search_rank

December 16, 2020

1 Ranking et search engine

C'est un petit exemple de ranking avec un très petit jeu de données, trop petit pour que le modèle soit performant, mais le code peut être réutilisé pour des exemples de taille raisonnable. C'est à dire probablement pas pour apprendre un moteur de recherche.

[1]:
```
%matplotlib inline
```

[2]:
```
from papierstat.datasets import load_search_engine_dataset
X, y, qid = load_search_engine_dataset()
X[:5,:6].todense()
```

[2]:
```
matrix([[3., 3., 0., 0., 3., 1.],
        [3., 0., 3., 0., 3., 1.],
        [3., 0., 2., 0., 3., 1.],
        [3., 0., 3., 0., 3., 1.],
        [3., 0., 3., 0., 3., 1.]])
```

[3]:
```
X.shape
```

[3]:
```
(582, 136)
```

Le tableau `qid` contient l'identifiant de la requête, toutes les lignes associées à un identifiant correspondent à des résultats associés à cette requête. Dans ce jeu, il y a 7 requêtes distinctes.

[4]:
```
set(qid)
```

[4]:
```
{1, 16, 31, 46, 61, 76, 91}
```

On peut essayer d'abord XGBoost. Ce petit jeu de données est aussi disponible sur github/papierstat/datasets/data.

[5]:
```
X_train, y_train, qid_train = load_search_engine_dataset()
```

[6]:
```
import pandas
df = pandas.DataFrame(qid_train)
df['c'] = 1
df.columns = ['qid', 'c']
gr_train = df.groupby('qid').count()
gr_train
```

[6]:

qid	c

```
1     86
16   106
31    92
46   120
61    59
76    45
91    74
```

[7]:
```
from xgboost import DMatrix
dtrain = DMatrix(data=X_train, label=y_train)
dtrain.set_group(gr_train.values)
```

[8]:
```
from xgboost import XGBRegressor, train
rk = train(params={'objective': 'rank:ndcg'}, dtrain=dtrain, num_boost_round=10)
```

[9]:
```
X_test, y_test, qid_test = load_search_engine_dataset(False)
```

[10]:
```
import pandas
df = pandas.DataFrame(qid_test)
df['c'] = 1
df.columns = ['qid', 'c']
gr_test = df.groupby('qid').count()
dtest = DMatrix(data=X_test)
dtest.set_group(gr_test.values)
```

[11]:
```
pred = rk.predict(dtest)
```

On peut calculer l'erreur au carré.

[12]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, pred)
```

[12]:
```
0.9505686427514932
```

Mais cela n'est valable que si le score a un sens, ce qui est le cas ici. Si ce n'est pas le cas, il est possible d'évaluer les résultats avec la corrélation des rangs des résultats (coefficient de Kendall). Le module lightgbm est une autre option.

[13]:
```
from lightgbm import LGBMRanker

model = LGBMRanker()
model.fit(X_train, y_train, group=gr_train.values.flatten())
```

[13]:
```
LGBMRanker()
```

[14]:
```
Epred = model.predict(X_test.toarray())
Epred[:5]
```

[14]:
```
array([-5.23613848, -1.07701321, -4.46995424, -3.28967461, -1.87554731])
```

Autre option lightfm (article : Learning to Rank Sketchfab Models with LightFM). scikit-learn ne propose pas de modèle de ranking, il faut implémenter soi-même la transformation des données.

[15]:
```
```