Documentação Técnica do *Text Similarity*

Data Science Team - Produto

Versão	Data	Autores	Último *status*
1.0	28/10/2022	João Souza	Criação do documento

Repositório do projeto

O repositório deste MVP pode ser acessado clicando no ícone ao lado: 

Ideia geral

Antes de apresentar os fluxogramas principais do MVP, é interessante introduzir a ideia macro deste MVP: comparar, **semanticamente**, dois textos. Existem, na literatura, diversas formas de comparação **estrutural** de textos, isto é, ao compará-los olhamos cada estrutura entre eles de forma *ipsis litteris*. Em algumas aplicações, essa pode não ser uma boa estratégia, uma vez que diferentes palavras podem ter o mesmo significado. Por exemplo, pense nas seguintes frases:

1. "*Ele tem **diabetes** e foi lá tomar insulina*";
2. "*Tomou insulina, pois é **diabético***";
3. "*Depois de desenvolver **TD2M,** começou a tomar insulina*".

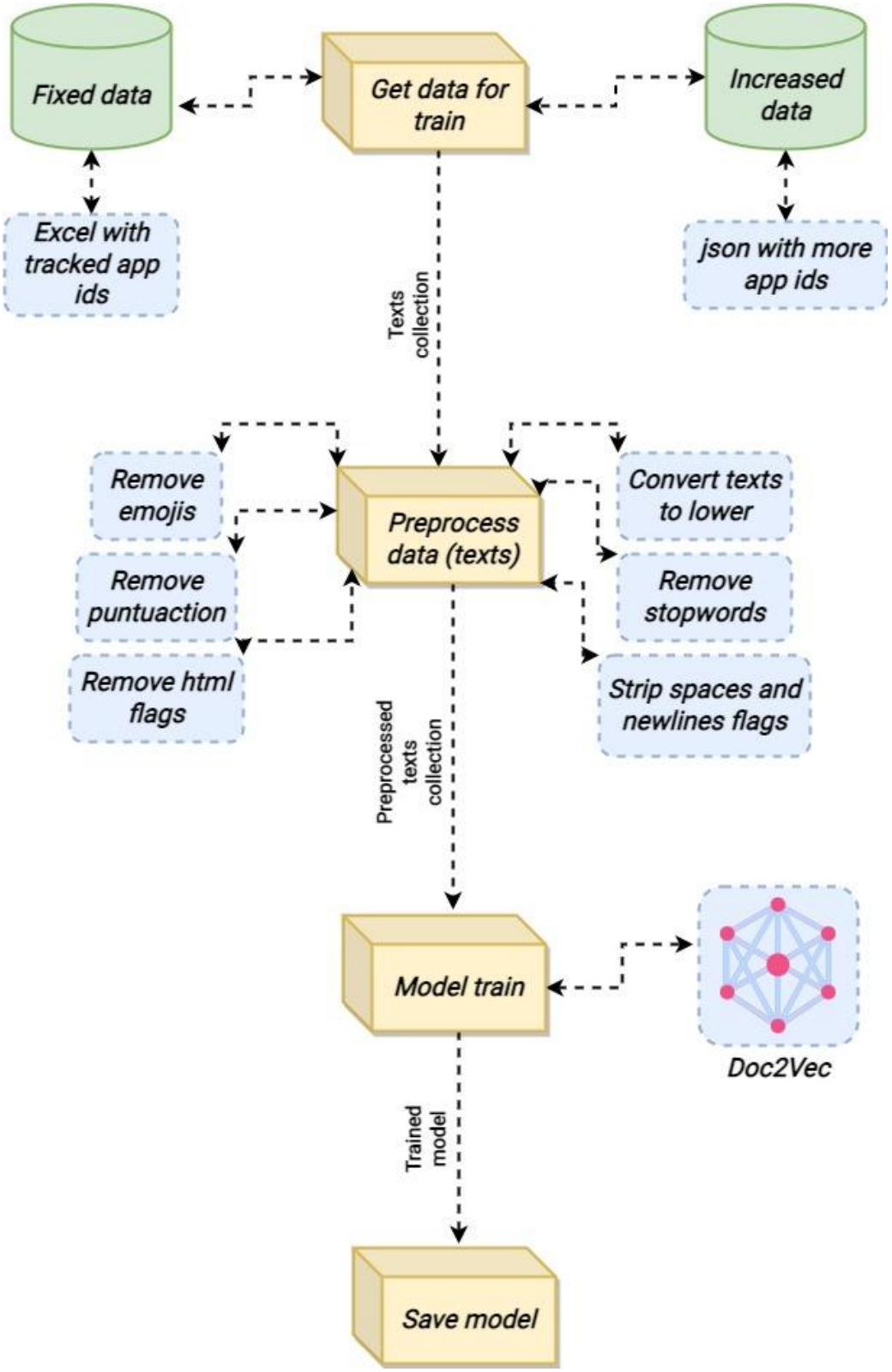
Semanticamente, possuem o mesmo sentido. Sintaticamente, não são iguais. Dessa forma, um comparador estrutural não veria a similaridade que o comparador semântico enxergaria.

