produto será 0; e vetores próximos terão produto interno elevado (JURAFSKY; MARTIN, 2019).

Para evitar que palavras com frequência elevada tenham maior relevância que palavras de baixa frequência, pela magnitude dos vetores produzidos, o produto interno simples é normalmente substituído pelo cosseno (similaridade pelo cosseno) entre vetores, conforme a Equação (1).

$$\cos\theta = \frac{\boldsymbol{A}.\boldsymbol{B}}{|\boldsymbol{A}| \cdot |\boldsymbol{B}|} = \frac{\sum_{i=1}^{N} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{N} A_i^2} \cdot \sqrt{\sum_{i=1}^{N} B_i^2}} \quad (1)$$

onde:

- $\boldsymbol{A}$ e $\boldsymbol{B}$: representam dois vetores quaisquer;
- θ: ângulo formato entre os vetores;
- N: número de dimensões do vetor.

O modelo de TF-IDF (*term frequency-inverse document frequency*), por sua vez, relaciona a frequência de uma dada palavra em todo o conjunto de dados/documentos com um peso determinado pelo inverso da frequência de documentos que não apresentam palavra, como mostrado na Equação (2). Isso porque palavras pouco frequentes podem não dispor de informações suficientes para compreender a relação entre elas, mas as excessivamente frequentes, como artigos e preposições, não adicionam nenhuma informação a respeito de sua similaridade com outras palavras.

$$tfidf_{d,p} = tf_{p,d} \,.\, idf_p \quad (2)$$

onde:

- $tf_{p,d}$: frequência da p-ésima palavra no documento;
- id_{fp}: quantidade de documentos que têm a p-ésima palavra.

O TF-IDF, no entanto, costuma produzir distribuições de palavras dispersas e com muitas instâncias nulas, sobretudo pelo fato de a maioria das palavras não pertencer a contextos similares, em razão da utilização de matrizes palavra-palavra como meio de representação (JURAFSKY; MARTIN, 2019).

Na verdade, esses modelos dispõem de pouca quantidade de informação relevante, apesar da vasta quantidade de dados existentes, especialmente no caso do TF-IDF, em que a dimensão de cada vetor será igual à quantidade de palavras. Mesmo assim, essa maior quantidade de dados utilizará mais recursos computacionais, tanto de memória para alocá-los quanto de processamento.

Fique atento

Apesar de o modelo BoW fazer uso de vetores, estes não têm qualquer informação sobre o contexto e a similaridade das palavras e seus documentos, diferentemente da *word embeddings*, que opera vetores semânticos, com informações mais complexas sobre os termos relacionados.

2 Algoritmos para *word embeddings*

Entre os modelos de *word embeddings* existentes, pode-se destacar o word2vec, o GloVe e o fastText, os quais, comparados ao TF-IDF, lidam com vetores e matrizes densas (com pouca quantidade de instâncias nulas) e com dimensão normalmente menor que 1.000, enquanto o modelo TF-IDF facilmente ultrapassaria 50 ou 100 mil dimensões, já que corresponde a mesma quantidade de termos existentes.

Em geral, soluções que utilizam vetores densos apresentam um desempenho superior àqueles que fazem uso de vetores esparsos (com muitas instâncias nulas), mesmo porque a quantidade excessiva de parâmetros facilita a superadaptação do modelo durante o treinamento (JURAFSKY; MARTIN, 2019).

Anteriormente a esses algoritmos, era comum o uso de algoritmos simples associados a volumosos conjuntos de dados para obter a melhor relação de desempenho e custo computacional possível. Assim, mesmo sem considerar relações mais complexas das palavras com o texto e entre elas mesmas, o desempenho ainda poderia ser superado pela ampla diferença de custo computacional (MIKOLOV *et al*., 2013a).

Além da melhora de desempenho, outros fatores contribuíram para a utilização de *word embeddings*, em razão das restrições intrínsecas aos demais modelos, por exemplo, o desempenho limitado pelo baixo volume de dados em aplicações de reconhecimento de voz, o volume de dados insuficiente para traduções de idiomas menos comuns, etc. (MIKOLOV *et al*., 2013b).

