Documents structurés

INALCO

Outils TAL

CHEN Xinlei
YI Keming
ZHANG Lidan
CONG Jinyu

Sommaire

JIEBA : bégayer

pour la tokénisation en mandarin

```
pip install jieba
import jieba
```

```
seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 默认模式
```

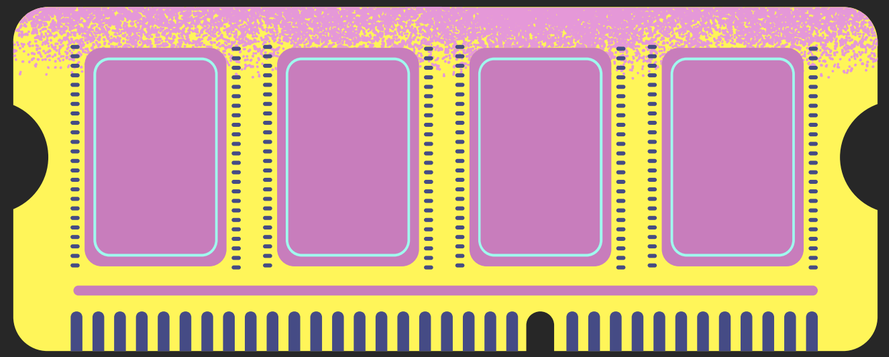
```
[Full Mode]: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

[Accurate Mode]: 我/ 来到/ 北京/ 清华大学
```

Limites :

1. les mots polysémiques
2. des néologismes

Avantages : dictionnaires personnalisées

```
jieba.load_userdict(file_name)
```



HanLP

Han Language Processing

1. Qu'est-ce que HanLP ?

- Une bibliothèque NLP principalement conçue pour le chinois.
- Supporte 104 langues

2. Utilisation de corpus ouverts :

- Universal Dependencies
- OntoNotes

3. Installation:

- pip install hanlp_restful
- pip install hanlp

1. Tokenisation

2. Reconnaissance des entités nommées (NER)

3. Étiquetage des parties du discours (POS tagging)

4. Analyse syntaxique des dépendances

5. Extraction de résumé

6. Extraction de mots-clés

7. Correction de texte

8. Classification de texte

1. Tokenisation

```
HanLP.tokenize('我喜欢吃草莓冰淇淋。')
```

✓ 2.0s

```
[['我', '喜欢', '吃', '草莓', '冰淇淋', '。']]
```

```
HanLP.tokenize('我喜欢吃草莓冰淇淋。', coarse=True)
```

✓ 1.2s

```
[['我', '喜欢', '吃', '草莓冰淇淋', '。']]
```

HanLP

Han Language Processing

Avantages :

- Jeux de données plus complets et riches.
- Standards d’analyse syntaxique plus avancés.
- Méthodes de division des données plus raisonnables.
- Transparence et progrès technologique grâce à l’open source.

HanLP

Han Language Processing

Inconvénients :

- Langues autres que le chinois : fonctionnalités basiques, précision et performances limitées.
- Limitation des appels : 2 appels/minute pour les utilisateurs anonymes.

BERT

BidirectionalEncoder
Representations
from Transformers

Qu’est-ce que BERT

- Un modèle de langage pré-entraîné d’ architecture Transformer bidirectionnelle
- BERT capture les relations contextuelles des mots dans les deux directions (avant et arrière), ce qui améliore considérablement la compréhension du langage.

BERT

BidirectionalEncoder
Representations
from Transformers

Fonctions :

- Classification de texte
- Reconnaissance des entités nommées (NER)
- Génération de texte et complétion de phrases
- Traduction automatique
- Désambiguïsation du sens des mots

BERT

BidirectionalEncoder
Representations
from Transformers

Avantages :

- Compréhension contextuelle bidirectionnelle : améliore la compréhension du sens global du texte
- Adaptabilité à plusieurs tâches : la classification ou les systèmes de questions-réponses
- Utilise une architecture basée sur plusieurs couches de Transformeurs, permettant d’obtenir des représentations sémantiques complexes et adaptées à des tâches NLP variées.

