Desafio 1:

Lorem Ipsum

“Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...“ (Latin)
“Não há ninguém que ame a dor por si só, que a busque e queira tê-la, simplesmente por ser dor...“ (Portuguese)
“There is no one who loves pain for itself, who seeks it and wants to have it, simply because it is pain...“ (English)

Fusce risus ex, posuere at ante at, condimentum vestibulum nunc. Proin dapibus egestas neque, a tempor odio pharetra eget. Nullam tempus felis eu consectetur tincidunt. Integer fermentum eu quam vitae tempus. Vivamus volutpat ut dui vitae sollicitudin. Donec ornare dolor a vestibulum pharetra. Mauris eget dui sapien. Mauris lobortis feugiat neque, nec congue leo viverra semper. In vel mauris nunc. Aenean scelerisque arcu a varius tempor. Proin quis nisi et mi dictum tempor. Pellentesque mattis risus metus, id luctus orci ultrices ac. Praesent bibendum lectus a nisl mollis, eu suscipit eros pretium.

Phasellus vitae elit efficitur, varius felis id, vulputate neque. Praesent dictum nunc in velit lobortis tempus. Duis pulvinar ut urna eget volutpat. In hac habitasse platea dictumst. Aenean eros libero, ultrices eu dolor id, ultrices bibendum mauris. Nunc quis nunc feugiat, dapibus sem eget, pretium orci. Mauris mauris felis, pulvinar nec dapibus in, laoreet ac nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In sed scelerisque massa. Morbi sit amet ex erat. Duis sit amet laoreet ipsum. Nullam rutrum dapibus metus id sollicitudin. Sed lorem metus, maximus id maximus ac, vehicula vitae neque.

Curabitur vestibulum lorem diam, nec dictum lectus sodales id. Morbi at dui at ex fermentum porttitor sit amet et ipsum. Nullam et nibh vitae arcu porta pretium. Aenean ultricies bibendum nibh vitae mattis. Donec nec dignissim lorem. Vivamus ut lectus nec sem suscipit gravida. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Proin nec metus at sapien tempor ultrices. Donec sed orci ante. Quisque finibus maximus justo, sit amet dignissim justo ornare ut. Morbi imperdiet sagittis tristique. Nunc vestibulum velit eget neque ultrices, sit amet egestas elit eleifend.

Curabitur facilisis elementum ligula vitae pellentesque. Nulla facilisi. Nulla vel tristique arcu. Sed dictum nisl ac laoreet venenatis. Proin eros est, suscipit id sodales eu, semper maximus eros. Sed maximus id nulla vel auctor. Nullam libero odio, auctor eget felis et, viverra gravida nisl. Vivamus sed luctus eros, nec mattis leo. Phasellus eget accumsan justo. Proin nisl sem, luctus ut gravida consequat, congue sit amet arcu.

4 paragraphs, 350 words

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/68ba1f5f-c2f7-4c98-9080-b1ab4117427a/Base_Text_-_Matrix_Representation.txt

> Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics.

> The process of converting words into numbers are called Vectorization.

▼ **NLP** → Natural Language Processing

→ Limpeza de texto e pré-processamento de normalização. Depois disso, converteremos o texto processado em vetores de recursos numéricos para que possamos alimentá-lo em computadores para aplicativos de aprendizado de máquina.

▼ **Familiar with Terminologies:**

Document

A document is a single text data point. **For Example,** a review of a particular product by the user.

Corpus

It a collection of all the documents present in our dataset.

Feature

Every unique word in the corpus is considered as a feature.

▼ **Exemplo**:

For Example, Let's consider the 2 documents shown below:

```
Sentences:
Dog hates a cat. It loves to go out and play.
Cat loves to play with a ball.
```

We can build a corpus from the above 2 documents just by combining them.

```
Corpus = "Dog hates a cat. It loves to go out and play. Cat loves to play with a ball."
```

And features will be all unique words:

```
Fetaures: ['and', 'ball', 'cat', 'dog', 'go', 'hates', 'it', 'loves', 'out', 'play', 'to', 'with
```

▼ **What is Word Embedding**?

→ Em termos muito simples, Word Embeddings são os textos convertidos em números e podem haver diferentes representações numéricas do mesmo texto.

▼ **Por que precisamos disso?**

→ Como sabemos, muitos algoritmos de aprendizado de máquina e quase todas as arquiteturas de aprendizado profundo não são capazes de processar strings ou texto simples em sua forma bruta. Em um sentido amplo, eles requerem números numéricos como entradas para realizar qualquer tipo de tarefa, como classificação, regressão, agrupamento, etc.

→ Além disso, da enorme quantidade de dados que está presente no formato de texto, é imperativo extrair alguns conhecimento a partir dele e construir quaisquer aplicações úteis. Resumindo, podemos dizer que para construir qualquer modelo em aprendizado de máquina ou aprendizado profundo, os dados do nível final devem estar na forma numérica porque os modelos não entendem dados de texto ou imagem diretamente como os humanos.

▼ **Como os modelos de PNL aprenderam padrões a partir de dados de texto**?

