Processamento de linguagem natural e reconhecimento de voz

Conceitos de processamento de linguagem natural (PLN) e reconhecimento de voz a partir de demonstrações práticas.

Prof.ª Fernando Durier

1. Itens iniciais

Propósito

Para os profissionais de tecnologia da informação, é importante compreender conceitos e técnicas de processamento de linguagem natural (PLN), bem como reconhecimento de voz. Enquanto a área de inteligência artificial se ocupa de pesquisar e desenvolver sistemas que se comportem e pensem como humanos, a área de PLN se ocupa de fazer com que a máquina entenda e se comunique em linguagem natural também.

Preparação

Para compreender os exemplos, é imprescindível que você instale em seu computador a versão 3.8 ou superior da linguagem Python e as bibliotecas Pandas, Numpy, Sklearn e Plotly. Além disso, é necessária uma ferramenta para execução de notebooks, como Jupyter, que pode ser instalada como uma biblioteca de Python. Você também pode executar os códigos dos exemplos diretamente em um ambiente de execução, como o Google Colab. Realize **aqui** o download dos notebooks desenvolvidos.

Objetivos

- Definir os conceitos de processamento de linguagem natural.
- Aplicar técnicas de processamento de linguagem natural com NLTK.
- Aplicar técnicas de reconhecimento de voz com PyAudio.

Introdução

Atualmente, vivemos em um mundo em que produzimos dados e informações a todo o momento, e muito disse se deve às redes sociais e aos demais sistemas colaborativos disponíveis.

Podemos extrair dessas mensagens, postagens, notícias e livros, todo conhecimento útil capaz de ajudar agentes inteligentes a compreenderem o que estamos falando e a contribuir para a discussão ou mapear padrões de interesse em nossas interações. Assim, esses agentes podem nos recomendar outros assuntos relacionados ou possíveis respostas para perguntas que são feitas frequentemente por quem lê ou fala sobre determinado assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Conceitos de PLN

Confira agora os conceitos de PLN, incluindo inteligência artificial x processamento de linguagem natural.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Inteligência artificial X Processamento de linguagem natural

A inteligência artificial se ocupa de pesquisar e desenvolver sistemas capazes de adaptar suas soluções de maneira automática para novos problemas impostos, assim como nós. Para isso, a máquina precisa pensar, enxergar, ler, escutar e compreender como um ser humano.

Nesse contexto, o processamento de linguagem natural é muito importante, para que a máquina exerça funções como:

- Ler textos, a fim de extrair informações.
- Escutar falas e diálogos, a fim de contribuir com a discussão.
- Expressar-se em texto ou som, a fim de se fazer entender.

Por exemplo, a tecnologia robotics trends é um conceito de negócio de varejo inteligente, em que um robô serve de assistente pessoal autônomo para o cliente de navegação pesquisar itens no shopping center de moda.

Tecnologia robotics trends.

Tipo de dados em PLN

No processamento de linguagem natural, nosso conjunto de dados de interesse são os dados semiestruturados, ou seja, textos regidos pelas regras de construção e formação da linguagem, que podem ser desde uma reportagem até um artigo ou um informe de outro sistema.

> Para que a máquina entenda o que está se falando, primeiro é necessário organizar o dado semiestruturado (texto) em algo mais compressivo para o computador: os números.

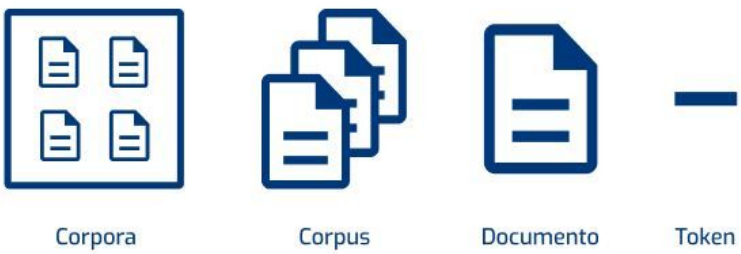
Um texto é composto de sentenças e que, por sua vez, são compostas de sujeito, verbo e predicado. Cada uma dessas partes é composta essencialmente por palavras. Desse modo, concluímos que os níveis hierárquicos de nossos dados são: texto, sentenças e palavras. O conjunto de textos é chamado de corpus, e o conjunto de corpus é chamado de corpora.

Comentário

Decompor o texto em partes facilita a operacionalização dos dados, pois podemos quebrá-los em partes de interesse à máquina. Esses átomos são os tokens que, na maioria das vezes, são feitos ao nível de palavra: os word tokens.

Dependendo do tipo de problema de PLN, podemos ver tokenização por sentenças, mas isso é raro. Assim, nosso texto passa a ser um vetor de palavras.

Em resumo, a hierarquia dos dados é apresentada do seguinte modo:



Hierarquia de dados de texto.

Tarefas de PLN

Depois que decompomos o texto em partes atômicas, devemos analisá-las de acordo com as subáreas de processamento de linguagem natural, a saber:

Análise sintática

Identifica cada classe gramatical de cada token, ou seja, se uma palavra é verbo, substantivo etc.

Análise semântica

Entende o sentido do uso das palavras, em que, normalmente, entra a análise de sentimentos.

Análise pragmática

Relaciona e constrói sentenças e referências, como a desambiguação de termos, a relação de palavras etc.

Com pelo menos a análise sintática feita, podemos começar a transformar textos em estruturas matemáticas.

TF-IDF e matematização dos textos

Term frequency inverse document frequency of records (TF-IDF), ou frequência do termo pelo inverso da frequência do documento, pode ser definido como o cálculo da relevância de uma palavra em uma série ou corpus para um texto. O significado aumenta proporcionalmente ao número de vezes que uma palavra aparece no texto, mas é compensado pela frequência da palavra no corpus (conjunto de dados).