O potencial desses algoritmos ficou ainda mais claro quando se observou que as relações aritméticas entre os vetores criados possibilitavam, entre outros, fazer comparações do tipo: vetor('Rei') – vetor('Homem') + vetor('Mulher') ≅ vetor('Rainha') (MIKOLOV *et al*., 2013a), literalmente efetuando operações entre palavras, dado o potencial de representação e aprendizado dessas novas técnicas.

A operação vetorial citada constitui um exemplo de analogia entre palavras (*word analogies*), curiosamente observada na representação vetorial de palavras quando obtidas por modelos capazes de detectar características sintáticas e semânticas das palavras.

Palavras similares tendem a apresentar atributos similares e, portanto, a se posicionar próximas no espaço de atributos quando exibidas em um gráfico (GOLDBERG; LEVY, 2014). A Figura 2 demonstra a relação entre os vetores do exemplo Rei/Rainha citado.

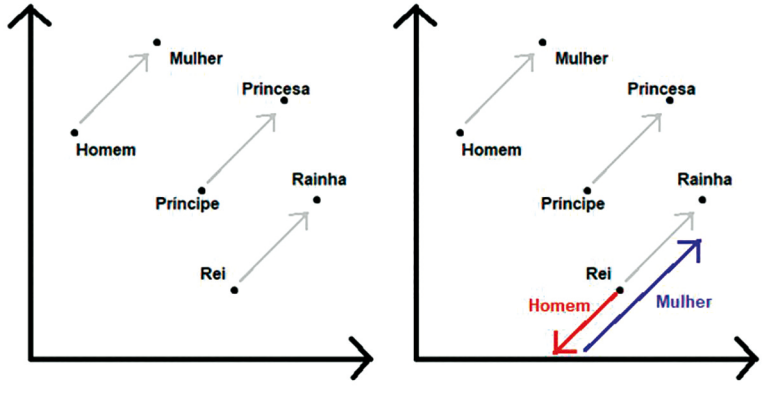


Figura 2. Analogia de palavras utilizando representação vetorial.

Facilmente percebemos que, se subtraído o vetor da palavra Homem do vetor da palavra Rei, seria obtida uma palavra desconhecida próxima ao eixo x de atributos do gráfico, mas, se após essa subtração fosse somado o vetor Mulher, a magnitude, a direção e o sentido deste vetor seriam similares à distância existente entre a palavra Homer e Mulher, observada entre as palavras Rei e Rainha. Espera-se, assim, que o resultado dessas operações seja um vetor muito próximo da palavra Rainha.

Uma operação similar poderia ser feita com as palavras Príncipe e Princesa, assim como muitas outras analogias, que não necessariamente relacionassem as mesmas características de gênero e título das palavras como nos casos apresentados.

Outro exemplo clássico de analogia consiste na relação entre os países e suas respectivas capitais. Por exemplo, seria possível obter um vetor muito próximo do vetor da palavra França se fosse subtraído o vetor da palavra Londres do vetor da palavra Inglaterra, e, posteriormente, somado ao vetor da palavra Paris. Nesse caso, espera-se que a distância, a direção e o sentido entre os vetores das palavras Londres e Inglaterra sejam similares aos encontrados nas palavras Paris e França.

O algoritmo word2vec utiliza tarefas de classificação para prever a probabilidade com que uma palavra aparece próxima de outra, definindo-a como a característica incorporada à palavra. Trata-se, portanto, de um método estatístico e de aprendizado supervisionado.

Ao representar a palavra por um vetor probabilístico, em substituição ao clássico vetor `one-hot-encoder` (no qual cada classe de um atributo categórico se torna uma coluna/atributo de valor binário, indicando ou não a presença da categoria), podemos determinar uma função de custo e incorporar o modelo a redes neurais com aprendizado por *backpropagation* (em que o erro dos neurônios de saída é retropropagado para as camadas anteriores para ajustar parcialmente os pesos buscando melhorar a previsão).

