Lab4-Assignment about Named Entity Recognition, Classification and Disambiguation

This notebook describes the assignment of Lab 4 of the text mining course.

Learning goals

- going from linguistic input format to representing it in a feature space
- working with pretrained word embeddings
- train a supervised classifier (SVM)
- evaluate a supervised classifier (SVM)
- perform feature ablation and gain insight into the contribution of various features
- Learn how to evaluate an entity linking system.
- Learn how to run two entity linking systems (AIDA and DBpedia Spotlight).
- Learn how to interpret the system output and the evaluation results.
- Get insight into differences between the two systems.
- Be able to describe differences between the two methods in terms of their results.
- Be able to propose future improvements based on the observed results.
- Get insight into the difficulty of NED and how this depends on specific entity mentions.

The assignment consists of 2 parts:

- Named Entity Recornition and Classificaiton: excersizes 1 & 2
- Named Entity Disambiguation and Linking: excersizes 3 & 4

Credits

This notebook was originally created by [Marten Postma (https://martenpostma.github.io)](https://martenpostma.github.io) and [Filip Ilievski (http://ilievski.nl)](http://ilievski.nl) and dapated by Piek vossen

Named Entity Recognition and Classification

Excercises 2 and 3 focus on Named Entity Recognition and Classification

[Points: 18] Exercise 1 (NERC): Training and evaluating an SVM using CoNLL-2003

[4 point] a) Load the CoNLL-2003 training data using the *ConllCorpusReader* and create for both *train.txt* and *test.txt*:

```
[2 points]  -a list of dictionaries representing the features for each tr
aining instances, e..g,
```
[
{'words': 'EU', 'pos': 'NNP'},
{'words': 'rejects', 'pos': 'VBZ'},
...
]
```

[2 points] -the NERC labels associated with each training instance, e.g.,
dictionaries, e.g.,
```
[
'B-ORG',
'O',
....
]
```
```

In [28]:

```python
from nltk.corpus.reader import ConllCorpusReader
```

In [29]:

```
train = ConllCorpusReader('nerc_datasets/CONLL2003', 'train.txt', ['words', 'po
s', 'ignore', 'chunk'])
training_features = []
training_gold_labels = []

for token, pos, ne_label in train.iob_words():
    a_dict = {
        #features:
        'words': token,
        'pos': pos
    }
    training_features.append(a_dict)
    training_gold_labels.append(ne_label)
print('First 10 elements from the training instances feautures:\n',training_feat
ures[:10])
print()
print('First 10 elements from the training instances NERC labels:\n', training_g
old_labels[:10])
```

```
First 10 elements from the training instances feautures:
 [{'words': 'EU', 'pos': 'NNP'}, {'words': 'rejects', 'pos': 'VB
Z'}, {'words': 'German', 'pos': 'JJ'}, {'words': 'call', 'pos': 'N
N'}, {'words': 'to', 'pos': 'TO'}, {'words': 'boycott', 'pos': 'V
B'}, {'words': 'British', 'pos': 'JJ'}, {'words': 'lamb', 'pos': 'N
N'}, {'words': '.', 'pos': '.'}, {'words': 'Peter', 'pos': 'NNP'}]

First 10 elements from the training instances NERC labels:
 ['B-ORG', 'O', 'B-MISC', 'O', 'O', 'O', 'B-MISC', 'O', 'O', 'B-PE
R']
```

In [30]:

```
### Adapt the path to point to the NERC_datasets folder on your local machine
test = ConllCorpusReader('nerc_datasets/CONLL2003', 'test.txt', ['words', 'pos',
'ignore', 'chunk'])

test_features = []
test_gold_labels = []
for token, pos, ne_label in test.iob_words():
    a_dict = {
        #features:
        'words':token,
        'pos': pos
    }
    test_features.append(a_dict)
    test_gold_labels.append(ne_label)
print('First 10 elements from the test instances feautures:\n',test_features[:10
])
print()
print('First 10 elements from the test instances NERC labels:\n', test_gold_labe
ls[:10])
```

```
First 10 elements from the test instances feautures:
 [{'words': 'SOCCER', 'pos': 'NN'}, {'words': '-', 'pos': ':'}, {'w
ords': 'JAPAN', 'pos': 'NNP'}, {'words': 'GET', 'pos': 'VB'}, {'wor
ds': 'LUCKY', 'pos': 'NNP'}, {'words': 'WIN', 'pos': 'NNP'}, {'word
s': ',', 'pos': ','}, {'words': 'CHINA', 'pos': 'NNP'}, {'words':
'IN', 'pos': 'IN'}, {'words': 'SURPRISE', 'pos': 'DT'}]

First 10 elements from the test instances NERC labels:
 ['O', 'O', 'B-LOC', 'O', 'O', 'O', 'O', 'B-PER', 'O', 'O']
```