Fluxogramas

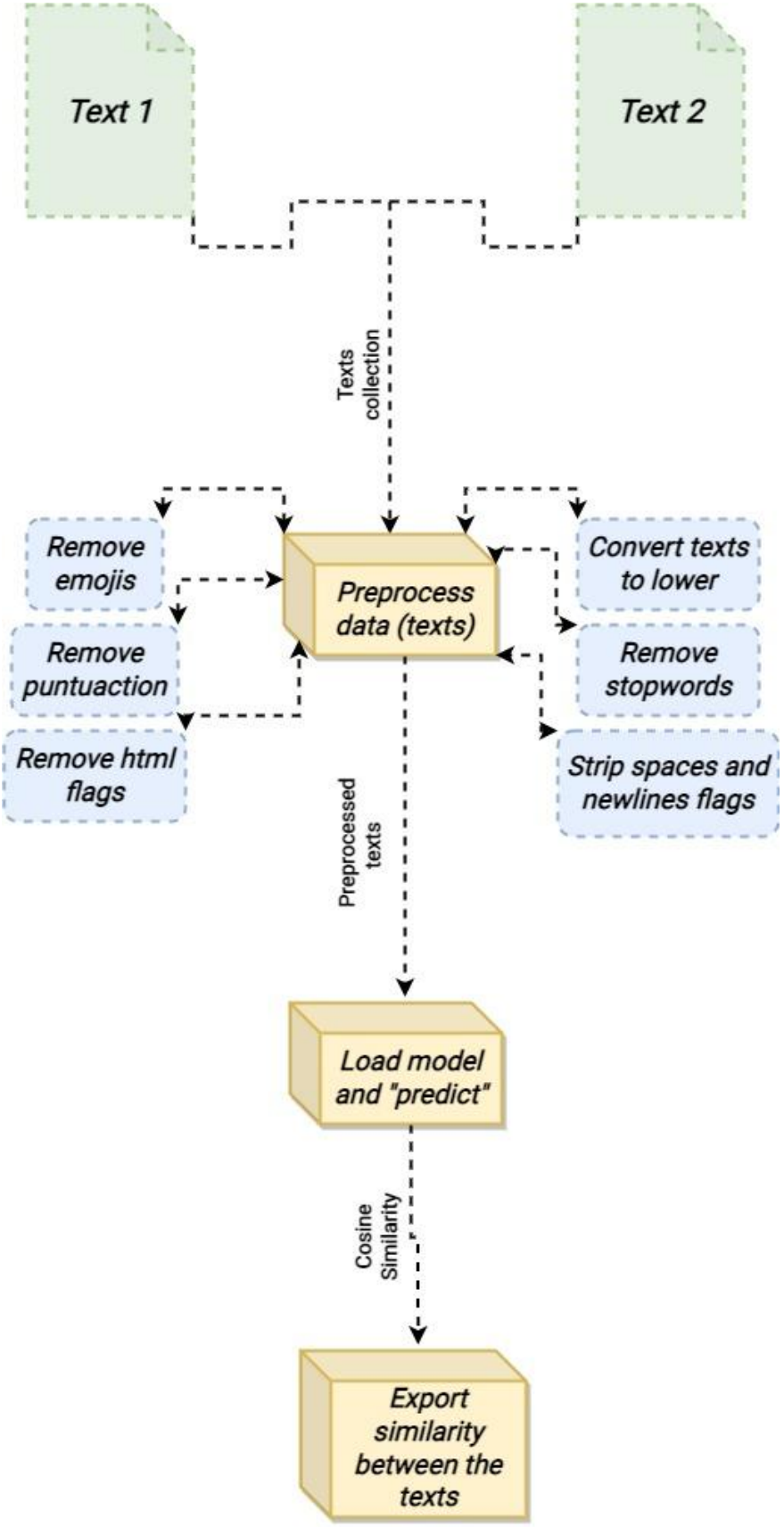
A seguir são apresentados os fluxogramas de execução de partes importantes do MVP, na premissa de ajudar a entender como o projeto está arquitetado.