Para reduzir o custo computacional de uma rede neural com vocabulários muito extensos, o word2vec utiliza apenas duas camadas, recebendo como entrada o conjunto de textos (*corpus*) para retornar na saída um vetor multidimensional. Esse algoritmo é muito empregado em aplicações que objetivam encontrar palavras ou sentenças análogas.

O GloVe (do inglês, *global vectors*), por sua vez, compreende um algoritmo de aprendizado não supervisionado e adiciona as informações do contexto local, como no word2vec, mas também do contexto global. Assim, ao contrário do word2vec, o GloVe consegue identificar palavras comuns no contexto global, como artigos e preposições.

No GloVe, o contexto global é determinado relacionando as palavras à matriz de coocorrência. Diferentemente do word2vec, que mostra o quão similar é o contexto de duas palavras, o GloVe demonstra a probabilidade de duas palavras aparecerem juntas.

O algoritmo fastText é similar ao word2vec, porém consegue generalizar com maior eficiência para palavras ainda não presentes no modelo de representação vetorial, o que é especialmente vantajoso em aplicações de tradução, já que muitas línguas não dispõem de conjuntos de dados e referências para treinamento quanto o inglês (BHATTACHARJEE, 2018). Para tanto, o fastText utiliza como unidade principal de aprendizado os caracteres, e não a palavra em si, tornando a palavra o contexto desses caracteres.

Além de possibilitar boas generalizações para palavras desconhecidas, o fastText aproveita melhor os conjuntos e textos à disposição, já que utiliza cada caractere como fonte de dados.

Por fim, entre tantos modelos pré-treinados disponíveis, o BERT (*bidiretional encoder representations from transformers*, ou "representação de codificador bidirecional de transformadores"), aplicado desde 2019 no motor de busca da Google, tem se destacado, sendo utilizado pela empresa para melhorar o refinamento das pesquisas considerando o contexto das palavras, e não apenas as palavras-chave.

Originalmente, o BERT é pré-treinado em conjuntos de dados não rotulados baseados principalmente na Wikipédia, o que diminui consideravelmente o tempo de treinamento de novos modelos. Além disso, trata-se da primeira implementação de *word embeddings* verdadeiramente bidirecional, que permite criar representações para as palavras tanto do contexto prévio quanto do contexto seguinte da sentença ou do texto (DEVLIN *et al*., 2018).

3 Implementação de *word embeddings*

Para implementar qualquer algoritmo de *word embeddings*, tarefas de pré-processamento e adequação dos textos e documentos utilizados são normalmente necessárias. Em línguas similares ao português, por exemplo, é comum diferenciar letras maiúsculas de minúsculas para representar início de frases, siglas, nomes e outras situações. No entanto, a máquina compreende os caracteres maiúsculos e minúsculos como letras diferentes.

Salvo algumas exceções, as palavras precisam estar uniformizadas para um único formato, a fim de poderem ser interpretadas sempre da mesma maneira, e ser tokenizadas, formando um saco de palavras, removidas as partes vazias do texto e outros elementos inválidos como rótulos XML/HTML, capazes de acompanhar o texto, e, em alguns casos, palavras denominadas *stop words*, de frequência elevada e que oferecem pouco conhecimento específico a respeito do texto ou do contexto (p. ex., artigos, preposições e outros elementos textuais, como "de, a, o, que, e, do, da, em, ...") (BHATTACHARJEE, 2018).

Exemplo

Exemplo 1 — pseudocódigo de pré-processamento

Ler todas as linhas do arquivo.
Para cada linha, fazer:
- Forçar para letra minúscula.
- Remover pontuações.
- Remover marcações de nova linha.
- Remover marcações de tabulação.
- Remover palavras presentes em *stop_words.*

Reordenar as linhas de maneira aleatória.
Dividir o arquivo em dois:
- Parte para um arquivo contendo textos de treinamento.
- Parte para um arquivo contendo texto de teste.

A partir dos conjuntos de treinamento, é possível implementar um teste um algoritmo de word2vec, que se divide nos tipos CBOW (*continuous bag-of-words,* ou "saco de palavras contínuo") e Skip-Gram.