No caso do TF-IDF, por exemplo, contamos quantas vezes determinada palavra aparece no corpus e em cada documento. Com essa técnica poderosa, podemos descobrir quais são as palavras mais frequentes e codificar os documentos.

> Palavras mais frequentes são um bom indicativo de o documento pertencer a uma dada categoria.

Por exemplo, em notícias de economia, devem aparecer muitas vezes os termos **dinheiro** e **negócios**; em notícias de esportes, são mais frequentes termos como: bola, juiz ou semelhantes.

Também podemos limpar nosso conjunto de dados removendo palavras que aparecem em excesso, como artigos, preposições e termos auxiliares de linguagem. Essas palavras são determinadas como stopwords. Além disso, podemos ignorar termos que só aparecem uma vez ou pouquíssimas vezes no corpus: as outliers. Veja a seguir:

Stopwords

Palavras que não acrescentam muito na informação em questão.

Outlies

Palavras que não acrescentam no ganho de informação.

Na análise de importância das palavras, para identificar aquelas que realmente podem agregar significado a uma sentença, é usual empregarmos a técnica da frequência inversa. Com o TF-IDF, consideramos a frequência de uma palavra na sentença, dividida pelo número de documentos em que ela aparece. Assim, temos:

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{\mathrm{N}}{\mathrm{df}_i}\right)$$ tfi, j = total number of occurences of i in j = número total de ocorrências de i em j dfi = total number of documents (speeches) containing i = número total de documentos (falas) contendo iN = total number of documents (speeches) = número total de documentos (falas)

Uma stopword (palavra de parada) é um termo comumente usado (como o, um, uma, em) que um mecanismo de pesquisa foi programado para ignorar, tanto ao indexar entradas para pesquisa quanto ao recuperá-las como resultado de uma consulta de pesquisa. Então, podemos reexecutar nossa tokenização, removendo as palavras indesejadas, e obter um relatório de frequência mais eficaz, que, por sua vez, pode ser usado para codificar os textos de um corpus.

O modelo Doc2Vec é usado para criar uma representação vetorial de um grupo de palavras tomadas coletivamente como uma única unidade, ou seja, é quando usamos uma técnica de transformação de texto em

vetorização. Uma dessas técnicas é o TF-IDF, no qual cada texto é um vetor de vocabulário – vetor no qual cada posição é um possível termo presente no corpus –, cujos valores de suas células são justamente a frequência do dado termo no dado documento/texto.

Exemplo

“O rato roeu a roupa do rei de roma” – (removemos stopwords) → [palavra 0, ..., rato, roeu, roupa, rei, roma, ..., palavra n] → [0.0,...,0.2, 0.01, 0.01, 0.1, 0.02, 0.03,...,0.0]. Os números são o TF-IDF de cada palavra codificada, em que, por exemplo, rato teria 20%. Logo, a frase poderia ser codificada como o vetor de frequências.

Com isso, podemos associar rótulos a cada vetor codificado e treinar diversos modelos para que a máquina possa entender e associar futuros textos em dada classificação.

Técnicas de PLN

Confira a apresentação das técnicas de PLN utilizando o notebook em execução.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Coletando dados textuais

Podemos coletar dados textuais de diversas maneiras: entrevistas, formulários de satisfação ou até escrevendo/transcrevendo material físico para o computador. Hoje, é muito comum usar uma técnica chamada de web scraping, que extrai de páginas web seu conteúdo por meio de código.

Um bom exemplo para praticarmos é extrair dados do Twitter (X). Para isso, você deve utilizar uma base de dados extraída do Twitter (X). Para tanto, você pode utilizar uma API da rede social ou procurar uma base de dados do Github ou GitLab. Para testar você deve garantir que sua base extraída ou lida seja chamada de tweets e possua o texto dos tweets e o nome do usuário.

Agora, vamos iniciar a implementação de uma solução de PLN utilizando a linguagem Python.

Curiosidade

Em geral, o mundo da computação é colaborativo. É comum encontrarmos erros nas bibliotecas utilizadas, copiarmos o log de erro e procurarmos em qualquer motor de busca. Provavelmente, alguma resposta será retornada em algum fórum de discussão, como StackOverflow, Github, Reddit. Isso não só é uma prática comum na comunidade de desenvolvimento de software e computação, como também nos possibilita aprender mais.

Após a extração dos, utilizaremos a biblioteca Pandas, uma das mais importantes na análise de dados para Python. É a ferramenta principal para construção de estrutura, manipulação e limpeza de dados, também utilizada com bibliotecas de processamento numérico e construção de gráficos.

Os dois principais objetos da biblioteca Pandas são:

Series

Matrizes unidimensionais com sequência de valores que apresentam uma indexação (numéricos inteiros ou rótulos). É muito parecida com uma única coluna no Excel.

DataFrames

Estruturas de dados tabular, semelhante à planilha de dados do Excel, em que tanto linhas quanto colunas apresentam rótulos.

Seguindo nosso exemplo prático, podemos transformar o conjunto de tweets recuperados em um dataframe com a biblioteca Pandas da seguinte maneira:

```
python

import pandas as pd
tdf = pd.DataFrame(tweets).reset_index(drop=True)
```

Preparação do conjunto de dados

Com os dados recuperados do Twitter, vamos tornar esse conjunto útil aos modelos de aprendizado de máquina. Começaremos com a remoção de URLs e alguns sinais especiais de pontuação. Faremos isso por meio da biblioteca re (regular expressions). Em ciência da computação, uma expressão regular ou "Regex" provê uma forma concisa e flexível de identificar cadeias de caracteres de interesse, como caracteres particulares, palavras ou padrões de caracteres.

