Predict tags on job skills with linear models

In this notebook we are going to predict tags for job skills. We are going to use TanitJobs + all the text taged "skill" from the previous model

Libraries

- Numpy - a package for scientific computing.
- Pandas - a library providing high-performance, easy-to-use data structures and data analysis tools for the Python
- Scikit-learn - a tool for data mining and data analysis.
- NLTK - a platform to work with natural language.

Data

Train and Validation Data

In [1]:

```
from ast import literal_eval
import pandas as pd
import numpy as np
```

In [2]:

```
df = pd.read_csv("./Tanitjobs (1).csv", sep=',')
df.dropna(inplace = True)

df['tag'] = df['tag'].apply(literal_eval)
```

In [3]:

```
df
```

Out[3]:

	text	tag
0	Skills and Qualifications Bachelor's degree i...	[Technologie de l'information , IT , Computer...
1	Profile: You have completed your technical stu...	[Développeur , Web , Full Stack , PHP , HTML...
2	Technical Skills: SQL Server / SQL, basic Sync...	[Responsable Applicatif , SQL , C , .NET]
3	In our R & D team, you will participate in the...	[Ingénieur , Développement , C++]
4	Job requirements you serious, motivated, punct...	[Formateur , Développement , Web]
...	...	...
185	- Minimum experience 3 years- Very good comman...	[Administrateur Systèmes et Réseaux , Réseaux ...
186	Your profile to complete this mission: Higher ...	[Web/Chef de projet/Suivi stratégique]
187	Deriving (e) of a higher training in communica...	[Community manager rédacteur de contenu]
188	Ability to work in a team. Degree in Computer...	[php , css , jquery , js , reactjs , redux , h...
189	Diplomas required business school degree, engi...	[Marketing , commerce , analyst , statistiq...

190 rows × 2 columns

train_test_split

In [4]:

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(df.text, df.tag, random_state
= 0)
```

In [5]:

```
X_train = X_train.values
X_val = X_val.values
y_train = y_train.values
y_val = y_val.values
```

Text preprocessing

In [6]:

```
from ast import literal_eval
import pandas as pd
import numpy as np
```

In [7]:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/fairouz/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [8]:

```
import re
```

In [9]:

```
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def text_prepare(text):

    text = text.lower()
    text = re.sub(REPLACE_BY_SPACE_RE, " ", text)
    text = re.sub(BAD_SYMBOLS_RE, "", text)
    text = " ".join([word for word in re.split(" ", text) if (not word in STOPWO
RDS and not word == "")])
    return text
```

Now we preprocess the text using function *text_prepare*:

In [10]:

```
X_train = [text_prepare(x) for x in X_train]
X_val = [text_prepare(x) for x in X_val]
```

In [11]:

```
X_train[:2]
```

Out[11]:

```
['strong analytical skills independent rigorous analysis presentatio
ns analyze sensitive telecom costs present customers analysis report
ing errors tolerated master fine excel crossing data multiple files
manipulating text files links databases excel formulas analyzes pivo
t tables graphics editing macros know least one scripting language m
anipulating data csv xls bdd vba python powershell programming langu
age net windev learn apply automate future treatment independently k
now sql language know create join select queries learn know edit rep
ort powerpoint speak write fluently french know document technical e
nglish training new analysis tools generally provided english assets
knowledge hfsql plus appreciated good oral editorial level french pl
us appreciated may need frequently interact clients email french pre
sent reports monthly meetings theoretical knowledge fixed telephony
mobile telephony networks internet mpls telecom operators possibly b
illing plus appreciated handle billing data french telecom operators
orange sfr bouygues subscriptions invoices consumption development i
nternet access service bi tools knowledge plus appreciated interest
new languages computer skills plus',
 'tray bts mastered computer tools without experience spot smiling d
ynamic presentable rigor']
```

For each tag and for each word we calculate how many times they occur in the train corpus.

In [12]:

