


PRÉ-PROCESSAMENTO DE DADOS - LINGUAGEM NATURAL

A PREPRINT

 **Nome***
Department
University
Pittsburgh, PA 15213
email@email.email

 **Nome**
Department
University
Santa Narimana, Levand
email@email.email

December 15, 2022

ABSTRACT

Este documento apresenta as principais técnicas utilizadas para o pré-processamento de textos.

Keywords First keyword · Second keyword · More

1 Introdução

Algumas das dificuldades no aprendizado de máquina sobre tarefas relacionadas a textos, é como representar o texto para que um modelo aprenda a solucionar o problema de linguagem natural. Estas representações tem como objetivo apresentar os textos em formato numérico para possibilitar manipulações matemáticas e contextualizar estes como valores de entrada para que um modelo aprenda a tarefa.

2 Pré-processamento do Texto

O Pré-processamento do texto equivale fazer alterações em textos para que um algoritmo possa construir um vocabulário e algum modelo utilize-o para resolver a tarefa. Este pré-processamento normaliza o texto sobre algum critério definido, para criar um vocabulário que tenha sentido para a tarefa e seja eficiente na quantidade de itens dentro deste vocabulário. Um exemplo custoso de construção deste, é obter a mesma palavra escrita de formas ou letras diferente entre os exemplos de texto e introduzir todas estas possibilidades de variações pode aumentar significativamente o tamanho do vocabulário. Logo, o pré-processamento do texto é necessário para reduzir estas combinações, minimizando a perda de contexto do texto.

2.1 Caracteres

O pré-processamento à nível de caractere, utiliza de técnicas de padronização ou remoção destes no intuito de remover pontuações ou símbolos que não são úteis para contextualização para tarefa ou transformar em uma única palavra, variações da mesma palavra escrita de forma diferentes.

2.1.1 Remoção

A remoção de acentuações em caracteres, ou caracteres especiais (&\$#), pontuações (!,?,") e emojis é comum para reduzir o texto sem perder o sentido do contexto. Segue o exemplo a seguir

De: Já é de manhã ? Não acredito que que perdi o nascer do Sol.
Para: Ja e de manha Nao acredito que que perdi o nascer do Sol

*algo

2.1.2 Padronização/Normalização

Este tem como objetivo padronizar os caracteres das palavras sob a mesma fonte ou contexto. (ex.: De: Sol, sOl, \$ol; Para: sol). O objetivo é não ter a mesma palavra escrita com caracteres diferentes.

De: Já é de manhã ? Não acredito que que perdi o nascer do \$ol.
Para: já é de manhã ? nao acredito que perdi o nascer do sol.

2.1.3 Expansão

Em algumas línguas, a palavra pode ser abreviada e é interessante utilizar as palavras na sua forma expandida para padronização em que o algoritmo possa identificar o contexto do texto

De: Y'all can't expand contractions I'd think! You wouldn't be able to. How'd you do it?
Para: You all cannot expand contractions I would think! You would not be able to. How did you do it?

2.2 Palavra

Outras técnicas de pré-processamento de texto constitui à nível de palavra. Em alguns casos, é interessante reduzir, remover, ou padronizar a palavra para que esta tenha uma contextualização geral no texto sem que perca o sentido principal deste para a tarefa.

2.2.1 Stemização

A stemização é o processo de padronizar as palavra a reduzir em a seu tronco, base ou raiz, removendo as inflexões da palavra

De: Amigos, já é de manhã ? Não acredito que perdi o nascer do sol.
Para: Amig, já é de manhã ? Não acredito que perd o nasc do sol.

2.2.2 Lematização

A lematização é o processo de padronizar as palavra na sua forma base e agrupar no mesmo sinônimo da palavra, alguns exemplos são: contexto gramatical ignorado para sua forma original (pessoas -> pessoa); Verbos normalizados para o tempo presente (aconteceu -> acontece); sinónimos unificados (melhor -> bom).