O método CBOW prevê a palavra dado um contexto, motivo pelo qual recebe múltiplas entradas, mas entrega apenas uma única saída. Na Figura 3a, há um procedimento em que uma janela de 2 palavras, tanto anteriores (w(t – 2) e w(t – 1)) quanto posteriores (w(t + 1) e w(t +2)), é utilizada para prever o alvo (w(t)).

Já o Skip-Gram prevê um contexto dada uma palavra, e, ao contrário do método CBOW, recebe uma única entrada (palavra) para retornar um conjunto de palavras que pertencem a esse contexto. Na Figura 3b, uma palavra (w(t)) é utilizada para determinar o contexto mais provável em que estaria inserida para uma janela de 2 palavras anteriores w(t +2) e (w(t + 1) e posteriores (w(t + 1) e w(t +2)) à palavra de entrada.

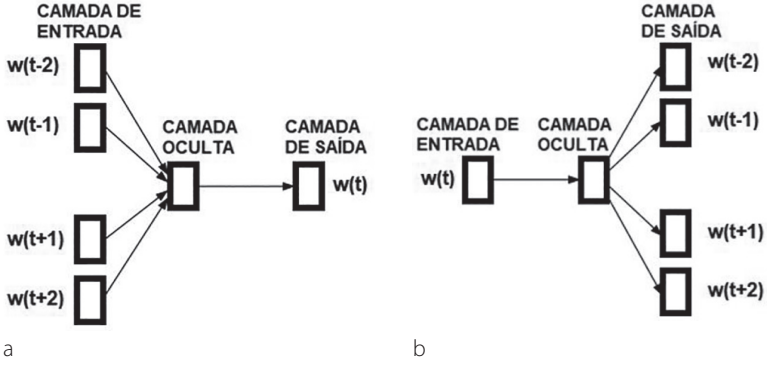


Figura 3. (a) Método CBOW. (b) Método Skip-Gram.
Fonte: Adaptada de Mikolov *et. al.* (2013a).

Os dois algoritmos lidam com um aprendizado do tipo supervisionado, exigindo que os rótulos para cada uma das palavras do vocabulário já sejam previamente conhecidos para treinar o modelo. Nesse caso, os rótulos se referem às representações utilizadas para cada um dos termos ou das palavras.

Entretanto, na maioria casos, isso não será possível, limitando as possibilidades de aplicação desse modelo de PLN. Para contornar esse problema, são criados rótulos falsos, escolhendo para cada termo a quantidade de termos vizinhos mais próximos a considerar, que corresponderão às células de saída da rede neural. Assim, o Skip-Gram é um híbrido entre a *word embeddings* e o modelo n-gram.

Durante o treinamento, a camada oculta da rede neural ajustará os pesos de cada conexão a fim de minimizar o erro obtido pelos rótulos falsos. Ao final, cada uma das células de saída representa a probabilidade com que duas palavras pertencem ao mesmo contexto, conforme mostrado na Equação (3).

$$J(\theta) = \prod_{t=1}^{T} \prod_{\substack{-c \le j \le c, \\ j \neq 0, \\ 0 < t+j \le T}} \log p\left(w_{t+j} | w_t, \theta\right) \tag{3}$$

onde:

- T: tamanho do vocabulário;
- w_t: t-ésima palavra;
- θ: parâmetros.

Os pesos determinam o impacto de cada palavra sobre a determinação do contexto previsto, no caso do algoritmo de CBOW, ou o peso dado ao contexto determina as palavras que pertencem a ele, no algoritmo de Skip-Gram. Esses pesos são justamente a representação vetorial que se pretende obter das palavras.

O treinamento de uma rede neural, no word2vec, visa a maximizar a função da Equação (3), ou seja, busca-se associar uma palavra ao seu contexto mais provável. No entanto, a Equação (3) pode ser substituída pela Equação (4), com o uso de funções logarítmicas que facilitem o cálculo computacional, já que os produtórios podem ser calculados como somatórios; ao contrário da Equação (3), a Equação (4) deve ser minimizada.

