



Frameworks

O que são frameworks

Frameworks são um conjunto de ferramentas que agilizam o desenvolvimento, unindo códigos e funções comuns a diversos projetos de software.

Em resumo, frameworks são ferramentas que possibilitam que códigos e soluções sejam utilizados para resolução de problemas genéricos, adotando o conceito de **reusabilidade**.

Pode-se dizer que Frameworks são conjuntos de bibliotecas ou componentes utilizados para criar uma base em que uma determinada aplicação pode ser construída.

O que são frameworks

De forma geral, é possível observar características que definem os frameworks. São elas:

- Frameworks provêm ferramentas para solução para uma família de problemas que apresentam semelhanças;
- São aplicações “quase completas”, mas com pedaços faltando, que serão desenvolvidas para atender as características específicas de cada problema;
- São compostos por uma grande variedade de parâmetros, garantindo que o desenvolvedor seja capaz de fazer personalizações, de acordo com as necessidades do projeto.

O que são frameworks

Os frameworks utilizam a técnica da Orientação a Objetos, voltada para a reutilização de software, se beneficiando de três características das linguagens de programação orientadas a objetos:

- **Abstração;**
- **Polimorfismo;**
- **Herança.**

O que são frameworks

Com base em sua composição, os frameworks são caracterizados por conter dois tipos de estruturas:

- **Frozen spots** (ou hook points): são serviços e funcionalidades que já foram implementados pelo framework;
- **Host spots**: são funcionalidades, serviços que devem ser implementados pelos desenvolvedores que irão inserir os seus códigos específicos à solução do problema.

O que são frameworks

A utilização de frameworks apresenta diversas vantagens. Dentre elas, podemos citar os seguintes benefícios:

- **Melhora a modularização;**
- **Aumenta a reutilização;**
- **Extensibilidade;**
- **Inversão de controle.**

Referências

RAVINDRA, S.; Como as Redes Neurais Convolucionais realizam o reconhecimento de imagem. 2017. Disponível em: <<https://www.infoq.com/br/articles/redes-neurais-convolucionais>>. Acesso em: 20 Jan. 2020.



Obrigada!

Ana Laurentino



A biblioteca Tensorflow

A biblioteca Tensorflow

A biblioteca Tensorflow é um recurso de código aberto que pode ser utilizado junto a linguagem Python.

O seu objetivo é contribuir com o desenvolvimento da computação numérica de forma associada ao aprendizado de máquina, fazendo com que ela ocorra de forma mais rápida e simples.

A biblioteca Tensorflow

O Tensorflow foi criado por uma das empresas do grupo Google, a Google Brain, e consegue trabalhar com dados em larga escala, por isso, apoia as técnicas de aprendizado de máquina.

Dentro da biblioteca Tensorflow estão vários algoritmos e modelos de aprendizado de máquina e aprendizado profundo.

A interface da biblioteca é própria para o Python, mas a sua execução interna trabalha com o C++ e uma velocidade de compilação bastante rápida.

O que pode ser feito com a Tensorflow?

Utilizando a biblioteca Tensorflow é possível criar boas aplicações para o reconhecimento de textos manuscritos, reconhecimento de imagens, reconhecimentos de caracteres e processamento de linguagens naturais.

Com os modelos utilizados para treinamento é possível estimar como eles se comportariam trabalhando em escala.

O funcionamento da Tensorflow

A biblioteca Tensorflow possui recursos gráficos para que as transformações dos dados possam ser trabalhadas de forma gráfica, ao invés de serem estabelecidas somente via linhas de códigos.

Em cada nó gráfico deve ser indicada uma operação matemática e as conexões entre os nós é representada por uma matriz ou tensor multidimensional.

O funcionamento da Tensorflow

Todos os recursos da biblioteca estão acessíveis através da linguagem Python que é bastante utilizada e conhecida.

Dentro do Python, utilizando a Tensorflow, os nós e tensores possuem objetos específicos que facilitam a manipulação através da linguagem.

O funcionamento da Tensorflow

Ao realizar operações matemáticas reais, elas não serão executadas no Python porque são escritas dentro do Tensorflow na linguagem C++.