Execute os comandos a seguir no notebook:

```
python

import re
clean_regex = r'((?:\@|https?\://)\S+)|(\b(?:(?:https?|ftp)://)?\w[\w-]*(?:\.[\w-]+)+\S*(?
```

Vamos efetivamente limpar os textos, executando o comando a seguir no notebook:

ATENÇÃO: repare que o “text” refere-se à coluna onde estão os tweets, no seu caso troque o text pelo nome da coluna em seu DataFrame e não esqueça das aspas.

```
python
tdf['text'] = tdf['text'].apply( lambda x: clean_text(x) )
```

Agora, vamos tokenizar nossos textos. Para isso, utilizaremos a Natural Language Toolkit (NLTK), biblioteca de PLN mais popular em Python.

> NLTK é um conjunto de bibliotecas e programas para processamento simbólico e estatístico de linguagem natural para inglês escrito em Python.

Para utilizá-la, vamos instalá-la por meio do comando pip install. Também vamos baixar com seu próprio gerenciador de recursos alguns artefatos importantes, para que a NLTK possa funcionar. Faremos isso executando os comandos a seguir como prompt de comando:

```
plain-text
pip install nltk
import nltk
nltk.download('punkt')
nltk.download("stopwords")
nltk.download('averaged_perceptron_tagger')
```

Leia o aviso a seguir:

> ⚠
>
> Atenção
>
> À medida que trabalhamos com a NLTK, pode ser que ela interrompa o funcionamento e peça para baixar outros recursos para completar as tarefas. Isso é normal. Basta observar o erro que ela emite, pois já haverá instruções do que baixar para seguir.

A tokenização é o processo de dividir uma frase em palavras ou tokens individuais. Durante esse processo, pontuações e caracteres especiais são completamente removidos. Execute os comandos a seguir no notebook:

```
python

from nltk.tokenize import word_tokenize
import nltk
tdf['word_tokens'] = tdf['text'].apply( lambda x: word_tokenize(x) )
```

Conforme aprendemos, as famosas stopwords são palavras que podem ser consideradas irrelevantes para o conjunto de resultados a ser exibido em uma busca realizada em uma ferramenta de busca.

Exemplo

as, e, os, de, para, com, sem, foi.

Lembre-se de que, para cada língua, as stopwords mudam. Em inglês, esses termos seriam **the**, **a**, **as** etc. Esse processo de remoção de stopwords faz parte do pré-processamento de dados realizado nas etapas iniciais de um pipeline de PLN. Quando montamos um bag-of-words, por exemplo, as stopwords são mais frequentes, uma vez que são palavras utilizadas o tempo todo para dar sentido ao texto. Portanto, removê-las reduz o ruído dos dados analisados.

Agora, vamos determinar stopwords de forma básica, executando os comandos a seguir no notebook:

```
python

import string

stopwords = nltk.corpus.stopwords.words("english") + string.punctuation.split() +
["@","#","$","%","&","*","(",")","-","_","=","+"]

def remove_stopwords(words):
    return [w for w in words if w not in stopwords]
```

Depois, efetivamente, vamos removê-las. Na sequência, extraímos as POS_Tags (etiquetas de partes do texto), em que a biblioteca utiliza dicionários de idiomas para definir as classes gramaticais de cada termo. Fazemos uma contagem para identificar quantas palavras efetivas cada tweet possui, executando os comandos a seguir no notebook:

```python
tdf['word_tokens'] = tdf['word_tokens'].apply(lambda x: remove_stopwords(x))
tdf['POS_Tag'] = tdf['word_tokens'].apply(lambda x: nltk.pos_tag(x))
tdf['number_of_words'] = tdf['word_tokens'].apply(lambda x: len(x))
tdf_filtered = tdf[tdf['number_of_words'] > 30]

print(tdf_filtered)
```

Análise de sentimentos

Confira agora a análise de sentimentos e a modelagem em PLN no notebook em execução.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vamos fazer uma análise de sentimentos rápida? É muito simples! Vamos apostar na **estratégia semissupervisionada**.

Para isso, usaremos uma lista de lexicons previamente calculada com a carga sentimental de cada palavra no vocabulário inglês (VaderLexicon), executando os comandos a seguir no notebook:

Estratégia semissupervisionada

Aprendizado supervisionado em que os dados de treinamento contêm poucos exemplos rotulados e muitos não rotulados.

```python
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
```

Depois, aplicamos o analisador em cada vetor de tokens, executando os comandos a seguir no notebook:

```
python

tdf['sentiment_polarity'] = tdf['word_tokens'].apply(lambda x: sia.polarity_scores( "
".join(x) )['compound'] )
```

Com isso, podemos até fazer uma limpeza prévia, pegar textos com pelo menos uma palavra ou algo assim, executando os comandos a seguir no notebook:

```
python

grouped_df = tdf.groupby(['username']).agg(['count','mean','min','max']).reset_index()
grouped_df.columns = ['_'.join(col).strip() for col in grouped_df.columns.values]
filtered_df = grouped_df.query("number_of_words_count > 1")
stop_words = ['a', 'an', 'the', 'and', 'but', 'or', 'on', 'in', 'with', 'for', 'to']
#Exemplo simplificado
```

Agora, podemos fazer nosso TF-IDF. Para isso, vamos utilizar o sklearn para nos auxiliar. Essa é uma biblioteca muito utilizada em Python, que contém muitas ferramentas eficientes para aprendizado de máquina e modelagem estatística, incluindo classificação, regressão, agrupamento e redução de dimensionalidade.

Faremos uma função para rodar nossos dataframes, executando os comandos a seguir no notebook:

```
python

from sklearn.feature_extraction.text import TfidfVectorizer
def tf_idfer(text_df):
    vocab = []
    for v in text_df.values:
        vocab+=v

    tfidf = TfidfVectorizer(tokenizer=word_tokenize, stop_words=stop_words)
    tfs = tfidf.fit_transform( vocab )
    X = tfidf.transform( [ " ".join(v) for v in text_df.values ] ).toarray()
    return X
```

Rodamos e armazenamos o resultado em uma nova variável, executando os comandos a seguir no notebook:

```
python

X = tf_idfer( tdf["word_tokens"] )
xdf = pd.DataFrame(X,columns=["tweet_"+str(c) for c in range(X.shape[1]) ])
```

Pronto! Temos uma transformação de texto para dados numéricos.