[2 points] b) provide descriptive statistics about the training and test data:

- How many instances are in train and test?
- Provide a frequency distribution of the NERC labels, i.e., how many times does each NERC label occur?
- Discuss to what extent the training and test data is balanced (equal amount of instances for each NERC label) and to what extent the training and test data differ?

Tip: you can use the following `Counter` functionality to generate frequency list of a list:

In [31]:

```
# from collections import Counter
# my_list=[1,2,1,3,2,5]
# Counter(my_list)
import pandas
```

In [32]:

```
print( '\033[1m How many instances are in train and test?\033[0m')
print('There are %d instances in train and %d instances in test.'%(len(training_
features),len(test_features)))
print()

#Provide a frequency distribution of the NERC labels, i.e., how many times does
 each NERC label occur?
df_train = pandas.DataFrame(training_gold_labels)
df_train.columns = ['frequency']
print('\033[1m NERC-label frequency distribution of train: \033[0m  \n', df_trai
n.apply(pandas.value_counts))
print()
df_test = pandas.DataFrame(test_gold_labels)
df_test.columns = ['frequency']
print('\033[1m NERC-label frequency distribution of test: \033[0m \n', df_test.a
pply(pandas.value_counts))
print()
print('\033[1m Balance and differences between test and train : \033[0m \n\
The data is reasonably balanced, in both the train and test data the\
highest frequency is the O NERC label, and the lowest frequency is the I-MISC NE
RC label.\
The only difference is the frequency of the B-LOC NERC label,\
the train data has relatively more instances.')
```

```
 How many instances are in train and test?
There are 203621 instances in train and 46435 instances in test.

 NERC-label frequency distribution of train:
          frequency
O            169578
B-LOC          7140
B-PER          6600
B-ORG          6321
I-PER          4528
I-ORG          3704
B-MISC         3438
I-LOC          1157
I-MISC         1155

 NERC-label frequency distribution of test:
          frequency
O             38323
B-LOC          1668
B-ORG          1661
B-PER          1617
I-PER          1156
I-ORG           835
B-MISC          702
I-LOC           257
I-MISC          216

 Balance and differences between test and train :
The data is reasonably balanced, in both the train and test data th
ehighest frequency is the O NERC label, and the lowest frequency is
the I-MISC NERC label.The only difference is the frequency of the B
-LOC NERC label,the train data has relatively more instances.
```

[2 points] c) Concatenate the train and test features (the list of dictionaries) into one list. Load it using the *DictVectorizer*. Afterwards, split it back to training and test.

Tip: You've concatenated train and test into one list and then you've applied the DictVectorizer. The order of the rows is maintained. You can hence use an index (number of training instances) to split the_array back into train and test. Do NOT use: `from sklearn.model_selection import train_test_split` here.

In [33]:

```
from sklearn.feature_extraction import DictVectorizer
```

In [34]:

```
vec = DictVectorizer()
all_features = training_features + test_features
the_array = vec.fit_transform(all_features).toarray()
new_train = the_array[:203621]
new_test = the_array[203621:]
print(new_train)
print(new_test)
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent ca
ll last)
<ipython-input-34-cf855ea8ce29> in <module>
      1 vec = DictVectorizer()
      2 all_features = training_features + test_features
----> 3 the_array = vec.fit_transform(all_features).toarray()
      4 new_train = the_array[:203621]
      5 new_test = the_array[203621:]

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/compressed.py
 in toarray(self, order, out)
   1022         if out is None and order is None:
   1023             order = self._swap('cf')[0]
-> 1024         out = self._process_toarray_args(order, out)
   1025         if not (out.flags.c_contiguous or out.flags.f_conti
guous):
   1026             raise ValueError('Output array must be C or F c
ontiguous')

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/base.py in _pr
ocess_toarray_args(self, order, out)
   1184             return out
   1185         else:
-> 1186             return np.zeros(self.shape, dtype=self.dtype, o
rder=order)
   1187 
   1188 

MemoryError: Unable to allocate array with shape (250056, 27361) an
d data type float64
```

[4 points] d) Train the SVM using the train features and labels and evaluate on the test data. Provide a classification report (sklearn.metrics.classification_report). The train (*lin_clf.fit*) might take a while. On my computer, it took 1min 53s, which is acceptable. Training models normally takes much longer. If it takes more than 5 minutes, you can use a subset for training. Describe the results:

- Which NERC labels does the classifier perform well on? Why do you think this is the case?
- Which NERC labels does the classifier perform poorly on? Why do you think this is the case?