As operações são escritas como binários C++, o que faz com que a sua execução aconteça de forma muito mais rápida.

O papel do Python é intermediar a comunicação entre a aplicação e os recursos da biblioteca, redirecionando os dados entre as partes.

Onde o Tensorflow pode ser aplicado?

Uma das grandes vantagens do Tensorflow é a possibilidade dele ser utilizado em diferentes ambientes:

- máquina local;
- cluster na nuvem;
- dispositivos Android;
- dispositivos IOS;
- unidade de processamento personalizada Tensorflow.

Facilidades do Tensorflow

A maior facilidade oferecida pelo Tensorflow é a abstração, através dela o programador não precisa implementar grandes e complexos algoritmos, podendo apenas parametrizá-los e utilizá-los.

O programador se preocupa em como utilizar os dados de uma abordagem em uma outra tarefa, como exibi-los, mas não em como criar abordagens.

Facilidades do Tensorflow

O Tensorflow possui recursos importantes para que o programador possa debugar as suas implementações. Com isso o profissional pode entender mais sobre abordagem e ações que estão sendo executadas sobre os dados.

É possível acompanhar a execução de cada passo de forma gráfica, através do modo ávido, sem a necessidade de se avaliar tudo de uma vez.

Facilidades do Tensorflow

A observação gráfica do funcionamento dos algoritmos é realizada através do recurso chamado de TensorBoard.

Por ser um projeto fomentado da Google, o Tensorflow tem se desenvolvido de forma rápida e propiciou a construção de uma unidade de processamento específica e mais rápida.



Obrigada!

Ana Laurentino



Obrigada!

Ana Laurentino

A biblioteca Keras

A biblioteca Keras oferece um conjunto de recursos em alto nível para a construção de redes neurais para aprendizado profundo.

Essa biblioteca foi implementada em Python e pode ser utilizada até sob a Tensorflow.

Dentro da biblioteca estão disponíveis redes dos tipos: recorrentes e convolucionais.

Motivos para utilizar a Keras

Os motivos para utilização da biblioteca Keras:

- prioriza a experiência do desenvolvedor;
- tem grande adoção pela comunidade de pesquisa e pela indústria;
- torna fácil transformar modelos em produtos;
- suporta múltiplos *backend engines* e não te prende em um só ecossistema;
- suporta a realização de treinamento distribuído.

Motivos para utilizar a Keras

Somados aos aspectos estruturais, existem outros que também reforçam a opção pelo keras, como:

- a biblioteca é amplamente utilizada e, por isso, podem ser encontrados muitos materiais para facilitar a vida de quem quer utilizá-la;
- os recursos da biblioteca estão dispostos de forma simples e intuitiva, sendo assim, a biblioteca é ideal para quem está começando no aprendizado profundo.

MXNet

A MXNet é uma das principais concorrentes da Keras e vem evoluindo cada vez mais depois de ter se tornado um projeto da Apache Foundation, no ano de 2017.

A biblioteca também possui um código fácil de ser entendido e implementado.

Os modelos de redes neurais são flexíveis e podem ser modificados de acordo com o treinamento que é realizado.

Possui uma boa velocidade de execução.

A biblioteca Keras

MXNet

É uma biblioteca que trabalha com código aberto e gratuito.

Possui um funcionamento bastante semelhante a biblioteca Keras.



Obrigada!

Ana Laurentino



Unyleya
EDUCACIONAL



A biblioteca Pytorch

A biblioteca Pytorch

A biblioteca Pytorch foi desenvolvida utilizando a linguagem Python e baseada no antigo Torch.

Essa biblioteca foi desenvolvida, assim como a Tensorflow, para auxiliar programadores a trabalharem com o aprendizado de máquina.

Através dessa biblioteca o programador não precisa implementar algoritmos conhecidos, apenas trabalhar com as versões disponibilizadas dentro da biblioteca.

A biblioteca Pytorch

O desenvolvimento da biblioteca Pytorch é de responsabilidade da empresa Facebook e seus recursos também são explorados por outras grandes empresas como a Uber.