De: O passado foi o que pensamos quando aconteceu os eventos. Mas o futuro, faria pensar o que seria.
Para: O passado é o que pensa quando acontece o evento. Mas o futuro, faz pensa o que é.

2.2.3 Remoção

É comum remover palavras menos significativas chamadas 'stop words', que ajudam apenas na gramática de algumas palavras mas não influencia no significado principal desta.

De: O passado foi o que pensamos quando aconteceu os eventos. Mas o futuro, faria pensar o que seria.
Para: passado foi que pensamos quando aconteceu eventos. Mas futuro, faria pensar que seria.

3 Vocabulário

O Vocabulário corresponde a um conjunto de caracteres ou palavras indexadas como dicionário, para representar textos como uma lista ordenada de índices deste dicionário. Este tem como objetivo representar o texto de forma numérica ao passo de possibilitar que algoritmos possam manipular ou contextualizar estes índices como vetores para a realização de dada tarefa utilizando modelos de aprendizado de máquina.

3.1 Tokenização do texto

A tokenização é o processo de transformar o texto em uma lista de índices do vocabulário. Este transforma caracteres ou palavras, chamados tokens, utilizando algum critério de separação no intuito de substituir estes por índices do vocabulário para criar uma lista ordenada e enumerada representando o texto passado

Vocabulário: {"Quando": 1, "olhei": 2, "para":3, "você":4, ",":5, "não":6, "acreditei":7, "no":8, "que":9, ".":10}

De: Quando olhei para você, não acreditei no que olhei.

Para: [1,2,3,4,5,6,7,8,9,2,10]

3.1.1 Byte-Pair Encoding (BPE)

BPE Sennrich et al. [2015] é uma técnica de pos-tokenização que caracteriza palavras desconhecidas em um conjunto de sub palavras conhecidas. O algoritmo BPE agrupa as palavras de um vocabulário com sua respectiva frequências contabilizada sob um grupo de exemplos de texto. Após construir estes grupos, estas palavras são separada por caracteres e é contabilizado a frequência dos pares destes para cada palavra. Os pares mais frequentes é escolhido para ser concatenado e virar um conjunto de caracteres como uma nova "palavra" ou símbolo, assim este processo continua agrupando estes caracteres ou grupo de caracteres por par de máxima frequência a cada interação. Esta interações continuam até certo critério pré-definido de parada, para construir um vocabulário final que reduza a quantidade de palavras ou represente palavras desconhecidas como "sub-palavras" conhecidas.

(Palavra, freq.): ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
 Interação 1: ("h" "u" "g"), ("p" "u" "g"), ("p" "u" "n"), ("b" "u" "n"), ("h" "u" "g" "s")
 Interação 2: ("h" "ug"), ("p" "ug"), ("p" "u" "n"), ("b" "u" "n"), ("h" "ug" "s")
 Interação 3: ("hug"), ("p" "ug"), ("p" "un"), ("b" "un"), ("hug" "s")
 Vocabulário final: {"hug", "p", "ug", "un", "b", "s"}

3.1.2 Wordpiece

Wordpiece é uma técnica de quebrar as palavras em sub-palavras, igual BPE, a principal diferença é que para o critério de concatenação das palavras, é utilizada a máxima-verossimilhança do que a maior a frequência. Esta máxima-verossimilhança é a probabilidade de que o par seja concatenada se a soma deste par é maior que qualquer outro par. Este principio avalia se vale a pena em juntar os símbolos para construir o vocabulário. Wu et al. [2016]

3.1.3 Unigram

Unigram em contraste com BPE ou Wordpiece, cria uma grande quantidade de combinações de caracteres como símbolos para o vocabulário, e a medida que passa as interações, vai reduzindo este vocabulário com base no quanto a função de perda aumenta/diminui ao retirar dado simbolo. Esta função de perda baseada na log-verossimilhança. Este processo acaba quando queremos uma quantidade de símbolos para o vocabulário, e este algoritmo guarda os caracteres básicos para que qualquer palavra possa ser tokenizada. Kudo [2018]

