Introdução ao Processamento da Linguagem Natural

Olá, seja bem-vindo!

A capacidade de comunicar-se é uma das principais características que nos permitem viver em sociedade. E a tecnologia é um aspecto que vem gradativamente evoluindo e facilitando a forma com que nos comunicamos.

Nesse sentindo, está cada vez mais fácil usar aplicativos para troca de mensagens e já existem, inclusive, sistemas que usam machine learning para simular uma conversa como se fossem uma pessoa respondendo. Quanta inovação, não é mesmo?!

Esses sistemas são chamados de *chatbots*. Eles já estão presentes em diversos sites de comércio eletrônico e em serviços de atendimento de várias empresas. Neles o usuário geralmente é apresentado a um *chat* com algumas opções de serviço e o assistente virtual vai direcionando o utilizador para determinado setor de atendimento da empresa, com base em suas respostas em mensagens de texto.

E como esses sistemas fazem isso? Como eles conseguem entender as mensagens do usuário?

Bem, por trás dessa tecnologia existe uma área da computação que é chamada de Processamento da Linguagem Natural, tem o objetivo de fazer com que um sistema consiga entender a linguagem dos humanos, ou seja, entender o que nós escrevemos ou falamos.

O Processamento da Linguagem Natural, também representado pela sigla PLN, ou NLP do inglês Natural Language Processing, utiliza conceitos baseados em linguística e regras gramaticais para a construção de algoritmos, que possam extrair alguma informação ou entendimento.

Dentre as aplicações do PNL pode-se destacar a sumarização ou resumo de textos, que permite, por exemplo, captar apenas as ideias principais que contém o sentido de um texto ou de um livro. Outra aplicação são os aplicativos de tradução que utilizam o reconhecimento de voz do usuário para traduzir uma frase. O PNL pode ser aplicado ainda em recuperação de informação, chatbots, entre outras utilidades.

Caro, cursista, agora você será apresentado, de fato, a alguns conceitos importantes no aprendizado de PLN. São eles: Corpus, Tokenização, e Stop words.

O Corpus é representado por um conjunto de textos escritos em um determinado idioma, que foram manualmente anotados e servirá de validação para as análises que serão realizadas.

Já a Tokenização, refere-se a divisão de um texto em partes menores que representam as palavras, ou também chamadas de tokens, e podem ser separadas por espaços, vírgulas ou pontuações.

Por último, mas não menos importante, os Stop words. Eles são palavras que geralmente são removidas no início do processamento dos textos para acelerar esse processo, porém a retirada dessas palavras não afeta a compreensão do mesmo.

Após a definição destes conceitos, para que você os entenda melhor e consiga utilizá-los em seu cotidiano de programador é importante acompanhar a aplicação do que foi estudado. Nesse caso, será usando uma a biblioteca do python chamada de NLTK que significa Natural Language Toolkit. Então, com o ambiente do jupyter aberto, faça inicialmente o download da biblioteca com o comando “! pip install nltk”.

Feito isso, agora você deve importar a biblioteca com o comando “import nltk”, e então será necessário fazer o download das stop words em português. Primeiramente escreva o comando “nltk.download('stopwords' para fazer o download das stop words.

Então, escreva o comando “stopwords = nltk.corpus.stopwords.words('portuguese')” para obter as stop words em português.

Para que você consiga ter uma ideia de quais são as stop words, utilize o comando “print(stopwords)”.

Assim poderá fazer um teste simples para remover as stop words de uma frase. Então, utilize o módulo “word_tokenize” que como o nome sugere, retorna uma lista de tokens para um texto dado como entrada. Para fazer isso, você deve importar o módulo através do comando “from nltk.tokenize import word_tokenize”. É necessário também usar o comando “nltk.download(‘punkt’, pois estes são os recursos necessários para fazer a tokenização.

Logo, você poderá criar uma variável frase, que aqui será “frase = 'Eu dirijo devagar porque nós queremos ver os animais.'”. Então para dividir a frase em tokens, é só passar a variável frase para o módulo “word_tokenize”, fazendo “tokens = word_tokenize(frase)” e escrevendo “print(tokens)” caso queira entender como os tokens foram criados.