A principal diferença entre o Pytorch e a Tensorflow é que a primeira trabalha com a definição de grafos ao longo da execução dos algoritmos, na Tensorflow acontece o contrário.

A biblioteca Pytorch

A biblioteca Pytorch

A biblioteca surgiu no ano de 2016 e já conta com muitos usuários, inclusive cientistas.

O grande incentivo para uso da biblioteca Pytorch é o fato dela trabalhar a construção de modelos com redes neurais bastante complexas de forma muito simples.

A Pytorch faz concorrência direta com a biblioteca Tensorflow.

A biblioteca Pytorch

A Pytorch é uma biblioteca criada de forma bastante simples e isso que faz com que ela tenha um desempenho tão bom no processamento de cálculos numéricos.

A biblioteca pode ser utilizada por designers de aprendizado profundo, designers de aprendizado de máquina e designers de sistemas neurais.

Incrementando o Pytorch

A Pytorch pode ser utilizado com outros recursos para acrescentar funcionalidades importantes, como:

- Numpy;
- Scipy;
- Cython.

Recursos do Pytorch

São os principais recursos do Pytorch:

- **Interface simples:** É simples para utilizar API, desta forma, é excepcionalmente fácil de trabalhar e executar como o Python.
- **Natureza Pythonica:** Esta biblioteca, sendo Pythonic, coordena facilmente com a pilha de ciência da informação do Python. Portanto, ele pode usar todas as administrações e funcionalidades oferecidas pela condição do Python.
- **Diagramas computacionais:** Além disso, o PyTorch oferece um excelente estágio que oferece gráficos computacionais dinâmicos. Ao longo destas linhas, você pode transformá-los em tempo de execução. Isso é muito valioso quando você não tem ideia de quanta memória será necessária para fazer uma exibição do sistema neural.



Obrigada!

Ana Laurentino



Unyleya
EDUCACIONAL



A biblioteca Pandas

A criação da biblioteca

- A biblioteca Pandas foi criada para facilitar a realização de análises e visualização de dados;
- Possui estruturas de dados diferentes, a Series e o Dataframe.



Importando a biblioteca

Com o ambiente Python devidamente configurado, a biblioteca Pandas pode ser importada dentro de um código da seguinte forma:

```
import pandas as pd
```

Series

A Series é uma das estruturas oferecidas pela a biblioteca Pandas e sua organização é similar a um array unidimensional ou uma lista de valores.

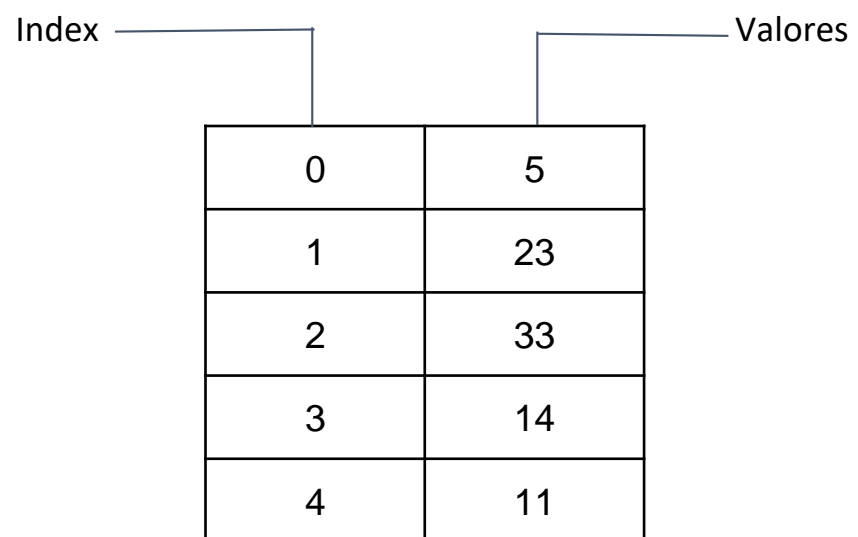
Os objetos do tipo Series possuem um índice para identificar cada valor, o index.