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \le j \le c, \\ j \neq 0, \\ 0 < t+j \le T}} \log p(w_{t+j} | w_t, \theta) \tag{4}$$

No algoritmo de CBOW, devemos fazer um procedimento similar, para criar rótulos "falsos" e permitir que a rede neural encontre os pesos que relacionam entradas e saída, já que impede que todos os vetores tenham o mesmo valor, o que normalmente é feito pela seleção de palavras fora do contexto de maneira aleatória (GOLDBERG; LEVY, 2014). Novamente, camada oculta é substituída por uma camada simples de projeção, o que contribui tanto no CBOW quanto no Skip-Gram para a redução do custo computacional.

Exemplo

Exemplo 2 — pseudocódigo de um algoritmo de CBOW

```
01 Para cada sentença do conjunto de treinamento, fazer:
02   Para cada palavraCentral da sentença, fazer:
03    janela = palavras vizinhas à direita e esquerda da palavraCentral
04     Para cada palavraJanela da janela, fazer:
05       Para cada dimensão dim do vetor, fazer:
06         camSaída[dim] += vetores[palavraJanela][dim]
07       Para cada palavraAmostra sortida com a palavraCentral:
08       Se palavraAmostra é igual a palavraCentral, então:
09         rótulo = verdadeiro
10       se não:
11         rótulo = falso
12       Para cada dimensão dim do vetor, fazer:
13           f += camSaída[dimensão] * camOculta[palavraAmostra][dim]
14         g = gradiente de f em relação ao rótulo
15       Para cada dimensão dim do vetor fazer:
16           erroSaída[dim] += g * camOculta[palavraAmostra][dim]
17         Para cada dimensão dim do vetor fazer:
18           camOculta[palavraAmostra][dim] += g * camSaída[dim]
19     Para cada palavraJanela da janela fazer:
20         Para cada dimensão dim do vetor fazer:
21           vetores[palavraAmostra][dim] += erroSaída[dim]
```

O pseudocódigo do Exemplo 2 implementa um algoritmo de CBOW, seguindo as etapas de propagação direta (*feedforward*) entre as linhas 4 e 7, passando pela amostragem negativa a partir da linha 8 e de retropropagação (*backpropagation*) para correção do erro de saída entre as linhas 19 e 21 (HIROSE *et al*., 2016).

Exemplo

Exemplo 3 — a famosa frase de Winston Churchill como exemplo
Texto: "Ter sucesso é falhar repetidamente, mas sem perder o entusiasmo".

Tupla 01: "**[Ter sucesso é]** falhar repetidamente, mas sem perder o entusiasmo."
Tupla 02: "**[Ter sucesso é falhar]** repetidamente, mas sem perder o entusiasmo."
Tupla 03: "**[Ter sucesso é falhar repetidamente]**, mas sem perder o entusiasmo."
Tupla 04: "Ter **[sucesso é falhar repetidamente, mas]** sem perder o entusiasmo."
Tupla 05: "Ter sucesso **[é falhar repetidamente, mas sem]** perder o entusiasmo."
Tupla 06: "Ter sucesso é **[falhar repetidamente, mas sem perder]** o entusiasmo."
Tupla 07: "Ter sucesso é falhar **[repetidamente, mas sem perder o]** entusiasmo."
Tupla 08: "Ter sucesso é falhar repetidamente, **[mas sem perder o entusiasmo.]**"
Tupla 09: "Ter sucesso é falhar repetidamente, mas **[sem perder o entusiasmo.]**"
Tupla 10: "Ter sucesso é falhar repetidamente, mas sem **[perder o entusiasmo.]**"

Utilizando a Equação (3), da primeira à sexta tupla, (t = {1…6}, T = 6), considerando uma janela de tamanho 2 (c = ±2), são calculadas as probabilidades das c-palavras mais próximas à esquerda e à direita da palavra alvo (w_t). No Exemplo 3, as palavras pertencentes à janela de cada tupla foram destacadas e delimitadas entre [ ], com maior destaque para a palavra-alvo em vermelho de cada caso. Então, os valores encontrados são incorporados às palavras para servir de rótulo para a tarefa de classificação executada pela rede neural.