```
# Dictionary of all tags from train corpus with their counts.
tags_counts = {}
# Dictionary of all words from train corpus with their counts.
words_counts = {}

from collections import Counter

print("counting words..")
words = []
for text in X_train:
    words = words + text.split()

print("counting tags..")
tags = []
for tag in y_train:
    tags = tags + tag

print("done")
words_counts = Counter(words)
tags_counts=Counter(tags)
```

```
counting words..
counting tags..
done
```

In [13]:

```
len(words_counts)
```

Out[13]:

```
1613
```

Bag Of Words

In [16]:

```
DICT_SIZE = 1500
WORDS_TO_INDEX = {word:i for i,word in enumerate([word for word,_ in sorted(word
s_counts.items(), key=lambda x: x[1], reverse=True)[:DICT_SIZE]])}
INDEX_TO_WORDS = dict(enumerate(list(WORDS_TO_INDEX)))
ALL_WORDS = WORDS_TO_INDEX.keys()

def my_bag_of_words(text, words_to_index, dict_size):

    result_vector = np.zeros(dict_size)
    for word,i in words_to_index.items():
        if word in text.split():
            result_vector[i] = result_vector[i] +1

    return result_vector
```

Now we apply the implemented function to all samples:

In [17]:

```
from scipy import sparse as sp_sparse
```

In [18]:

```
X_train_mybag = sp_sparse.vstack([sp_sparse.csr_matrix(my_bag_of_words(text, WOR
DS_TO_INDEX, DICT_SIZE)) for text in X_train])
X_val_mybag = sp_sparse.vstack([sp_sparse.csr_matrix(my_bag_of_words(text, WORDS
_TO_INDEX, DICT_SIZE)) for text in X_val])
# X_train_mybag.toarray() pour "decompresser" la matrice
print('X_train shape ', X_train_mybag.shape)
print('X_val shape ', X_val_mybag.shape)
```

```
X_train shape  (142, 1500)
X_val shape  (48, 1500)
```

We transform the data to sparse representation, to store the useful information efficiently.

Sklearn algorithms can work only with **csr** matrix, so we will use this one.

Training

Each example can have multiple tags. To deal with such kind of prediction, we need to transform labels in a binary form and the prediction will be a mask of 0s and 1s. For this purpose it is convenient to use MultiLabelBinarizer from *sklearn*.

In [19]:

```
from sklearn.preprocessing import MultiLabelBinarizer
```

In [20]:

```
mlb = MultiLabelBinarizer(classes=sorted(tags_counts.keys()))
y_train = mlb.fit_transform(y_train)
y_val = mlb.fit_transform(y_val)
```

```
/home/fairouz/.local/lib/python3.6/site-packages/sklearn/preprocessi
ng/label.py:951: UserWarning: unknown class(es) ['Administrateur Sys
tèmes et Réseaux ', 'Agile  ', 'Computer science ', 'Gestion de Proj
et  ', 'Ingénieur ', 'JIRA ', 'Leps Administrator '] will be ignored
  .format(sorted(unknown, key=str)))
```

In [21]:

```
print(len(mlb.classes_))
print(mlb.classes_[:10])
```

```
302
[' .NET Core ' ' .Net ' ' AGILE ' ' AJAX ' ' ALM ' ' Analyste '
 ' Angular ' ' Avaloq ' ' BOOTSTRAP ' ' Bootstrap 4 ']
```

In [22]:

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression, RidgeClassifier
```

In [23]:

```
def train_classifier_LogisticRegression(X_train, y_train):

    clf = OneVsRestClassifier(LogisticRegression()).fit(X_train, y_train)
    return clf
```

In [24]:

```
def train_classifier_RidgeClassifier(X_train, y_train):

    clf = OneVsRestClassifier(RidgeClassifier()).fit(X_train, y_train)
    return clf
```

Train the classifiers for different data transformations: *bag-of-words* and *tf-idf*.