BERT

BidirectionalEncoder
Representations
from Transformers

```python
# 自注意力机制的简单PyTorch代码示例
import torch.nn.functional as F

class SelfAttention(nn.Module):
    def __init__(self, embed_size, heads):
        super(SelfAttention, self).__init__()
        self.embed_size = embed_size
        self.heads = heads
        self.head_dim = embed_size // heads

        assert (
            self.head_dim * heads == embed_size
        ), "Embedding size needs to be divisible by heads"

        self.values = nn.Linear(self.head_dim, self.head_dim, bias=False)
        self.keys = nn.Linear(self.head_dim, self.head_dim, bias=False)
        self.queries = nn.Linear(self.head_dim, self.head_dim, bias=False)
        self.fc_out = nn.Linear(heads * self.head_dim, embed_size)

    def forward(self, values, keys, queries, mask):
        N = queries.shape[0]
        value_len, key_len, query_len = values.shape[1], keys.shape[1], queries.shape[1]

        # Split the embedding into self.head different pieces
        values = values.reshape(N, value_len, self.heads, self.head_dim)
        keys = keys.reshape(N, key_len, self.heads, self.head_dim)
        queries = queries.reshape(N, query_len, self.heads, self.head_dim)

        values = self.values(values)
        keys = self.keys(keys)
        queries = self.queries(queries)

        # Scaled dot-product attention
        attention = torch.einsum("nqhd,nkhd->nhqk", [queries, keys])
        if mask is not None:
            attention = attention.masked_fill(mask == 0, float("-1e20"))

        attention = torch.nn.functional.softmax(attention, dim=3)

        out = torch.einsum("nhql,nlhd->nqhd", [attention, values]).reshape(
            N, query_len, self.heads * self.head_dim
        )

        out = self.fc_out(out)
        return out
```

BERT

BidirectionalEncoder
Representations
from Transformers

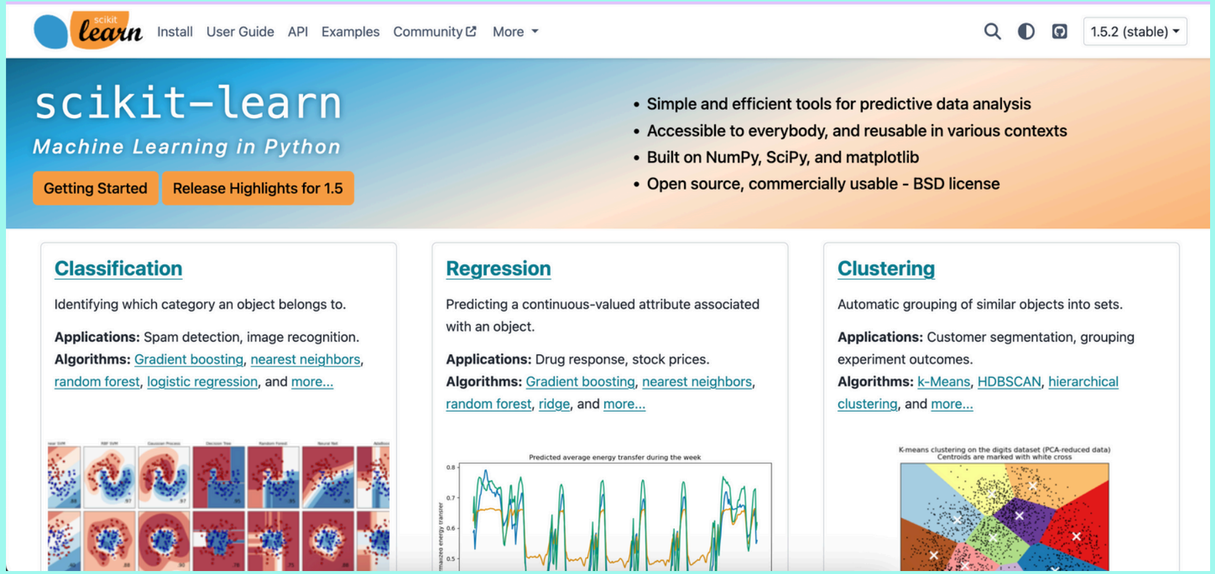
Limitations :