Modelagem em PLN (classificação de textos)

Com o texto devidamente passado para forma compreensiva à máquina, podemos fazer uma modelagem.

O scikit-learn, ou sklearn, é uma biblioteca gratuita, de código aberto, para machine learning (aprendizado de máquina) em Python. Essa biblioteca também fornece uma seleção de recursos eficientes para modelagem estatística, análise e mineração de dados.

Verificando o aprendizado

Questão 1

O PLN permite que a máquina compreenda o contexto da mensagem, a partir, por exemplo, de análises morfológicas, semânticas ou sintáticas. Assim, é possível criar respostas automáticas que atendam às demandas específicas apresentadas pelos clientes. Nesse contexto, qual é a hierarquia de dados para PLN?

A

Imagem → Corpus → Texto → Sentença → Sílaba

B

Corpora → Texto → Sentença → Palavra → Letra

C

Banco de Dados → Texto → Sentença → Palavra → Preposição

D

Corpora → Texto → Corpus → Palavra → Sentença

E

Corpora → Corpus → Texto → Sentença → Palavra

A alternativa E está correta.

A hierarquia correta é:

- Corpora – conjunto de corpus, como um ministério ou uma sociedade de bibliotecas.
- Corpus – conjunto de textos, como uma biblioteca.
- Texto – um documento, um livro, um conto, uma notícia ou até mesmo um tweet.
- Sentença – sujeito + verbo + predicado.
- Palavra – nível atômico de interesse.

Questão 2

Em processamento de linguagem natural, a primeira coisa que devemos fazer é permitir que os dados textuais sejam compreensíveis para as máquinas poderem trabalhar. Qual é a técnica mais básica utilizada para realizar essa transformação?

A

KMeans

B

Naive Bayes

C

Doc2Vec

D

TF-IDF

E

Análise sintática

A alternativa C está correta.

Embora TF-IDF seja uma possibilidade de operacionalização do Doc2Vec, a resposta é Doc2Vec, quando utilizamos um algoritmo de codificação para associar cada texto a uma forma vetorial. Kmeans e Naive Bayes são algoritmos de aprendizado de máquina, que podem ser usados em etapas posteriores. Análise sintática é uma das tarefas de PLN, responsável por analisar as classes gramaticais e os artefatos linguísticos presentes no texto por meio de seus termos.

2. Técnicas de reconhecimento de voz

PLN: pré-processamento

O reconhecimento de voz é uma subárea da computação que se preocupa em estudar maneiras de ler, filtrar e interpretar os sinais sonoros capturados pelo microfone dos computadores para deles extrair alguma informação relevante.

Essa informação relevante se subentende como a transcrição do áudio, que, eventualmente, pode ser inserida em sistemas capazes de transformar essa transcrição em insumo para as técnicas de PLN, a fim de extrair tokens, a categorização desses tokens, a análise no tom de voz, e assim por diante.

O computador entende as letras e forma as palavras a partir dos sinais sonoros, de modo semelhante ao processo de aprendizado supervisionado, no qual os conjuntos de treinamento para esses motores são linhas de áudio com seus respectivos metadados. Uma letra ou palavra é associada a cada linha.

Esses sistemas são treinados. Quando os utilizamos, podem associar nossa fala com a transcrição adequada. É claro que existem diferentes modelos para distintas linguagens.

> Não adianta tentar interpretar uma fala em português com um transcritor em inglês.

Tokenização de sentenças

Confira agora, por meio do notebook em execução, a tokenização de sentenças em PLN.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Com a linguagem Python, o processamento de áudio é muito facilitado. Primeiro, vamos instalar as dependências Pyaudio, SpeechRecognition e NLTK.

Dica

O PyAudio varia suas configurações de instalação de acordo com o sistema operacional e a máquina usada. Logo, em caso de erros, utilize os logs fornecidos pela biblioteca. Ao pesquisar na internet, você obterá um caminho de conserto alternativo.

PyAudio é uma biblioteca de áudio que permite usar Python facilmente para reproduzir e gravar áudio em uma variedade de plataformas. Já a biblioteca SpeechRecognition foi desenvolvida para realizar reconhecimento de voz.

Vejamos sua instalação no prompt de comando:

```
plain-text

pip install pyaudio Speechrecognition nltk
```

Feito isso, precisamos instalar alguns recursos adicionais da biblioteca NLTK para processar áudios em nossa linguagem (português), executando os seguintes comandos no notebook:

```
python

import nltk
nltk.download('punkt')
nltk.download("stopwords")
nltk.download('mac_morpho')
from nltk.corpus import stopwords

stop_words = set(stopwords.words("portuguese"))
```

Vamos agora executar esses comandos no notebook a fim de determinar as funções de tokenização:

```
python

import nltk
from nltk.tokenize import sent_tokenize, word_tokenize

def sentence_tokenization(text):
    try:
        return sent_tokenize(text)
    except Exception as e:
        print("Sent Tokenization Error: ", e)
        return []

def word_tokenization(text):
    try:
        return [token for token in word_tokenize(text) if token not in stop_words]
    except Exception as e:
        print("Word Tokenization Error: ", e)
        return []
```

Etiquetamento das palavras

Confira agora, por meio do notebook em execução, o taggeamento de palavras em PLN.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Após a tokenização, vamos taggear (criar etiquetas) as palavras. As etiquetas morfossintáticas atribuídas às palavras permitem indicar sua função léxica. Para isso, precisaremos fazer algo especial, uma vez que o NLTK

naturalmente só suporta o inglês.

A tabela a seguir mostra o conjunto de etiquetas usadas pelo MacMorpho – um corpus em português do Brasil formado de palavras validadas manualmente com anotações morfossintáticas:

Classe gramatical	Etiqueta

Conjunto de etiquetas usadas pelo MacMorpho.
Barbosa *et al.*, 2017, n. p.