In [25]:

```
classifier_LogisticRegression = train_classifier_LogisticRegression(X_train_myba
g, y_train)
classifier_RidgeClassifier = train_classifier_RidgeClassifier(X_train_mybag, y_t
rain)
```

```
/home/fairouz/.local/lib/python3.6/site-packages/sklearn/linear_mode
l/logistic.py:432: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Now you can create predictions for the data. You will need two types of predictions: labels and scores.

In [26]:

```
y_val_predicted_labels_LogisticRegression = classifier_LogisticRegression.predict(X_val_mybag)
y_val_predicted_scores_LogisticRegression = classifier_LogisticRegression.decision_function(X_val_mybag)

y_val_predicted_labels_RidgeClassifier = classifier_RidgeClassifier.predict(X_val_mybag)
y_val_predicted_scores_RidgeClassifier = classifier_RidgeClassifier.decision_function(X_val_mybag)
```

Now take a look at how Ridge Classifier works for a few examples:

In [27]:

```
y_val_pred_inversed = mlb.inverse_transform(y_val_predicted_labels_RidgeClassifi
er)
y_val_inversed = mlb.inverse_transform(y_val)
for i in range(3):
    print('Title:\t{}\nTrue labels:\t{}\nPredicted labels:\t{}\n\n'.format(
        X_val[i],
        ','.join(y_val_inversed[i]),
        ','.join(y_val_pred_inversed[i])
    ))
```

```
Title:  deriving e engineering background electronics automation ins
trumentation industrial proven experience 2 5 years minimum position
similairevous control software design automation autocadvous indepth
knowledge schneider siemens design supervisionvous knowledge electro
nic electrical diagrams knowledge supervision software type intouch
win cc ignition plusvous control cost calculation position chiffrage
le requires significant mobility abroad good level english good inte
rpersonal skills rigor organization responsiveness adaptability stre
ngths succeed mission order consider application carefully appreciat
e contact vidal associates consulting search filing application refe
rence vt10180f
True labels:    Automatisme ,Electronique ,Informatique industrielle
,Ingénieur Automatisme ,Instrumentation
Predicted labels:       Automatisme ,Electronique ,Informatique indu
strielle ,Ingénieur Automatisme ,Instrumentation


Title:  bts degree specialized experience technical competence field
computer maintenance efficiency communicate direct work environment
customers suppliers apply contact +216 98248675 send email info medi
ahousetncom
True labels:    Maintenance ,Technicien en Maintenance,infrastructur
e
Predicted labels:       Maintenance ,Technicien en Maintenance,infra
structure


Title:  graduate e business school engineering specialty finance e r
eal sense customer service strong ability work team strong taste fig
ures good analytical synthesis good organizational skills good comma
nd english french pack office freshly graduated e pfe send cv
True labels:    Business Analyst ,Commerce  ,Finance  ,Gestion
Predicted labels:
```

Evaluation

To evaluate the results we will use several classification metrics:

- Accuracy
- F1-score

In [28]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
```

In [29]:

```
def print_evaluation_scores(y_val, predicted):

    accuracy=accuracy_score(y_val, predicted)
    f1_score_macro=f1_score(y_val, predicted, average='macro')
    f1_score_micro=f1_score(y_val, predicted, average='micro')
    f1_score_weighted=f1_score(y_val, predicted, average='weighted')

    print("Accuracy:", accuracy)
    print("F1-score macro:", f1_score_macro)
    print("F1-score micro:", f1_score_micro)
    print("F1-score weighted:", f1_score_weighted)
```

In [31]:

```
print('LogisticRegression')
print_evaluation_scores(y_val, y_val_predicted_labels_LogisticRegression)
print()
print('RidgeClassifier')
print_evaluation_scores(y_val, y_val_predicted_labels_RidgeClassifier)
```

```
LogisticRegression
Accuracy: 0.8333333333333334
F1-score macro: 0.4666666666666667
F1-score micro: 0.9081364829396326
F1-score weighted: 0.8746268656716418

RidgeClassifier
Accuracy: 0.8541666666666666
F1-score macro: 0.477924944812362
F1-score micro: 0.9222797927461139
F1-score weighted: 0.8955223880597015
```