- BERT est un modèle très volumineux, cela nécessite des ressources de calcul importantes.
- En raison de son architecture bidirectionnelle et de sa complexité, BERT est plus lent en inférence par rapport aux modèles unidirectionnels comme GPT.
- BERT est un modèle basé sur un encodeur, conçu principalement pour comprendre le texte, mais moins efficace pour générer du texte.

Scikit-Learn

Qu’est-ce que c’est?

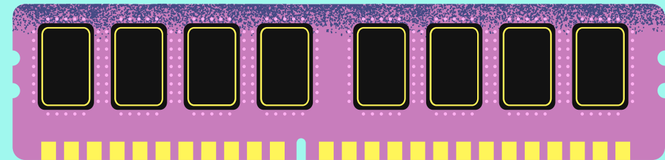
Scikit-learn est une bibliothèque open-source en Python dédiée au machine learning. Elle fournit des outils simples et efficaces pour des tâches de modélisation prédictive.



Langage de programmation

Python

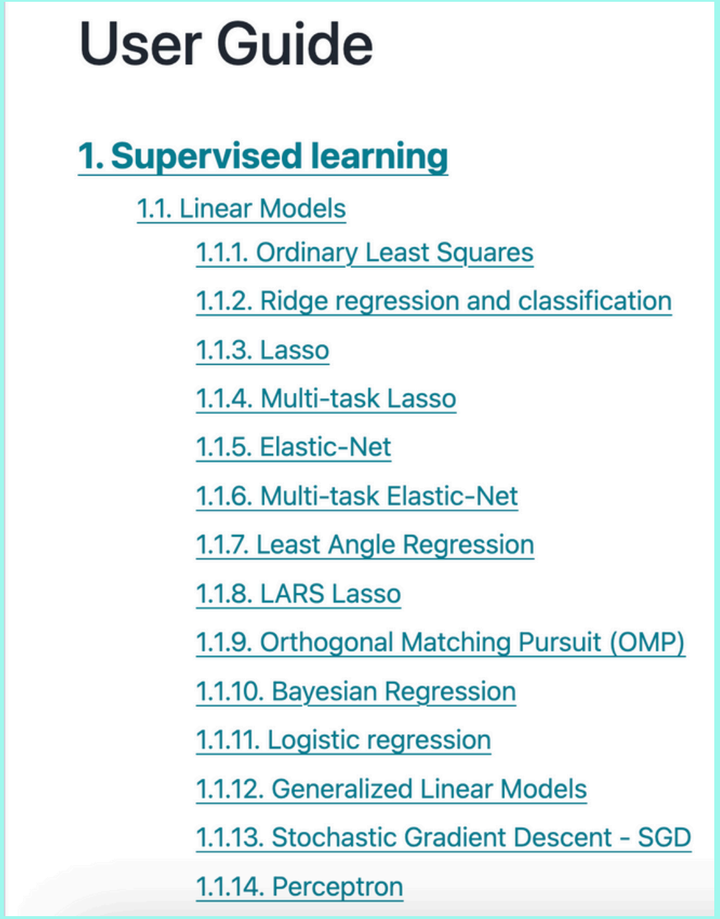
https://scikit-learn.org/stable/



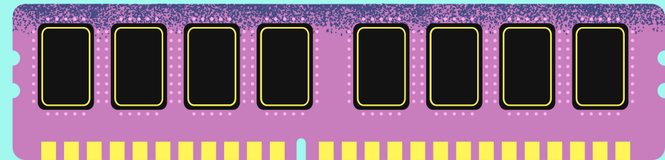
Scikit-Learn

Fonctionnalité 1

Il propose une large gamme d'algorithmes de machine learning supervisés et non supervisés, tels que la régression, la classification (SVM, Random Forest), le clustering (k-means), et bien d’autres...



https://scikit-learn.org/stable/user_guide.html



Scikit-Learn

Fonctionnalité 2

Il fournit des outils pour évaluer la performance des modèles avec des métriques comme l'accuracy, le F1-score.