→ Para converter os dados de texto em dados numéricos, precisamos de algumas formas inteligentes que são conhecidas como vetorização ou, no mundo da PNL, são conhecidas como Word embeddings.

→ Desse modo, dizemos que extraímos recursos com a ajuda de texto com o objetivo de construir múltiplas linguagens naturais, modelos de processamento, etc.

▼ **Count Vectorizer:**

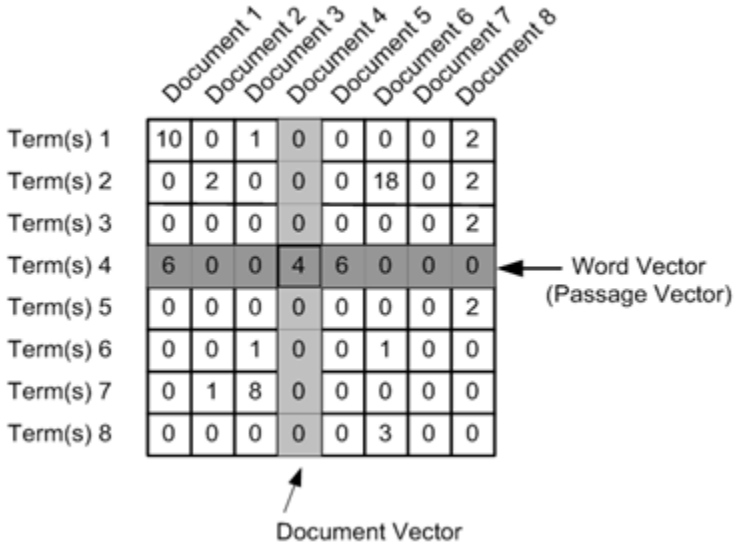
1. É uma das maneiras mais simples de fazer vetorização de texto.
2. Ele cria uma matriz de termos de documento, que é um conjunto de variáveis fictícias que indica se uma determinada palavra aparece no documento.
3. O vetorizador de contagem ajustará e aprenderá o vocabulário de palavras e tentará criar uma matriz de termos de documento em que as células individuais

denotem a frequência dessa palavra em um documento específico, que também é conhecido como termo frequência, e as colunas são dedicadas a cada palavra do corpus.

▼ **A forma como a contagem é identificada para cada palavra:**

→ Temos as duas opções a seguir em mãos para a entrada na matriz de contagem:

- Use a frequência (número de vezes que uma palavra apareceu no documento); ou
- Presença (a palavra apareceu no documento?) Para ser a entrada na matriz de contagem M. Mas, em geral, preferimos o método de frequência ao último.



▼ **Bag-of-Words (BoW):**

→ Essa técnica de vetorização converte o conteúdo do texto em vetores de recursos numéricos. Bag of Words pega um documento de um corpus e o converte em um vetor numérico mapeando cada palavra do documento em um vetor de recurso para o modelo de aprendizado de máquina.

Disadvantages of Bag of Words

1. This method doesn’t preserve the word order.

2. It does not allow to draw of useful inferences for downstream NLP tasks.

▼ **Tokenization**:

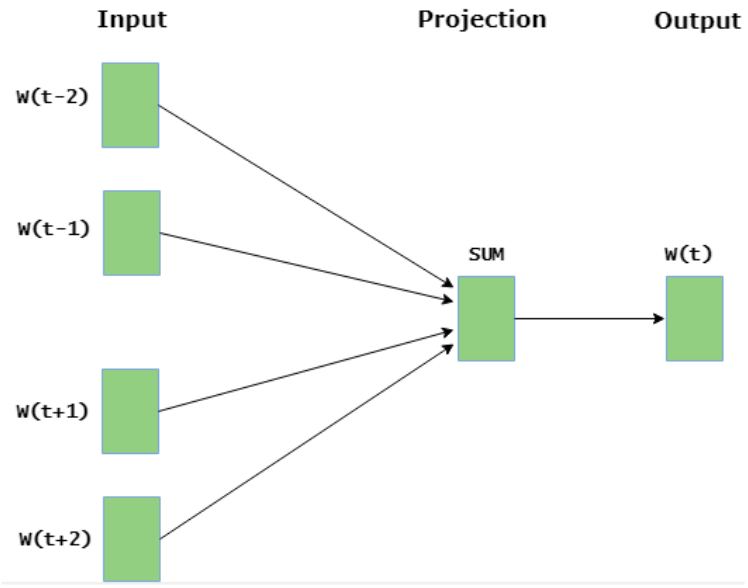
→ É o processo de dividir cada frase em palavras ou partes menores, conhecidas como tokens. Após a conclusão da tokenização, iremos extrair todas as palavras exclusivas do corpus. Aqui corpus representa os tokens que obtemos de todos os documentos e usados para a criação do pacote de palavras.