Com a lista de tokens criada, faça um loop simples usando a condição if para não imprimir na tela a stopword caso ela exista em cada token. Então na primeira linha do loop escreva “for t in tokens:” seguido da indentação, continue com “if t not in stopwords:”, finalizando com mais uma indentação e “print(t)”.

Com isso é possível perceber que foram impressos sete tokens, são eles: eu, dirijo, devagar, porque, queremos, ver, animais. Dentre eles não aparece na tela os tokens “nós” e

“os”, isso significa que estes dois são as stop words da variável frase, e ao removê-las a frase ainda continua fazendo sentido.

Interessante, não é mesmo?

Outra técnica muito importante na análise de textos, é a identificação da frequência e importância que uma palavra pode ter em um texto.

Existe um cálculo estatístico chamado de TF-IDF, essa sigla vem do inglês *Term Frequency – Inverse Document Frequency*, e quer dizer: Frequência do termo - Frequência Inversa do Documento. Este cálculo identifica a importância que um termo tem em um texto, ou seja, ele permite que você descubra quais termos são mais relevantes em um dado texto ou documento.

Em linhas gerais o TF-IDF busca atribuir um valor que representa um peso para definir a importância do termo em um documento, com base na frequência em que ele ocorre (TF), compensando, porém, esse peso, caso a ocorrência desse termo seja muito grande (IDF).

Em resumo, se um termo aparece algumas vezes no texto, o valor do TF-IDF aumenta, significando que aquela palavra é importante, porém se esse termo se repete bastante, esse valor é compensado, e a importância dele é diminuída.

Deu pra entender o TF-IDF? Agora acompanhe um exemplo prático, para melhorar sua compreensão sobre esse assunto, utilizando o módulo “TfidfVectorizer” da biblioteca sklearn para calcular os valores de TF-IDF de um texto.

Com o ambiente do jupyter aberto, primeiramente importe o módulo através do comando “from sklearn.feature_extraction.text import TfidfVectorizer”, e importe também a biblioteca pandas, fazendo “import pandas as pd”.

Em seguida crie uma variável texto1 para aplicar o cálculo do TF-IDF. Para isso, basta escrever texto1 = 'A matemática é muito importante para compreendermos como a natureza funciona'.

Então você poderá instanciar o módulo fazendo “tf_idf = TfidfVectorizer()”.

Feito isso use o método “fit_transform” que retorna uma matriz com os termos e os scores associados, passando o texto1 com o conteúdo. Em seguida para colocar no formato de um array use o comando “vetor = vetor.todense()”.

Para obter os nomes das palavras mapeadas, pode usar o código “nomes = tf_idf.get_feature_names()”.

Assim é possível associar os scores aos nomes das palavras e o resultado pode ser criado em um dataframe, fazendo “df = pd.DataFrame(vetor, columns=nomes)”.

Analisando o resultado dos scores das palavras você pode notar que o TF-IDF deu o mesmo score para todos os termos, pois só existe uma ocorrência de cada termo.

Porém se criar outro texto com o seguinte conteúdo: “texto2 = 'A matemática é incrível, quanto mais estudo matemática, mais eu consigo aprender matemática'”, e repetir os procedimentos feitos anteriormente para o texto1, poderá obter scores diferentes.

Logo perceberá que a palavra ‘matemática’ recebe o maior score, pois ocorre 3 vezes no texto. A palavra ‘mais’ recebe o segundo maior score, pois só aparece 2 vezes, e o restante recebe o mesmo score, pois só aparecem 1 vez.

Muita coisa, não é mesmo?

Até aqui você pôde aprender sobre o Processamento da Linguagem Natural de uma forma introdutória, e conhecer alguns conceitos importantes para o processamento de texto como, Corpus, Stopwords e Tokenização. Também entendeu como é realizado o cálculo TF-IDF para verificar a importância que um termo tem dentro de um texto. Além de tudo isso, pôde acompanhar a aplicação desses conceitos utilizando Python. Então, agora é com você! Tente praticar e resolver os exercícios propostos para aprofundar seus conhecimentos.

Bons estudos e até mais!