Scoring	Function
Classification	
'accuracy'	`metrics.accuracy_score`
'balanced_accuracy'	`metrics.balanced_accuracy_score`
'top_k_accuracy'	`metrics.top_k_accuracy_score`

Scikit-Learn

Fonctionnalité 3

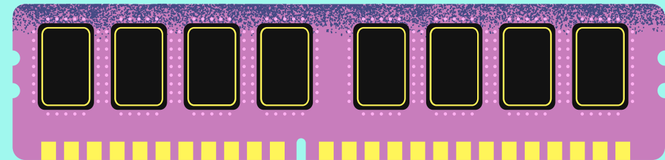
Scikit-learn propose aussi une API simple pour créer des visualisations liées au machine learning.

5. Visualizations

Scikit-learn defines a simple API for creating visualizations for machine learning. The key feature of this API is to allow for quick plotting and visual adjustments without recalculation. We provide `Display` classes that expose two methods for creating plots: `from_estimator` and `from_predictions`. The `from_estimator` method will take a fitted estimator and some data (`X` and `y`) and create a `Display` object. Sometimes, we would like to only compute the predictions once and one should use `from_predictions` instead. In the following example, we plot a ROC curve for a fitted support vector machine:

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import RocCurveDisplay
from sklearn.datasets import load_wine

X, y = load_wine(return_X_y=True)
y = y == 2  # make binary
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
svc = SVC(random_state=42)
svc.fit(X_train, y_train)

svc_disp = RocCurveDisplay.from_estimator(svc, X_test, y_test)
```

https://scikit-learn.org/stable/visualizations.html



Scikit-Learn

Installation simple :

Scikit-learn peut être installé via des gestionnaires de paquets Python comme pip ou conda.

pip install -U scikit-learn

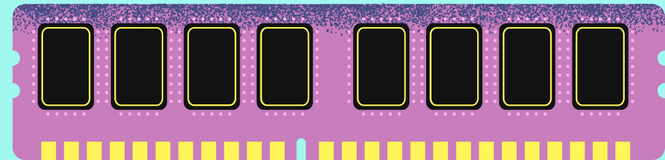
Now create a virtual environment (venv) and install scikit-learn. Note that the virtual environment is optiona
recommended, in order to avoid potential conflicts with other packages.

```
PS C:\> python -m venv sklearn-env
PS C:\> sklearn-env\Scripts\activate  # activate
PS C:\> pip install -U scikit-learn
```

In order to check your installation, you can use:

```
PS C:\> python -m pip show scikit-learn  # show scikit-learn version and location
PS C:\> python -m pip freeze             # show all installed packages in the environment
PS C:\> python -c "import sklearn; sklearn.show_versions()"
```

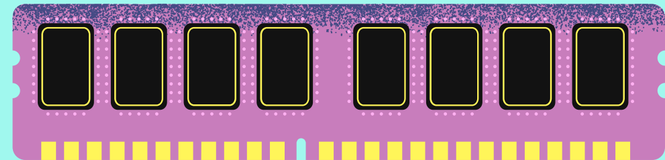
https://scikit-learn.org/stable/



Scikit-Learn

Avantage

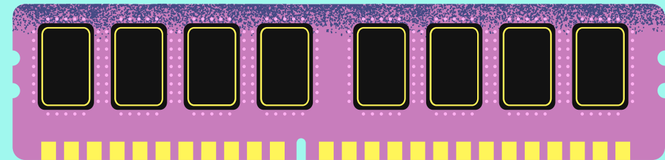
- Facilité d'utilisation
- Large communauté et documentation
- Intégration avec d'autres outils(Numpy, pandas...)
- Performant pour des projets à petite et moyenne échelle.



Scikit-Learn

Limites

- Pas optimisé pour les grandes données
- Pas de deep learning



Merci de votre attention :)