Fluxograma de Treino

A estratégia de verificação de similaridade é baseada em um algoritmo de *Machine Learning* não supervisionado chamado *Doc2Vec*. Para realizar o treinamento do modelo, o seguinte fluxo foi adotado:


Fixed data
Get data for train
Increased data
Excel with tracked app ids
json with more app ids
Texts collection
Remove emojis
Remove puntuaction
Remove html flags
Preprocess data (texts)
Convert texts to lower
Remove stopwords
Strip spaces and newlines flags
Preprocessed texts collection
Model train
Doc2Vec
Trained model
Save model

Fluxograma de Predição



Detalhes acerca do cálculo da similaridade

Como mostrado no fluxograma anterior, o primeiro passo para calcular a similaridade entre os textos é tratá-los de forma a deixar apenas palavras que trazem de fato informações úteis. A seguir mostraremos o formato de entrada e o que sai desta etapa:

Estrutura de *input*

Texto 1: *"Então, quero que vocês entendam, que o melhor que a gente pode ter na vida, são as coisas básicas: é a nossa saúde, é a família, é um amigo, é um lugar pra viver, ta ligado?!... É ter no que acreditar, é viver em função de um sonho... Eu tenho uma alma, que é feita de sonhos !"*

Texto 2: *"Tão natural quanto a luz do dia Mas que preguiça boa, me deixa aqui à toa Hoje ninguém vai estragar meu dia Só vou gastar energia pra beijar sua boca Fica comigo então, não me abandona, não Alguém te perguntou como é que foi seu dia? Uma palavra amiga, uma notícia boa Isso faz falta no dia a dia A gente nunca sabe quem são essas pessoas Eu só queria te lembrar Que aquele tempo eu não podia fazer mais por nós Eu estava errado e você não tem que me perdoar Mas também quero te mostrar Que existe um lado bom nessa história Tudo que ainda temos a compartilhar E viver, e cantar Não importa qual seja o dia Vamos viver, vadiar O que importa é nossa alegria"*

Array 1:

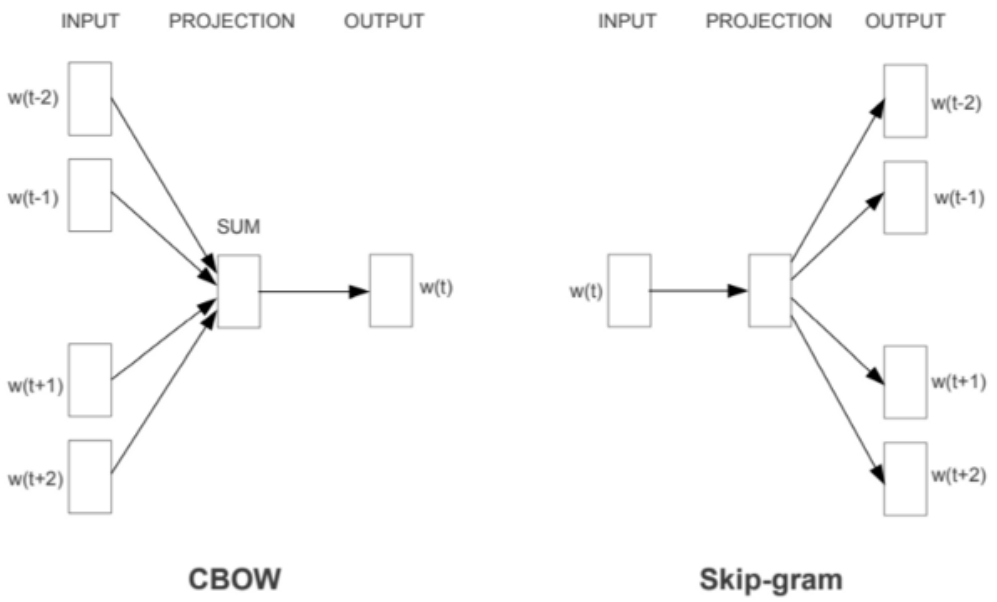
['então','quero','entendam','melhor','gente','pode','ter','vida','coisas','básicas','saúde','família','amigo','lugar','pra','viver','ta','ligado','ter','acreditar','viver','função','sonho','alma','feita','sonhos']