Agora que vimos alguns conjuntos de etiquetas, vamos treinar um pos_tagger com o corpus em português Mac_Morpho executando os comandos a seguir no notebook:

```
python

from nltk.corpus import mac_morpho

def simplify_tag(t):
     if "+" in t:
        return t[t.index("+")+1:]
     else:
        return t

tsents = mac_morpho.tagged_sents()
tsents = [[(w.lower(),simplify_tag(t)) for (w,t) in sent] for sent in tsents if sent]
train = tsents[100:]
test = tsents[:100]

tagger0 = nltk.DefaultTagger('n')
tagger1 = nltk.UnigramTagger(train, backoff=tagger0)
tagger2 = nltk.BigramTagger(train, backoff=tagger1)

def word_pos_tag(word_tokens):
    pts = tagger1.tag(word_tokens)
    return pts
```

O truque é usar a função simplify tag para deixar as tags mem um formato mais agradável. Depois, treinamos um tagger default, caso ocorram erros de tokenização. Na sequência, treinamos um tagger para unigrama (marcador de um termo apenas) e, em seguida, um para bigrama (marcador para dois termos seguidos). Dando continuidade, colocamos o backoff do unigrama e do bigrama para seus antecessores: tagger0 e tagger1, respectivamente. Por fim, definimos uma função word_pos_tag para interfacear o método e ficar uniforme.

Reconhecimento de voz

Confira agora, utilizando o notebook em execução, o reconhecimento de voz em PLN, incluindo configurações para o processamento de áudio e speech recognition.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Configurações para processamento de áudio

Para realizar o reconhecimento de voz, vamos executar os seguintes comandos no notebook:

```
python
```

```
# reprodução de arquivo de áudio em disco
import IPython
import pyaudio
import wave
import speech_recognition as sr

CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "output.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)

print("* recording")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

Podemos observar que, por meio do código, estamos importando os seguintes itens, nesta ordem:

1. As bibliotecas de reconhecimento de voz e processamento de áudio (SpeechRecognition e Pyaudio).
2. O IPython, que nos sevirá no Jupyter Notebook para podermos chamar o player de áudio.
3. A biblioteca wave, para leitura de arquivos de áudio.

Em seguida, configuramos os seguintes parâmetros de gravação:

- O chunk de 1024.
- O formato bytes de int16 do próprio Pyaudio.
- Channels 1 (normalmente são 2, mas, para nosso exemplo, somente funcionou com 1).
- O rate de 44100.
- Record_seconds para 5 (ou seja, 5 segundos de gravação apenas).
- O path de salvamento do arquivo, que será .wav.

Como um stream, abrimos o cliente do PyAudio com os parâmetros e gravamos nossa voz pelo microfone do próprio computador. Por fim, processamos o stream como frames, para assim os salvarmos como arquivo output.wav.

Speech recognition

Speech recognition (reconhecimento de fala) é um campo da ciência da computação com o objetivo de desenvolver tecnologias que permitam o reconhecimento e a transcrição da linguagem falada por nossos computadores.

O ato de falar nada mais é que criar vibrações no ar. Por meio de um conversor, essas vibrações podem ser convertidas em dados digitais compreendidos pelo computador. Para isso, ele coleta ou digitaliza o som, realizando medições precisas da onda em intervalos de frequência, que é o comprimento que as ondas sonoras têm, percebidas pelos seres humanos como diferenças de afinação.

Retornando a nosso exemplo com o IPython, vamos tocar no Jupyter Notebook a gravação de 5 segundos que fizemos. Para isso, basta usarmos este código:

```
python

print("Reproducing Audio:")
IPython.display.Audio(WAVE_OUTPUT_FILENAME)
```

Observe na imagem a seguir:

Nesse exemplo, foi dita a seguinte frase: “Batata quando nasce espalha a rama pelo chão”. Para conseguirmos que o computador entenda isso, vamos executar os seguintes comandos no notebook:

```
python

r = sr.Recognizer()

with sr.AudioFile(WAVE_OUTPUT_FILENAME) as source:
    read_audio = r.record(source)  # read the entire audio file

transcribed_audio = r.recognize_google(
    audio_data=read_audio,
    language="pt-BR"
)

print( "Transcription: ", transcribed_audio )
```

Instanciaremos o cliente da biblioteca SpeechRecognition e para a qual passaremos, a partir do método AudioFile, o caminho em que nosso áudio foi salvo. Depois, vamos usar o método recognize_google() para passar os dados da gravação e especificar o idioma falado.

Por fim, poderemos visualizar a transcrição:

```
result2:
{   'alternative': [   {   'confidence': 0.91506195,
                           'transcript': 'Batata quando nasce espalha ram
a '
                                         'pelo chão'},
                       {   'transcript': 'Batata quando nasce espalha a r
ama '
                                         'pelo chão'},
                       {   'transcript': 'a batata quando nasce espalha r
ama '
                                         'pelo chão'},
                       {   'transcript': 'a batata quando nasce espalha a
rama '
                                         'pelo chão'}],
    'final': True}
Transcription:  Batata quando nasce espalha rama pelo chão
```

Visualização da transcrição do áudio.

O resultado esperado deve ser similar ao da figura. Veja temos algumas alternativas. A escolhida é a primeira, com 0.915 de confiança.

Com a transcrição em mãos, podemos fazer a tokenização com nosso tagger em português, executando os comandos no notebook:

```
python

word_tokens = word_tokenization( transcribed_audio.lower() )
word_pos_tag(word_tokens)
```

O resultado esperado de POS_Tag será o seguinte:

Resultado de POS_Tag.

Verificando o aprendizado

Questão 1

Para um computador entender qualquer informação, precisamos traduzir essa entrada de dados para um formato aceito pela máquina, seja numérico seja por bytes. Considerando essa máxima, a máquina compreende nossas vozes por meio de que tipo de entrada de dados?

A

Dicção.

B

Fonemas.

C

Sílabas faladas.

D

Sinais de ondas sonoras.

E

Gestos com as mãos para passar mensagens.

A alternativa D está correta.

O reconhecimento de voz parte da inteligência artificial, cujo objetivo é permitir a comunicação falada entre seres humanos e computadores eletrônicos por meio de processos de classificação de sinais em sequências de padrões, a fim de processar a mensagem contida na onda acústica.

Questão 2

O processamento de linguagem natural (PLN) é uma área interdisciplinar do conhecimento que trabalha com a interação entre computadores e humanos, usando a linguagem natural, presente em diversos serviços na atualidade. Nesse contexto, avalie as afirmativas a seguir:

I. MacMorpho é um corpus de textos em português do Brasil anotados com tags de parte do discurso.

II. PyAudio é uma biblioteca que permite usar o Python para reproduzir e gravar áudio em várias plataformas, como GNU/Linux, Microsoft Windows e Apple macOS.

III. Etiquetamento é o processo de dividir uma frase em palavras, permitindo uma posterior tokenização.

Quais são as afirmativas corretas?

A

I, II e III.

B

I e III.

C

I e II.

D

II e III.

E

III.

A alternativa C está correta.

A tokenização é o processo de tokenizar ou dividir uma string em uma lista de tokens, que são partes de um texto. Por exemplo, uma palavra é um token em uma frase, e uma frase é um token em um parágrafo.

3. Aplicações de PLN

Importância do processamento de linguagem natural

Confira agora a importância do processamento de linguagem natural.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Agentes inteligentes

O processamento de linguagem natural é fundamental. A sociedade moderna só está no ponto atual de evolução e avanço graças à linguagem e à escrita, que permitiram o desenvolvimento de comunicação, registros em memória e abstrações de pensamento. Até mesmo a matemática é uma linguagem, assim como a programação.

É muito importante que agentes inteligentes sejam capazes de se deparar com nossa linguagem, seja por áudio, escrita e visualização. Esse conhecimento é fundamental para se adequarem ao contexto e resolverem o problema imposto.

Exemplo

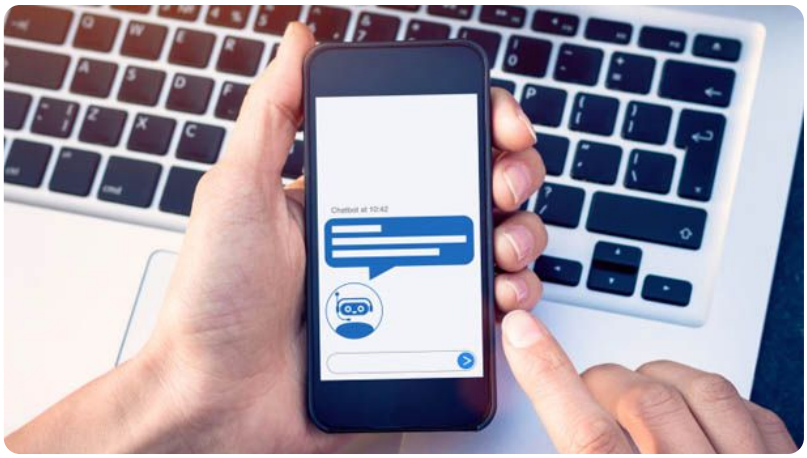
É por meio de notícias que importantes decisões são tomadas nas grandes bolsas de valores, em que um relatório de rendimentos ou desempenho divulgado por empresas pode estimular ou afastar investidores. Em outro contexto, na procura de um bem ou serviço de uso pessoal, é comum o pesquisarmos em redes sociais ou sites para entender melhor sua reputação/avaliação antes de o comprarmos.

Máquinas capazes de entender essas informações disponibilizadas na rede podem agregar essa informação, enriquecendo seus conjuntos de treinamento e regras lógicas para aprimorar suas decisões.

Casos conhecidos

Existem muitos casos conhecidos de processamento de linguagem natural. Vamos ao mais comum e simples. Pegue seu celular e chame seu assistente. Basta dizer “Ok, Google”, “Hello, Siri” ou apertar um botão em aparelhos Samsung para chamar a Bixby.

Esses assistentes virtuais são chatbots ativados por voz, que interpretam e transcrevem o sinal sonoro. A partir disso, retornam a resposta mais adequada para a questão imposta. Um jeito de fazer isso é ter um conjunto de perguntas e respostas para treinar um classificador ou um sistema de recomendação.



Atendimento por chatbot.

Outro caso são os chatbots de lojas ou hospitais, a partir dos quais podemos pedir ajuda ao SAC ou marcar consultas, por exemplo. O princípio é semelhante ao dos assistentes pessoais. A diferença, nesse caso, é que há mais entradas de dados em texto, e não áudio.

Além disso, podem ser desenvolvidos sistemas mais poderosos e específicos para determinadas tarefas graças a subsistemas de gerenciamento de diálogo, como o Watson Assistant ou a biblioteca Python Chatterbot.

Também podemos usar PLN para análise de marketing, quando sistemas vasculham a web por meio de web crawling e web scraping, a fim de recuperar o máximo de informação possível sobre determinado assunto, produto, serviço ou organização.

Com esse conjunto em mãos, aplicamos as técnicas de PLN para mapear o perfil de possíveis clientes, a opinião da população e a reputação da organização ou do serviço.

Comentário

Normalmente, os resultados de mapeamento com técnicas de PLN são nuvens de palavras, mapas de conexão ou mapas de calor de opinião.

Demonstração de captura de dados textuais e tokenização

Confira agora uma demonstração de captura de dados textuais e tokenização.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Tokenização de sentenças é uma atividade muito relevante na área de PLN, principalmente em português. Afinal, podemos isolar alvos de intenção de escrita de um locutor, pois as sentenças são compostas por sujeito, verbo e predicado. Dentro do predicado, normalmente, encontra-se o alvo da ação, que é o verbo da sentença. E o sujeito, em geral, é o locutor.

Para possibilitar isso, podemos usar os mesmos métodos anteriores:

```
python
from nltk import sent_tokenize
```

No caso do NLTK, podemos usar o sent_tokenize para nos ajudar. Com esse código, passamos um texto e, por meio da pontuação, o método será capaz de separar as sentenças do texto. Devemos fazer isso antes da limpeza do corpus, pois, ao removermos pontuação, o método se perde ao ler o texto.

No exemplo em que buscamos tweets, vamos acrescentar essa célula, que fará uma amostragem de 10 tweets aleatórios daqueles que mineramos do Twitter:

```
python
tdf_samples = tdf.sample(10)
```

Feito isso, vamos chamar a função lambda nos textos. Por meio do método de sent_tokenize, alocaremos o resultado em uma nova coluna do dataframe, que será sents:

```
python
tdf_samples["sents"] = tdf_samples["text"].apply( lambda x: sent_tokenize(x) )
```

Ainda com essa coluna em mente, iremos medir o tamanho de cada sentença. Como estamos no domínio do Twitter, serão mais comuns casos de uma única sentença, mas, com certeza, algumas pessoas escrevem mais. Por isso, ao extrairmos a medida, podemos obter registros que têm mais de uma sentença:

```python
tdf_samples["sents_size"] = tdf_samples["sents"].apply(lambda x: len(x))
```

Com essa medida, podemos filtrar exemplos com mais de uma sentença para visualizarmos a separação. Lembre-se de que, caso não ocorram tweets com mais de uma sentença em sua amostragem, basta refazê-la. Logo:

```python
tdf_samples.query("sents_size > 1")
```

Vejamos o seguinte exemplo:

Exemplo

"Travel code and health code played a huge role in tracking down potential infected people. Now, facing the new strategy towards covid, it's time to say goodbye.""O código de viagem e o código de saúde desempenharam um papel importante no rastreamento de possíveis pessoas infectadas. Agora, diante da nova estratégia em relação ao covid, é hora de dizer adeus."

Esse exemplo é um tweet, de 13 de dezembro de 2022, com duas sentenças. O autor elogiou as medidas de precaução da covid-19 nos processos de viagem e saúde para identificação de doentes, em seguida refletiu que a nova estratégia de lidar com essa doença pode não ser tão eficaz e causar novas ocorrências graves (*time to say goodbye*). Com a separação de sentenças, podemos entender melhor impressões, sarcasmos e até mesmo correferências.

Classificação com árvore de decisão

Confira agora, utilizando o notebook em execução, uma demonstração de classificação com árvore de decisão.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Investigue a documentação da biblioteca scikit-learn, pois utilizaremos a classe de algoritmos de classificação GaussianNaiveBayes para fazer o treino e a classificação de nosso conjunto de dados de tweets.

Considere os seguintes cenários:

Exemplo

Você é um gerente de produto que deseja categorizar o feedback do cliente em duas categorias – favorável e desfavorável. Você é um gerente de produto que deseja categorizar o feedback do cliente em duas categorias – favorável e desfavorável. Como analista de saúde, você deseja prever quais pacientes provavelmente desenvolverão complicações diabéticas.

Todas as instâncias têm o mesmo tipo de desafio quando se trata de categorizar revisões, pedidos de empréstimo e pacientes.

> Naive Bayes é o método de classificação mais fácil e rápido, disponível e adequado para lidar com grandes quantidades de informações.

Em diversas aplicações, como filtragem de spam, classificação de texto, análise de sentimentos e sistemas de recomendação, o classificador Naive Bayes tem se mostrado eficaz. Ele faz previsões sobre classes desconhecidas, usando a **teoria de probabilidade de Bayes**. Essa classificação Naive Bayes está disponível em Python na biblioteca Sklearn.

Como mineramos textos e temos até maneiras de rotulá-los de forma semiautomática, com as queries de busca que fazemos nos motores da internet ou redes, podemos treinar classificadores para nos ajudar com futuras entradas de texto. Para isso, podemos usar um modelo clássico: a árvore de decisão.

Teoria de probabilidade de Bayes

Teoria baseada no teorema de Bayes, fórmula matemática usada para o cálculo da probabilidade de um evento dado que outro evento já ocorreu, o que é chamado de probabilidade condicional.

Comentário

A árvore de decisão é um algoritmo de separação de dados de acordo com o ganho de informação, à medida que cada nó interno da árvore torna o conjunto observado cada vez mais uniforme em uma das classes em questão.

Para isso, vamos usar a biblioteca do scikit learn e o método DecisionTreeClassifier, digitando o seguinte código no Jupyter Notebook:

```
python

from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

Feito isso, vamos nos aproveitar do primeiro exemplo com tweets e pegar o mesmo conjunto xdf. Dessa vez, vamos fazer um split simples:

```python
y = xdf["topic"]
X = xdf[ [ c for c in xdf.columns if c!= "topic" ] ]
```

Com isso, vamos realizar a separação em conjuntos de treino e teste, fazendo o split tradicional de 66% para treino e 33% para teste:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Assim, podemos treinar os dois classificadores e compará-los para saber se estão na direção certa. O correto seria a decision tree superar a acurácia do naive bayes. Afinal, a decision tree é um algoritmo considerado mais capaz. Já o naive bayes é considerado baseline ou caso base, pois assume independência entre as colunas, o que nem sempre é verdade. Por isso, ele é usado como algoritmo de comparação entre outros.