Tomando a tupla 04 como exemplo, temos as informações disposta no Quadro 1.

Quadro 1. Somatório da tupla 04 do exemplo 3

$p(w_{4-2} \mid w_4)$	$p(w_2 \mid w_4)$	sucesso
$p(w_{4-1} \mid w_4)$	$p(w_3 \mid w_4)$	é
$p(w_{4+1} \mid w_4)$	$p(w_5 \mid w_4)$	repetidamente,
$p(w_{4+2} \mid w_4)$	$p(w_6 \mid w_4)$	mas

A probabilidade de cada instância na tabela utiliza uma função do tipo *softmax* que corresponde à razão da Equação (5):

$$p(w_{t+j}|w_t) = \frac{e^{w_{t+j} \cdot w_t}}{\sum_{i=1}^{T} e^{w_i \cdot w_t}} \quad (5)$$

Os extremos de uma função (máximo e mínimo) correspondem ao ponto que a sua derivada se iguala a zero. Assim, para encontrar o máximo local da Equação (3) ou o mínimo local da Equação (4), será necessário utilizar uma função gradiente que facilita essa busca para variáveis vetoriais, demonstrando a direção em que variam os parâmetros associados à função, o que a tornando cada vez maior.

Do Exemplo 3, ainda é possível observar detalhes que precisamos considerar na preparação do texto e dos documentos que serão utilizados. Por exemplo, as pontuações existentes no texto não foram removidas, logo a palavra e a vírgula após "repetidamente" e o ponto após "entusiasmo" serão compreendidos como parte desses termos. Se, em textos maiores, houver novamente essas palavras, mas sem pontuação associada, será criado um vocabulário com duas palavras para representar "repetidamente" (com e sem vírgula) e duas para representar "entusiasmo" (com e sem ponto).

Ainda, são exigidos outros cuidados e análises para obter um bom desempenho, como o tamanho da janela de vizinhança: se muito pequena, o contexto pode não ser devidamente observado, mas, se muito grande, muitas palavras estarão associadas a sentenças e contextos diferentes.

Por fim, os métodos CBOW e Skip-Gram apresentam diferenças de desempenho em geral resumidas da seguinte maneira: enquanto o CBOW tem a vantagem de efetuar o processo de treinamento mais rapidamente e lidar melhor com palavras de alta frequência, o Skip-Gram apresenta melhor atuação para pequenos volumes de dados e palavras de baixa frequência.

Independentemente da abordagem adotada, o word2vec tem como intuito encontrar representações vetoriais para as palavras capazes de incorporar as características presentes no contexto em que estão inseridas. Assim, os modelos de *word embeddings* permitem comparar palavras em um sentido mais amplo e mais próximo da realidade na qual se utilizada o idioma.

Graças a essas representações, tornaram-se possíveis a execução de operações de analogia e similaridade com grande eficiência, além de muitas aplicações complexas, desde a análise de sentimentos embutidos em uma

sentença e a tradução de frases e palavras até a interação autônoma de uma máquina com uma pessoa se expressando de forma muito similar a como se faria fisicamente.

Referências

BHATTACHARJEE, J. *FastText quick start guide*: get startedwith facebook's library for text representation and classification. Birmingham: Pack Publishing, 2018.

DEVLIN, J. *et al. BERT*: pre-training of deep bidirectional transformers for language understanding. 2018. Disponível em: https://arxiv.org/pdf/1810.04805.pdf. Acesso em: 27 abr. 2020.

GOLDBERG, Y.; LEVY, O. *Word2vec explained*: deriving mikolov et al.'s negative-sampling word-embedding method. 2014. Disponível em: https://arxiv.org/pdf/1402.3722.pdf. Acesso em: 27 abr. 2020.

HELLRICH, J. *Word embeddings*: reliability & semantic change. Amsterdam: IOS, 2019.