3.1.4 SentencePiece

SentencePiece utiliza a ideia de que nem todas as línguas tem como tokenizar as palavras separadas por espaço, assim é tratado a sentença inteira para construir o vocabulário incluindo os espaços, no qual pode ser utiliza BPE, Wordpiece ou Unigram para construir o vocabulário. O SentencePiece é um algoritmo que otimiza a quantidade tokens em um vocabulário pré-definido, ao invés de construir um novo vocabulário como é feito por Unigram, WordPiece ou BPE. DBL

4 Contextualizando vocabulário

Ao definir o vocabulário do dicionário, utilizando a tokenização depois de ter feito os critérios de pré-processamento do texto, é importante que palavras similares tenham representações numericamente próximas, para que um algoritmo consiga manipular e criar relações para a tarefa. Estas representações contextuais das palavras são aprendidas para codificar em algum sentido quantitativo, a caracterizar palavras similares com operações lógicas com as palavras.

4.1 One-Hot-Enconde

A contextualização dos tokens de um vocabulário em formato One-Hot-Enconde transforma os índices da ordem do dicionário em um vetor do tamanho do vocabulário. Estes índice é representado como valor 1 na respectiva casa do vetor, e o resto igual a zero.

Vocabulário: {"Quando": 1, "olhei": 2, "Você":3}
 One-Hot-Enconde: {"Quando": (1,0,0), "olhei": (0,1,0), "Você":(0,0,1)}

4.2 Bag of Words (BOW)

Em Bag of Words (BOW), o texto é representado como uma "bolsa de palavras", no qual é verificado a ocorrência ou a frequência da ocorrência, de cada palavra do vocabulário com base no texto passado como exemplo. Este modelo desconsidera a gramática e a ordem das palavras, assim apenas é criando um vetor com o tamanho e a ordem do vocabulário, representado se alguma destas palavras ocorre ou quantas vezes ocorre, para cada exemplo.

Vocabulário: {"João", "Clara", "gosta", "de", "assistir", "filmes", "também"}
 Texto: "João gosta de assistir filmes. Clara também gosta de filmes."
 BOW: {"João": 1, "Clara": 1, "gosta": 2, "de": 2, "assistir": 1, "filmes": 2, "também": 1}
 vetor-BOW: (1, 1, 2, 2, 1, 2, 1)

Uma variação deste modelo é utilizar n-grams, no qual junta n-palavras para serem contadas (Ex.: {"João gosta": 1})

4.3 TF-IDF

O modelo Term Frequency-Inverse Document Frequency (TF-IDF) é uma técnica que quantifica a probabilidade das palavras de um vocabulário estar presente entre vários documentos de texto, ponderando se a palavra é frequente ou rara entre estes. Sammut and Webb [2010]

Vocabulário: {"feliz", "gosta", "ama", "filmes"}
 Documento 1 (D1): "João gosta de assistir filmes, ama filmes de comédia."
 Documento 2 (D2): "Clara detesta filmes de terror."
 BOW (D1): {"feliz": 0, "gosta": 1, "ama": 1, filmes: 2}
 BOW (D2): {"feliz": 0, "gosta": 0, "ama": 0, filmes: 1}
 tfidf (D1): {"feliz": 0, "gosta": (1/4)log(2/1), "ama": (1/4)log(2/1), filmes: (2/4)log(2/2)}
 tfidf (D2): {"feliz": 0, "gosta": 0, "ama": 0, filmes: (2/2)log(2/2)}
 vetor-tfidf (D1): (0, (1/4)log(2/1), (1/4)log(2/1), 0)
 vetor-tfidf (D2): (0, 0, 0, 0)

Este modelo cria um vetor de quantificação para cada palavra do vocabulário, codificando cada documento como um vetor do vocabulário que representa a estatística dada por TF-IDF. O calculo deste é feito pela função $tfidf(.)$, dado por:

- $tf(.)$ representa a frequência de uma palavra, entre as que constitui o vocabulário, no respectivo documento analisá-lo (Ex.: $tf("gosta", D1) = 1/4$).
- $idf(.)$ representa em quantos documentos (D documentos), aparece a palavra certa palavra do vocabulário. Este calculo é feito como: $idf("ama", D) = \log(D / \sum_D I("ama" \in D_j)) = \log(2/1)$.²
- $tfidf(.)$ é a estatística que representa se dada palavra do vocabulário é frequente entre os artigos ou não. O calculo é feito multiplicando os resultados dos termos acima. $tfidf(.) = tf(.) \times idf(.)$

Este algoritmo é bastante utilizado para classificação, ou como fator de ponderação para buscas de recuperação de informação, mineração de texto e modelagem de usuários. Se o vocabulário é raro obter algumas palavras nos documentos, é utilizado um valor $\epsilon > 0$ para que não ocorra divisão por zero.

²A função $I(x \in C)$ é uma função indicadora que representa $I(x \in C) = 1 \iff x \in C$ caso contrário $I(x \in C) = 0$.

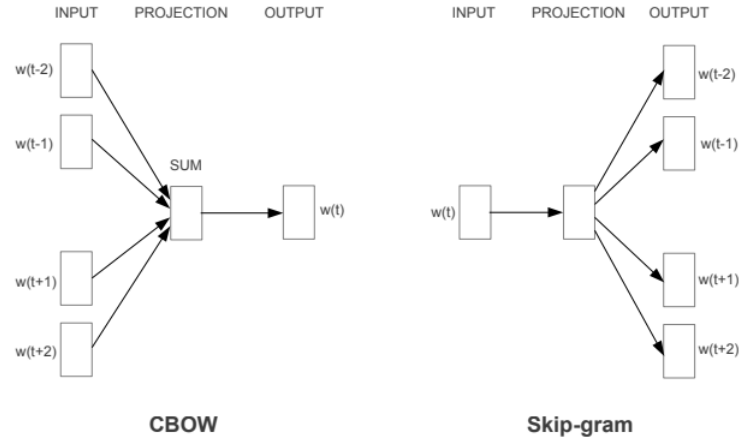


Figure 1: Modelos do Word2Vec.

4.4 Word Embedding

Word Embedding é uma transformação que representa palavras com representações ou sentidos similares, em vetores com o mesmo sentido espacial. Estas representações vetoriais são uteis para criar um espaço de representação que codifica sentidos como gênero ou profissão, assim dando liberdade para que algum modelo possa mapear e manipulá-la com operações matemáticas que tenha um sentido gramatical e/ou semântico.

4.4.1 Word2Vec

Word2vec é um modelo baseado rede neural que aprende a fazer associações de palavras com base em um grande corpus de texto, assim produz um espaço vetorial no qual cada palavra é atribuída a um vetor correspondente no espaço. Estes vetores são posicionados no espaço vetorial de modo que as palavras que compartilham contextos comuns no corpus estejam localizadas próximas umas das outras no espaço. Mikolov et al. [2013]

Este modelo consegue capturar o sentido de tarefas com questões de analogias da forma: "a" está para "b" como "a" é para "c" ?. Este modelo representa as palavras em formato de vetores no qual cálculos podem ser feitos para abstrair estas analogias como por exemplo: " 'homem' está para 'mulher' assim como 'tio' está para 'tia' ?". Estas operações usa do deslocamento vetorial, baseado na distância do cosseno, para verificar o grau de similaridade da analogia. Esse tipo de composição vetorial também nos permite responder a operações como "Rei — Homem + Mulher = ?" e chegar a um resultado como "Rainha".