Para criar um objeto do tipo Series contendo quatro elementos, basta:

```
notas = pd.Series([5,23,33,14,11])
```

Como não especificamos rótulos para os valores inseridos, a biblioteca os atribui de forma automática, indo de 0 a 4.

Series



The diagram illustrates the structure of a Pandas Series. It consists of a table with two columns. The first column is labeled 'Index' and the second column is labeled 'Valores'. The table contains five rows of data.

0	5
1	23
2	33
3	14
4	11

Series

Também é possível criar uma Series determinando uma chave para cada um dos valores que estiverem sendo inseridos:

```
notas = pd.Series([5,23,33,14,11], index=['a', 'b', 'c', 'd', 'e'])
```

Series

Os valores de uma Series podem ser acessados através da referência de seu rótulo. Acessando a nota 'b' teremos o retorno do valor 23.

```
notas = pd.Series([5,23,33,14,11], index=['a', 'b', 'c', 'd', 'e'])  
notas['b']
```

Series

Caso os valores dentro da Series sejam numéricos, é possível acessar a média e o desvio padrão de forma simples.

Média:

```
notas.mean()
```

Desvio padrão:

```
notas.std()
```

DataFrame

Os DataFrame são estruturas bidimensionais e podem comportar diferentes tipos de atributo, formando uma estrutura com múltiplas informações.

```
clientes = pd.DataFrame({'Nomes' : ["Ana", "Camila", "Saulo", "Rafael", "David"],  
                        'Idade'  : [23,35,42,21,25],  
                        'Tamanho' : [36,44,50,48,40]})
```

DataFrame

A estrutura de um DataFrame é semelhante a uma planilha:

	Nome	Idade	Tamanho
0	Ana	23	36
1	Camila	35	44
2	Saulo	42	50
3	Raphael	21	48
4	David	25	40

A biblioteca Pandas

DataFrame

É possível verificar todas as informações sobre a estrutura de um objeto do tipo DataFrame:

```
clientes.dtypes
```

DataFrame

Também podem ser verificadas apenas as colunas presentes no objeto DataFrame e acessados todos os valores de uma coluna, respectivamente.

```
clientes.columns  
clientes["Nomes"]
```

DataFrame

É possível ordenar os dados de um DataFrame por alguma coluna específica. A seguir é apresentada a ordenação do objeto pelo nome e depois pela idade.

```
clientes.sort_values(by="Nomes")  
clientes.sort_values(by="Idade")
```


DataFrame

No caso do DataFrame, se o objetivo for acessar um valor a partir do index é necessário utilizar uma função específica, chamada *loc*.

```
clientes.loc[2]
```

DataFrame

Em um objeto DataFrame é possível fazer uma filtragem direta nos dados retornado, informando uma determinada condição. No exemplo a seguir é realizado um filtro nos dados para retornar apenas os clientes que tenha idade inferior a 40 anos.

```
clientes[clientes["Idade"]<40]
```

Leitura de dados

A biblioteca dispõe de algumas funcionalidades para a leitura de arquivos, uma funcionalidade muito importante para quem vai trabalhar com dados que serão importados de dentro de arquivos. A seguir são apresentadas as funções para leitura de arquivos, respectivamente, do tipo csv, xlsx e html.

- `pd.read_csv`
- `pd.read_xlsx`
- `pd.read_html`

Selecionar registros não NaN

Em diversas situações pode ser necessário selecionar os registros de um DataFrame que não contenham atributos nulos.

```
clientes.dropna()
```

Preenchendo atributos NaN

A biblioteca oferece um recurso importante para preencher todos os registros NaN com um determinado valor. Isso pode ser feito de forma simples através da função *fillna*.

```
clientes.fillna()
```

Verificando valores NaN

Em muitas situações é interessante verificar se no Dataframe existem valores NaN ou não. O comando a seguir irá retornar True ou False para cada um dos atributos dos elementos do DataFrame.

```
clientes.isna()
```



Obrigada!

Ana Laurentino



A biblioteca Numpy

A biblioteca Numpy

Essa é uma biblioteca que apoia muito o trabalho com dados, especialmente para a realização de operações matemáticas sobre arrays.