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = ["This burger is very tasty and affordable.," This burger is not tasty and is affordable.","This burger is
countvectorizer = CountVectorizer()
X = countvectorizer.fit_transform(corpus)
result = X.toarray()
print(result)
```

```
[[1 1 1 0 1 0 1 1 1]
 [1 1 1 0 2 1 1 1 0]
 [0 0 1 1 1 0 0 1 2]]
```

▼ **Continuous Bag-of-Words Model (CBOW):**

We denote this model as CBOW. The CBOW architecture is similar to the feedforward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all the words; thus all words get projected into the same position.



▼ **TF-IDF Vectorization:**

→ Como discutimos nas técnicas acima, o método BOW é simples e funciona bem, mas o problema com isso é que ele trata todas as palavras igualmente. Como resultado, ele não consegue distinguir palavras muito comuns ou raras.

→ Term frequency-inverse document frequency (TF-IDF) fornece uma medida que leva em consideração a importância de uma palavra dependendo da frequência com que ela ocorre em um documento e em um corpus.

- Term frequency (TF)
- Inverse document frequency (IDF)

▼ **Term-Frequency:**

Term Frequency

Term frequency denotes the frequency of a word in a document. For a specified word, it is defined as the ratio of the number of times a word appears in a document to the total number of words in the document.

Or, it is also defined in the following manner:

It is the percentage of the number of times a word (x) occurs in a particular document (y) divided by the total number of words in that document.

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

The formula for finding Term Frequency is given as:

```
tf ('word') = Frequency of a 'word' appears in document d / total number of words in the documen
```

▼ **Inverse Document Frequency:**

Inverse Document Frequency

It measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus.

It is the logarithmic ratio of no. of total documents to no. of a document with a particular word.

$$IDF(\boldsymbol{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \boldsymbol{term}\ in\ it}\right)$$

This term of the equation helps in pulling out the rare words since the value of the product is maximum when both the terms are maximum. What does that mean? If a word appears multiple times across many documents, then the denominator df will increase, reducing the value of the second term. (Refer to the formula given below)

Thus common words would have lesser importance.

$$TFIDF(\boldsymbol{term}) = TF(\boldsymbol{term}) * \text{IDF}(\mathbf{term})$$

The formula for finding TF-IDF is given as:

$$W_{x,y} = tf_{x,y} * log\left(\frac{N}{df_x}\right)$$

where,

$W_{x,y}$= Word x within document y

$tf_{x,y}$= frequency of x in y

df_x= number of documents containing x

N=total number of documents

▼ **Cosine Similarity:**

→ Antes de entender o word2vec, vamos primeiro entender o que é similaridade de cosseno porque word2vec usa similaridade de cosseno para descobrir a palavra mais semelhante (que será discutida na próxima seção).

→ A similaridade de cossenos não só indica a similaridade entre dois vetores, mas também testa a ortogonalidade do vetor:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

→ Onde theta é o ângulo entre dois vetores, se o ângulo está próximo de zero podemos dizer que os vetores são muito semelhantes entre si e se theta é 90 então podemos dizer que os vetores são ortogonais entre si e se teta é 180, podemos dizer que ambos os vetores são opostos um ao outro.

▼ **Google Word2Vec:**

It is deep learning technique with two-layer neural network.Google Word2vec take input from large data (in this scenario we are using google data) and convert into vector space. Google word2vec is basically pretrained on google dataset. To download the dataset you can follow this link — Google Dataset. So before diving into Google word2vec lets first understand what is word2vec.

Word2vec basically place the word in the feature space is such a way that their location is determined by their meaning i.e. words having similar meaning are clustered together and the distance between two words also have same meaning. Consider an example given below:

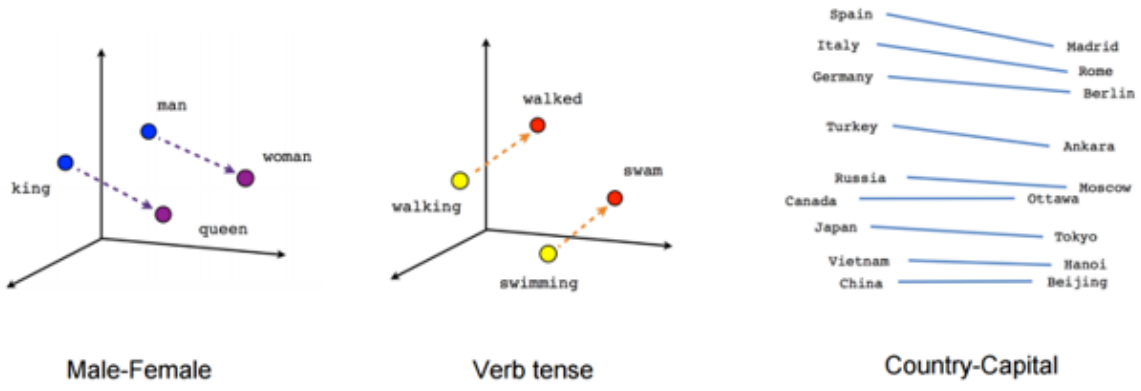


Figure 6 – Mapping of words

▼ **Referência Bibliográfica:**

- https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223
- https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/
- https://www.analyticssteps.com/blogs/word-embedding-nlp-python-code
- https://towardsdatascience.com/understanding-feature-engineering-part-3-traditional-methods-for-text-data-f6f7d70acd41
- https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7
- https://towardsdatascience.com/how-to-turn-text-into-features-478b57632e99