Para criar estas representações, o Word2vec pode ser treinado baseado em dois modelos, Continuous Bag-of-Words Model (CBOW) ou Continuous Skip-gram Model (Skip-gram), Figura 1:

- No modelo CBOW, a rede neural é treinada utilizando frases, no qual a palavra no meio da frase é apagada e utiliza das palavras antes e depois desta, para predizer qual é a palavra que mais se encaixa no contexto.
- No modelo Skip-gram, é passado uma palavra para o modelo, é predito quais palavras antes e depois a esta.

Estas palavras são codificadas como one-hot-enconde baseado no tamanho do vocabulário, e a saída destes modelos são probabilidades entre as palavra do vocabulário que mais se assemelha a ser a correta. Este treinamento guia uma representação para o espaço latente que é utilizado para codificar uma palavra do vocabulário em formato de vetor contextualizado.

4.4.2 GloVe

O algoritmo Global Vectors for Word Representation, ou GloVe, é uma extensão do método word2vec para o aprendizado eficiente de vetores de palavras. O GloVe constrói uma matriz de contexto de palavras ou de coocorrência de palavras explícitas usando estatísticas em todo o corpus de texto. O resultado é um modelo de aprendizado que pode resultar em embeddings de palavras geralmente melhores. Pennington et al. [2014]

4.5 Deep-Learning based Embedding

Algoritmos de word Embedding baseados em aprendizado profundo como ELMo ou BERT, codificam o texto de entrada em representações latentes que podem ser utilizadas como word Embedding Devlin et al. [2019], Peters et al. [2018]. A principal diferença destes modelos, é a capacidade de contextualizar os tokens de palavras com as respectivas ordens no texto passado, assim codificando a nível de frases, representações intermediárias e únicas para uma representação vetorial da palavra pelo contexto da frase.

Estes modelos, diferentemente do Glove ou Word2Vec, cria uma representação vetorial para cada token com base no texto passado como exemplo, em vez de codificar palavra por palavra. Os vetores resultantes, em particular em modelos baseados em transformers, é possível utilizar representações latentes de cada cabeça de atenção, ou cada saída de camada para representar os tokens de forma contextualizada pelo texto, no qual cada uma das optativas codificam de forma única, alguma relação da palavra com o texto de entrada.

5 Discussão

O pré-processamento do texto é necessário para que um algoritmo baseado em linguagem natural tenha melhor eficiência, no qual este padroniza os textos de entrada sem que perca a contextualização para uma tarefa e possa ser tokenizada para passar para algum modelo. Em princípio, é criado um vocabulário com base na tarefa desejada e as palavras tokenizadas consigam codificar o texto sem perder o contexto.

Para que seja captado a similaridade entre palavras tokenizadas, o Word Embedding é um metodo comumente utilizado que transforma estes tokens em vetores associativos. Estes vetores conseguem captar palavras que possuem o mesmo sentido contextual, como homem e mulher em relação a sexo, e manipulações vetoriais como soma, subtração e similaridade que possa captar sentidos lógicos para operações com palavras em formado de vetores.

As representações vetoriais contextualizadas para cada palavra são úteis para modelos mais avançados. Em geral, modelos baseados em embeddings conseguem ter uma pré-representação de textos, no qual é mais fácil aprender dada tarefa sem ter que aprender "ler" desde o início. Estes modelos podem ser utilizados para de classificação de texto, síntese de texto, Question answering (QA), recomendação, entre outros.

References

- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL <http://arxiv.org/abs/1508.07909>.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL <http://arxiv.org/abs/1609.08144>.
- Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *CoRR*, abs/1804.10959, 2018. URL <http://arxiv.org/abs/1804.10959>.
- Sentencepiece: A tokenizer de subpalavra simples e independente de idioma e detokenizador para processamento de texto neural. abs/1808.06226. URL <http://arxiv.org/abs/1808.06226>.
- Claude Sammut and Geoffrey I. Webb, editors. *TF-IDF*, pages 986–987. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi:10.1007/978-0-387-30164-8_832. URL https://doi.org/10.1007/978-0-387-30164-8_832.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL <http://arxiv.org/abs/1802.05365>.