Seus recursos são projetados para permitir a realização de cálculos numéricos de uma forma mais simples e direta.

Contém muitas funcionalidades para a manipulação de imagens.

ndarray

O array do Numpy tem características e funcionalidades diferentes, sendo especialmente chamado de *ndarray*.

Utiliza um esquema diferente para mapeamento dos dados do ndarray na memória. Essa indexação é responsável por fazer o manuseio desse tipo de objeto ser executado de forma rápida, mesmo quando possui uma grande quantidade de dados.

A biblioteca Numpy

ndarray

A criação de um ndarray é bastante simples e pode ser realizada logo depois da biblioteca ter sido importada.

```
import numpy as np  
numeros = np.array([6, 7, 2, 1, 9])
```

Verificando a estrutura

A estrutura de ndarray pode ser verificada através da função *shape* que retorna uma tupla com as dimensões do objeto, linhas e colunas. Para o objeto “numeros” apresentado anteriormente o retorno da função shape seria o seguinte:

```
numeros.shape  
(5,)
```

Array com duas dimensões

Muitas ocasiões necessitarão de arrays com múltiplas dimensões para representar determinadas informações. A criação de um array com múltiplas dimensões é muito simples:

```
numeros = np.array([[6, 7, 2, 1, 9], [6, 7, 2, 1, 9]])
```

Array com duas dimensões

Se a função *shape* fosse utilizada sobre esse novo array, o resultado seria diferente. A tupla resultante está indicado que o array possui duas dimensões com 5 valores cada.

```
numeros = np.array([[6, 7, 2, 1, 9], [6, 7, 2, 1, 9]])  
numeros.shape  
(2, 5)
```

Acessando o *ndarray*

A indexação do *ndarray* começa da posição zero, sendo possível acessar os valores do objeto através de seus índices.

```
numeros[1]
```

Alternado elemento ndarray

A alteração de um elemento dentro de *ndarray* é feita de forma simples, basta referenciar o índice que se deseja modificar e fazer a atribuição direta do valor.

```
numeros[1] = 7
```


Criando ndarray vazio

Um array vazio pode se útil em várias situações. Com os recursos do numpy isso pode ser feito de forma fácil. A função `empty` é adequada para essa situação.

```
numeros = np.empty([3,2], dtype = int)
```

Criando ndarray com zeros

A criação de um vetor com valores 0 pode ser interessante em muitas situações dentro da programação. Para isso, deve-se utilizar uma função específica, a *zeros*. No exemplo será criado *ndarray* com 5 zeros.

```
numeros = np.zeros(5)
```

Criando ndarray com valores aleatórios

A função *random* pode ser utilizada para a geração de valores aleatórios, podendo os valores ser diretamente alocados em *ndarray*.

```
numeros = np.random.random(5)
```

Manipulando ndarray

São muitas as operações matemáticas que podem ser executadas em um *ndarray* e algumas delas funcionam muito diferente do que acontece em grande parte das linguagens de programação.

Vejamos que a operação de adição de *ndarrays* pode ser executada de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])  
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])  
numeros = numeros1 + numeros2
```

```
Adição =  
[[ 8.  10.]  
 [13. 17.]]
```

Manipulando ndarray

Vejamos que a operação de subtração de *ndarrays* pode ser executada de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])  
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])  
numeros = numeros1 - numeros2
```

Subtração =

```
[[ -4.  -4.]
```

```
[7. -1.]]
```

Manipulando ndarray

Vejamos que a operação de multiplicação por elementos de *ndarrays* pode ser executada de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])  
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])  
numeros = numeros1 * numeros2
```

```
Multiplicação =  
[[ 12.  21.]  
 [30. 72.]]
```

Manipulando ndarray

Vejamos que a operação de divisão de *ndarrays* pode ser executada de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])  
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])  
numeros = numeros1 / numeros2
```

```
Divisão =  
[[ 0.3333  0.4285]  
 [3.3333 0.8888]]
```



Obrigada!

Ana Laurentino