HIROSE, A. *et al.* (ed.). *Neural information processing*. Kyoto: Springer, 2016.

JURAFSKY, D.; MARTIN, J. H. *Speech and language processing*: an introduction to natural language processing, computational linguistics, and speech recognition. 3rd ed. 2019. Disponível em: https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf. Acesso em: 27 abr. 2020.

MIKOLOV, T. *et al. Distributed representations of words and phrases and their compositionality*. 2013b. Disponível em: https://arxiv.org/pdf/1310.4546.pdf. Acesso em: 27 abr. 2020.

MIKOLOV, T. *et al. Efficient estimation of word representations in vector space*. 2013a. Disponível em: https://arxiv.org/pdf/1301.3781.pdf. Acesso em: 27 abr. 2020.

OSGOOD, C. E.; SUCI, G. J.; TANNENBAUM, P. H. *The Measurement of meaning*. Champaign: University of Illinois Press, 1957.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

Muitas aplicações até então complexas ou improváveis puderam ser desenvolvidas com o auxílio de modelos que implementam o conceito de *word embeddings*. O grande diferencial, se comparado aos demais modelos de processamento de linguagem natural, está nas características reconhecidas pelo modelo, que envolvem o contexto em que as palavras estão inseridas, o qual se aproxima muito mais da forma como as pessoas compreendem o idioma.

Assista à Dica do Professor para conhecer os conceitos básicos de representação de palavras por vetores, em especial *word embeddings*, e os métodos desenvolvidos a partir desse conceito.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Métodos de *word embeddings* possibilitaram, entre outras coisas, o processamento de textos, considerando relações de similaridade semântica. Essa estrutura está baseada na hipótese distributiva, segundo a qual é possível afirmar que:**

A) a frequência com que palavras sinônimas aparecem tem relação direta com o ambiente em que se fazem presentes.

B) não há qualquer relação entre a frequência das palavras e o contexto em que se encontram.

C) significados diferentes podem ser encontrados nas mesmas palavras conforme o contexto.

D) quanto mais disperso um conjunto de palavras, mais difícil de se determinar o contexto e, portanto, o sentido dado.

E) quanto mais disperso um conjunto de palavras, mais fácil de se determinar o contexto e, portanto, o sentido dado.

2) **Grande parte das aplicações de processamento da linguagem natural envolve a comparação de textos e palavras. Nesse contexto, para medir a proximidade de dois vetores semânticos, pode ser utilizado(a):**

A) o cosseno.

B) a hipótese distributiva.

C) o comprimento dos vetores.

D) o algoritmo de *backpropagation*.

E) a tangente.

3) **A partir da hipótese distributiva, muitos métodos de processamento da linguagem natural foram desenvolvidos utilizando o conceito de representação vetorial contínua. Entre esse métodos, um dos primeiros e mais utilizados é o *word2vec*. Sobre tal método, assinale a alternativa correta:**

A) O *word2vec* é um método estatístico.

B) O *word2vec* utiliza a matriz de co-ocorrência para comparar dois vetores de um texto.

C) O *word2vec* utiliza o conceito de saco de palavras (*bag-of-words*).

D) O *word2vec* é derivado de outro método, conhecido como *GloVe*.

E) O *word2vec* utiliza apenas relações sintáticas nas representações vetoriais.

4) Dois métodos diferentes podem ser utilizados para se aplicar o *word2vec*: Skip-Gram e CBOW. Escolha a alternativa que apresenta corretamente a principal diferença entre ambos:

A) O CBOW prevê uma palavra a partir de um contexto, enquanto o Skip-Gram faz o inverso.

B) O Skip-Gram utiliza redes neurais, enquanto o CBOW utiliza regressão logística.

C) O CBOW utiliza redes neurais, enquanto o Skip-Gram utiliza regressão logística.

D) O Skip-Gram converge mais rapidamente do que o CBOW.

E) O CBOW tem desempenho melhor para palavras de baixa frequência, enquanto o Skip-Gram tem desempenho melhor para palavras de alta frequência.

5) O método Skip-Gram é usado para, a partir de uma palavra de entrada, prever um vetor de outras palavras que fazem parte do seu contexto.