Para treinarmos os algoritmos, instanciaremos cada um, chamando os métodos fit de cada um para o mesmo conjunto de treino. Depois, mediremos os scores para cada conjunto de teste:

```python
dec_tree_classifier = DecisionTreeClassifier(random_state=0)
gnb_classifier = GaussianNB()
dec_tree_classifier.fit(X_train, y_train)
gnb_classifier.fit(X_train, y_train)
```

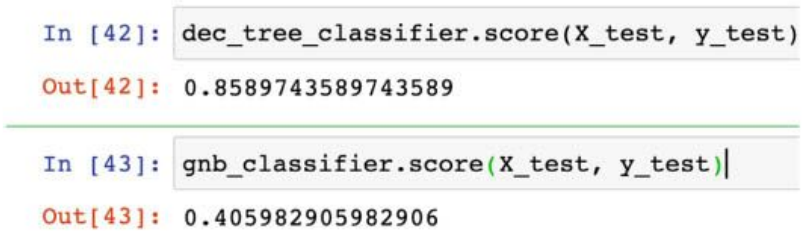
Agora, com os algoritmos treinados, vamos chamar os scores. Primeiro, o score da decision tree:

```python
dec_tree_classifier.score(X_test, y_test)
```

Depois, o score do naive bayes:

```python
gnb_classifier.score(X_test, y_test)
```

Como esperado, a árvore de decisão se saiu melhor do que o naive bayes: mais que o dobro em acurácia. Isso é importante porque, agora, para próximos tweets aleatórios recebidos ou qualquer outro texto de mesmas proporções, o modelo pode rotulá-lo como uma das categorias para as quais foi treinado, como visto no nosso exemplo de mineração de Twitter. Desse modo, temos:



Resultado: árvore de decisão X naive bayes.

Verificando o aprendizado

Questão 1

Quando falamos de rotulação automática de textos, nos referimos a um problema de aprendizado de máquina chamado de classificação, no qual um algoritmo aprende a mapear características de um conjunto de atributos para um(a) dado(a) classe/categoria/rótulo. No caso de uma classificação de textos simples, qual algoritmo podemos usar?

A

Web Crawler

B

Tokenização

C

Naive Bayes

D

Web Scraper

E

Corpus

A alternativa C está correta.

Um conjunto de textos pode ser rotulado automaticamente por um algoritmo de naive bayes. Ainda que não seja o melhor, pode ser muito útil, pois a maioria dos detectores de spam são naive bayes.

Questão 2

A computação afetiva é um dos temas recentes e muito polêmicos da atualidade: há sistemas se passando por pessoas famosas, conhecidas ou até mesmo entes queridos. Esses sistemas que conversam com os humanos tentando se passar por outros são chamados de

A

chatbots.

B

mensageiros.

C

correios eletrônicos.

D

corpora.

E

SVM.

A alternativa A está correta.

Sistemas de conversação são os famosos chatbots, que podem ou não implementar um algoritmo de inteligência artificial para simular interações com humanos ou simplesmente ter estruturas de diálogos predefinidas, capazes de estabelecer comunicação básica.

4. Conclusão

Considerações finais

A linguagem é a base do intelecto humano e da comunicação, passando para as próximas gerações informação, regras e processos a serem seguidos a fim de facilitar suas vidas futuras.

O processamento de linguagem natural para inteligências artificiais é uma forma de estabelecer uma ponte entre agentes cognitivos e humanos. Desse modo, os conhecimentos humanos oriundos de gerações podem ser organizados e transmitidos para que sistemas e motores de busca se especializem na recuperação de informação. A ideia é poder até mesmo incorporar tais bases de conhecimento em possíveis soluções futuras automatizadas.

Por meio do processamento de linguagem natural, poderão ser desenvolvidas novas formas de interação humano/máquina que facilitarão processos de negócios e até mesmo treinamentos de novas inteligências artificiais supervisionadas por outras mais experientes.

Lembre-se apenas de que todas as abstrações de outras ciências, como matemática e física, baseiam-se em linguagem. Por isso, temos poder ilimitado quando aprendemos a ler e escrever, assim como as inteligências artificiais.

Podcast

Ouça agora conceitos de inteligência artificial, processamento de linguagem natural, tokenização entre outros assuntos abordados.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Confira agora as indicações que separamos especialmente para você!

Aprenda sobre desenvolvimento de Chatbots com a biblioteca Chatterbot acessando o site do Python Package Index.

Saiba mais sobre o uso de PLN na detecção de fake news lendo o artigo **Can machines learn to detect fake news? A survey focused on social media**, de Fernando Cardoso Durier da Silva, Rafael Vieira, Ana Cristina Garcia.

Referências

BABBAR, P. *et al*. **Connectionist model in Artificial Intelligence**. International Journal of Applied Engineering Research, v. 13, n. 7, p. 5154-5159, 2018.

BARBOSA, J. L. N. *et al*. **Introdução ao processamento de linguagem natural usando Python**. *In*: VERAS, R. M. *e t al*. (org.). Anais/III Escola Regional de Informática do Piauí: livro texto dos artigos e minicursos. Picos: Sociedade Brasileira de Computação, 2017.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. 3. ed. São Paulo: GEN LTC, 2013.

SILVA, F. C. D.; GARCIA, A. C. B. **Fake news and sarcasm, what is the limit of a critic and what is intentionally fake?** *In*: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, XV., 2019, Rio de Janeiro. Anais [...]. Rio de Janeiro: SBC, 2019. p. 58-61.

SILVA, F. C. D.; VIEIRA, R.; GARCIA, A. C. B. **Can machines learn to detect fake news? A survey focused on social media**. *In*: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 52., 2019, Wailea. Proceedings [...]. Wailea: HICSS, 2019. p. 2763-2770.

SILVA, F. C. D. **Judice verum**: a methodology for automatically classify fake, sarcastic and true portuguese news. Dissertação (Mestrado em Informática) – Centro de Ciências Exatas e Tecnologia, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

YESHCHENKO, A. *et al*. **Context-aware predictive process monitoring**: the impact of news sentiment. *In*: PANETTO, H. *et al*. (ed.). On the move to meaningful internet systems – OTM 2018 conferences. Cham: Springer, 2018. p. 586-603.