Array 2:

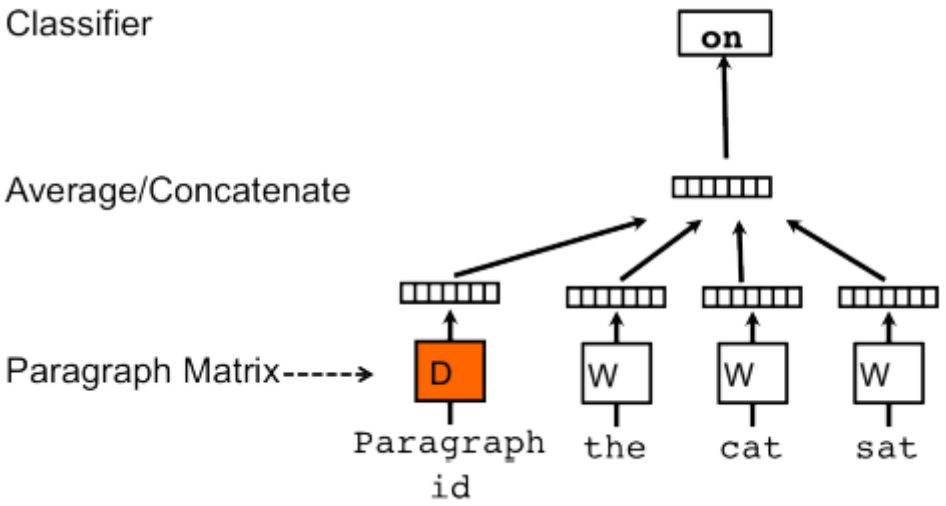
['tão','natural','quanto','luz','dia','preguiça','boa','deixa','aqui','toa','hoje','ninguém','vai','estragar','dia','vou','gastar','energia','pra','beijar','boca','fica','comigo','então','abandona','alguém','perguntou','dia','palavra','amiga','notícia','boa','faz','falta','dia','dia','gente','nunca','sabe','pessoas','queria','lembrar','tempo','podia','fazer','errado','perdoar','quero','mostrar','existe','lado','bom','nessa','história','tudo','ainda','compartilhar','viver','cantar','importa','dia','vamos','viver','vadiar','importa','alegria']

Doc2Vec

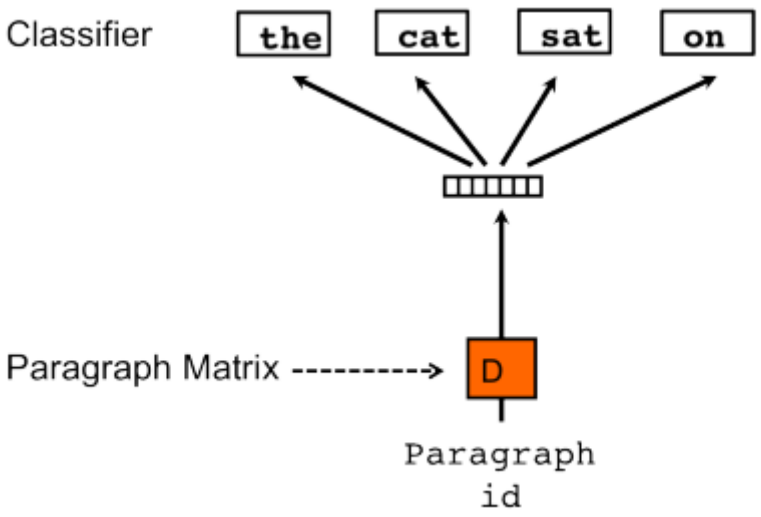
Após os *outputs* dos textos processados serem gerados, carregamos então o modelo *Doc2Vec* já treinando com uma base de descrições. O *Doc2Vec* opera através da **combinação** de duas estratégias: PV-DM (*Paragraph Vector - Distributed Memory)* e PV-DBOW (*Paragraph Vector - Distributed Bag of Words)*.



Fonte: Medium - *Doc2Vec: Computing Similarity between Documents.*



Fonte: Medium - *Doc2Vec: Computing Similarity between Documents.*



Fonte: Medium - *Doc2Vec: Computing Similarity between Documents.*

A abordagem da Rede Neural do tipo **PV-DBOW** funciona respondendo à seguinte pergunta: "*dado um contexto de N palavras vizinhas a uma palavra W, qual será essa possível palavra W?*"

Já a abordagem **PV-DM,** funciona de maneira inversa: ela busca entender qual o contexto que uma palavra está inserida: *"dada uma palavra W, quais são, em geral, as N palavras vizinhas dessa palavra W?"*

Com a combinação dessas duas abordagens, saímos apenas do aspecto estrutural do texto e conseguimos captar os aspectos contextuais de cada palavra ao longo dos textos.