Dada a frase a seguir:

"A representação vetorial de textos revolucionou o processamento de linguagem natural"

Determine o contexto da palavra *textos*, considerando uma janela de tamanho 2, e assinale a alternativa correta:

A) vetorial; de; revolucionou; o.

B) A; representação; vetorial; de; revolucionou; o; processamento; de.

C) A; representação; vetorial; de; revolucionou; o; processamento; linguagem.

D) vetorial; de.

E) de; revolucionou.

Na prática

A representação vetorial facilita o cálculo de similaridade entre as palavras e as sentenças. Com isso, muitas aplicações são possíveis, como tradução, análise de sentimentos, classificação textual, entre outras. Os algoritmos utilizados nessas aplicações para processamento de linguagem natural normalmente demandam grande volume de dados e muitas iterações de treinamento, principalmente em virtude de a linguagem se organizar de forma complexa e envolver múltiplas dimensões para os vetores que a representam.

O *word2vec* é uma das formas mais populares de implementação de *word embeddings*. No entanto, desde o *word2vec*, muitos outros algoritmos foram desenvolvidos — é o caso, por exemplo, do *fastText* e do *GloVe*. Conhecer as vantagens e as desvantagens de cada abordagem é primordial para facilitar o processo de treinamento e obter modelos mais adequados a cada aplicação.

Na Prática, a partir de um cenário fictício, veja uma comparação entre os modelos *GloVe* e *fastText*.

A ESCOLHA DO MODELO ADEQUADO:

GLOVE VERSUS FASTTEXT

O programador Henrique, ao tentar desenvolver um aplicativo de tradução do inglês para o português, está avaliando **qual é o algoritmo mais adequado para esse tipo de problema**.

Apesar de existirem muitos *corpus* conhecidos e adequados para o treinamento de qualquer algoritmo que seja, Henrique sabe que a maioria deles é em inglês e que existe uma quantidade muito menor de *corpus* para o português.

Independentemente do algoritmo a ser escolhido, o programador sabe que o maior problema estará **na dificuldade de encontrar dados suficientes em português para o processo de treinamento**. Preencher essa defasagem certamente deverá ser a prioridade dessa seleção.

Então, ele resolveu buscar algoritmos similares ao *word2vec* e ainda mais eficientes, encontrando os seguintes:

GLOVE

O *GloVe* amplia o contexto a ser considerado pelas palavras ao utilizar uma **abordagem global**, contabilizando quantas vezes duas palavras aparecem em um mesmo contexto em vez de considerar apenas o contexto das palavras mais próximas.

FASTTEXT

O *fastText* é ainda mais parecido com o *word2vec* clássico, mas, em vez de considerar as palavras como um todo, o algoritmo **divide as palavras em partes menores**, permitindo que o significado de uma palavra desconhecida no vocabulário do modelo possa ser aproximado pelos vetores de partes de outras palavras similares combinadas.

A partir de seus estudos, Henrique ficou confiante, pois **o modelo *fastText* atende justamente o seu caso**, já que o algoritmo é capaz de **presumir algumas definições combinando partes de palavras existentes no seu vocabulário**, o que diminui o impacto da escassez de conjuntos de treinamento e dados para o português.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

word2vec e sua importância na etapa de pré-processamento

No *link* a seguir, acompanhe uma revisão dos conceitos básicos de *word2vec* (CBOW e Skip-Gram) comparados aos modelos antigos que utilizavam, na maioria, técnicas de *one-hot-encoder*.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Avaliando atributos para a classificação de estrutura retórica em resumos científicos

Neste artigo, são utilizados *word2vec*, *GloVe*, entre outros modelos de *word embeddings*, para classificar e relacionar estruturas retóricas presentes em resumos científicos. Confira.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

word2vec e TensorFlow

Este vídeo traz uma breve apresentação do TensorFlow, uma biblioteca de código aberto utilizada na criação e no treinamento de redes neurais, além de alguns conceitos básicos de *word2vec*. Aproveite.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.