# Tokenização

Processo de dividir uma string em listas de pedaços ou "tokens". Um token é uma parte inteira. Por exemplos: uma palavra é um token em uma sentença. Uma sentença é um token em um parágrafo.

Input -> string

Output -> list

Ex: Divide o parágrafo em uma lista de sentenças.

from nltk.tokenize import sent\_tokenize

sent\_tokenize(paragrafo)

Ex: Divide a frase em uma lista de palavras

from nltk.tokenize import word\_tokenize

word\_tokenize(frase)

# StopWords

Stopwords são palavras comuns que normalmente não contribuem para o significado de uma frase, pelo menos com relação ao propósito da informação e do processamento da linguagem natural. São palavras como "The" e "a" ((em inglês) ou "O/A" e "Um/Uma" ((em português). Muitos mecanismos de busca filtram estas palavras (stopwords), como forma de economizar espaço em seus índices de pesquisa.

Output -> list

Ex: from nltk.corpus import stopwords

portuguese\_stops = set(stopwords.words('portuguese'))

stopwords.fileids() #mostra os alfabetos que tem

# Stemming

Stemming é a técnica de remover sufixos e prefixos de uma palavra, chamada stem. Por exemplo, o stem da palavra cooking é cook. Um bom algoritmo sabe que "ing" é um sufixo e pode ser removido. Stemming é muito usado em mecanismos de buscas para indexação de palavras. Ao invés de armazenar todas as formas de uma palavra, um mecanismo de busca armazena apenas o stem da palavra, reduzindo o tamanho do índice e aumentando a performance do processo de busca.

Input -> str

Output -> str

Esse primeiro é específico para o pt-br.

Ex: stemmer = nltk.stem.RSLPStemmer()

stemmer.stem("apertou")

Esse foi definido para pt-br

Ex: from nltk.stem.snowball import SnowballStemmer

stemmer = SnowballStemmer("portuguese")

stemmer.stem("apertou")

# Lemmatization

Lemmatization leva em consideração a análise morfológica das palavras. Para fazer isso, é necessário ter dicionários detalhados nos quais o algoritmo possa procurar para vincular o formulário ao seu lema. Veja um exemplo:

- Forma flexionada: organizando

- Lema: organiza

- Forma flexionada: organizado

- Lema: organiza

Com Lemmatization as duas formas flexionadas organizando e organizado seria representadas somente pelo lema organiza.

Não consegui achar uma função para fazer a lemmatização para o alfabeto portugues do mesmo jeito que com o inglês, apenas usando o spacy.

Para alfabeto inglês.

Input -> str

Output -> str

Ex: from nltk.stem import WordNetLemmatizer

WordNetLemmatizer().lemmatize('mice')

# POS Tag

O processo de classificar palavras em suas classes gramaticais (adjetivos, substantivos, artigos, etc) e rotulá-las de acordo é conhecido como pos\_tag.

Input -> str

Output -> list

Para a língua inglesa.

Ex: from nltk import pos\_tag

pos\_tag(word\_tokenize(frase))

Para a língua portuguesa, está no script de ngram.

# Collocations/Ngram

Collocations são duas ou mais palavras que tendem a aparecer frequentemente juntas, como "Estados Unidos", "Rio Grande do Sul" ou "Machine Learning". Essas palavras podem gerar diversas combinações e por isso o contexto também é importante no processamento de linguagem natural.

Os dois tipos mais comuns de Collocations são bigramas e trigramas. Bigramas são duas palavras adjacentes, como "tomografia computadorizada", "aprendizado de máquina" ou "mídia social". Trigramas são três palavras adjacentes, como "fora do negócio" ou "Proctor and Gamble".

- Bigramas: (Nome, Nome), (Adjetivo, Nome)

- Trigramas: (Adjetivo/Nome, Qualquer\_Item, Adjetivo/Nome)

Input -> str

Output -> data frame (tabela de freq)

# TD-IDF

TF-IDF significa "Frequência do Termo - Frequência Inversa de Documentos".

Essa é uma técnica para quantificar uma palavra nos documentos; geralmente calculamos um peso para cada palavra, o que significa a importância da palavra no documento e no corpus.

TD: A Frequência do Termo mede a frequência de uma palavra em um documento.

Fórmula: tf(t,d) = contagem de t em d / número de palavras em d

DF: mede a importância do documento em todo o corpus e é muito semelhante ao TF. A única diferença é que TF é contador de frequência para um termo t no documento d, onde DF é a contagem de ocorrências do termo t no conjunto de documentos N, ou seja, DF é o número de documentos em que a palavra está presente.

Fórmula: df(t) = ocorrência de t nos documentos

IDF: é o inverso da frequência do documento que mede a informatividade do termo t. Quando calcularmos o IDF, será muito baixo para as palavras que mais ocorrem. Por isso, em um corpus grande o valor do IDF explode, logo calculamos a log dele

Fórmula1: idf(t) = N / df

Fórmula2: idf(t) = log(N / (df + 1))

TD-IDF: será um valor multiplicativo de TF e IDF

Fórmula: tf-idf(t, d) = tf(t, d) \* log(N / (df + 1))

Input -> str

Output -> float