Ao final da combinação, é gerado um *array* unidimensional que é a representação numérica dos textos através da utilização do *Doc2Vec* para cada texto de entrada.

Medindo a similaridade

Para medir a similaridade entre os dois textos, utilizamos atualmente a métrica da Similaridade de Cossenos. Já que possuímos dois vetores numéricos, podemos calcular a similaridade como:

$$\textbf{similarity} = \cos\theta = \frac{A \bullet B}{||A||\,||B||}$$

onde A e B são os vetores numéricos gerados para os textos pelo *Doc2Vec*.
Logicamente, quando a similaridade é máxima é quando o **cos θ = 1,** ou seja, **θ=0°,** dizendo que os dois vetores são idênticos. Dessa forma, à medida que o **θ** aumenta

em sentido anti-horário, a distância entre os dois vetores aumenta, diminuindo a similaridade entre os textos. Exemplo:

A = [1, 2, 3]

B = [1, 2, 3]

A·B = (1*1) + (2*2) + (3*3) = 14

||A|| = $\sqrt{1^2 + 2^2 + 3^2} = \sqrt{1 + 4 + 9)} = \sqrt{14}$

||B|| = $\sqrt{1^2 + 2^2 + 3^2} = \sqrt{1 + 4 + 9)} = \sqrt{14}$

$$\textbf{similarity} = \cos\theta = \frac{14}{\sqrt{14}\sqrt{14}}$$

$$\textbf{similarity} = \cos\theta = \frac{14}{14}$$

$$\textbf{similarity} = \cos\theta = 1$$

Alguns resultados

Para testar a aplicação, foram calculadas as similaridades entre 8635 combinações de aplicativos da Google Play de diversas categorias.

Similaridades acima de 60%:

	Text 1	Text 2	Cosine Similarity
7060	OurTime Namoro e Encontros.txt	Par Perfeito EncontrosNamoro.txt	0.889347
2498	Buscapé Menor preço Cashback.txt	Zoom Compare preços Cashback.txt	0.741046
839	Amazon Prime Video.txt	Pluto TV – TV Ao vivo e Filmes.txt	0.693342
3076	Centauro loja de esportes.txt	Netshoes Loja de Esportes.txt	0.675561
3932	Disney.txt	HBO Max Assista à TV e filmes.txt	0.674829
6690	Namoro Encontros e Batepapo .txt	POF Brasil Site de Namoro.txt	0.659059
723	Amazon Music Ouvir músicas.txt	Resso Músicas e Podcasts.txt	0.644711
2699	Cabify.txt	Unayo.txt	0.634925
5806	Kwai Ver Vídeos Bacanas.txt	TikTok.txt	0.634246
6328	Meu Moto Táxi.txt	Pop Táxi.txt	0.629933
5430	Instagram.txt	Snapchat.txt	0.626480
3240	Crunchyroll.txt	Disney.txt	0.624560
6391	Moto Táxi Oficial.txt	Pop Táxi.txt	0.623762
49	99 Food Pedir Comida.txt	iFood comida e mercado em casa.txt	0.623418
7468	Plex Streaming de filmes e TV.txt	ViX Filmes e TV sem limites.txt	0.619008
3681	Delivery Much Pedir Comida.txt	pedeai delivery de comida.txt	0.617000
5131	HBO Max Assista à TV e filmes.txt	ViX Filmes e TV sem limites.txt	0.616912
1243	ARTE TV – Streaming et Replay.txt	ZKB Mobile Banking.txt	0.615964
7790	Santander Brasil.txt	Santander.txt	0.612809
3596	Deezer Ouvir Música e Podcast.txt	Resso Músicas e Podcasts.txt	0.611360
660	Amazon Music Ouvir músicas.txt	Deezer Ouvir Música e Podcast.txt	0.609513
3174	Coroa Metade.txt	Namoro e Encontros Mequeres.txt	0.609137
7762	Resso Músicas e Podcasts.txt	SoundCloud música e áudio.txt	0.607723
5088	HBO Max Assista à TV e filmes.txt	Netflix.txt	0.606167

Boxplot dos dados de similaridade