In [5]:

```
from sklearn import svm
from sklearn.metrics import classification_report
```

In [6]:

```
lin_clf = svm.LinearSVC()
```

In [10]:

```
lin_clf.fit(new_train, training_gold_labels)
predict_label = lin_clf.predict(new_test)
print(classification_report(test_gold_labels, predict_label))
```

```
              precision    recall  f1-score   support

       B-LOC       0.81      0.78      0.79      1668
      B-MISC       0.78      0.66      0.72       702
       B-ORG       0.79      0.52      0.63      1661
       B-PER       0.86      0.44      0.58      1617
       I-LOC       0.62      0.53      0.57       257
      I-MISC       0.57      0.59      0.58       216
       I-ORG       0.70      0.47      0.56       835
       I-PER       0.33      0.87      0.48      1156
           O       0.98      0.98      0.98     38323

    accuracy                           0.92     46435
   macro avg       0.72      0.65      0.65     46435
weighted avg       0.94      0.92      0.92     46435
```

2d) The outside tags perform well, this is probably because there is not much overlap/ no connection between named entities and the locations, verbs, prepositions, etc.. The inside person tags performs poorly, probably because of last names that also accur on their own and are therefor classified wrongly.

[6 points] e) Train a model that uses the embeddings of these words as inputs. Test again on the same data as in 2d. Generate a classification report and compare the results with the classifier you built in 2d.

In [7]:

```
import gensim
```

In [8]:

```
word_embedding_model = gensim.models.KeyedVectors.load_word2vec_format('model/GoogleNews-vectors-negative300.bin', binary=True)
```

In [9]:

```
embedding_train=[]
embedding_gold_labels=[]
embedding_test=[]
embedding_gold_test_labels=[]
for token, pos, ne_label in train.iob_words():

    if token!='' and token!='DOCSTART':
        if token in word_embedding_model:
            vector=word_embedding_model[token]
        else:
            vector=[0]*300
        embedding_train.append(vector)
        embedding_gold_labels.append(ne_label)

for token, pos, ne_label in test.iob_words():

    if token!='' and token!='DOCSTART':
        if token in word_embedding_model:
            vector=word_embedding_model[token]
        else:
            vector=[0]*300
        embedding_test.append(vector)
        embedding_gold_test_labels.append(ne_label)
```

In [16]:

```
lin_clf.fit(embedding_train, embedding_gold_labels)
predict_label = lin_clf.predict(embedding_test)
print(classification_report(embedding_gold_test_labels, predict_label))
```

```
              precision    recall  f1-score   support

       B-LOC       0.76      0.80      0.78      1668
      B-MISC       0.72      0.70      0.71       702
       B-ORG       0.69      0.64      0.66      1661
       B-PER       0.75      0.67      0.71      1617
       I-LOC       0.51      0.42      0.46       257
      I-MISC       0.60      0.54      0.57       216
       I-ORG       0.48      0.33      0.39       835
       I-PER       0.59      0.50      0.54      1156
           O       0.97      0.99      0.98     38323

    accuracy                           0.93     46435
   macro avg       0.68      0.62      0.64     46435
weighted avg       0.92      0.93      0.92     46435
```

2e) Comparison classification reports 2d and 2e: Overall the precision is a bit lower when using the embeddings as input, however, the difference is not great and the accuracy, macro avg and weighted avd are still good. The only exeption is the inside person tag, when using the embeddings of the words as input it performs much better.

[Points: 10] Exercise 2 (NERC): feature inspection using the [Annotated Corpus for Named Entity Recognition (https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus)](https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus)

[6 points] a. Perform the same steps as in the previous exercise. Make sure you end up for both the training part (*df_train*) and the test part (*df_test*) with:

- the features representation using **DictVectorizer**
- the NERC labels in a list

Please note that this is the same setup as in the previous exercise:

- load both train and test using:
  - list of dictionaries for features
  - list of NERC labels
- combine train and test features in a list and represent them using one hot encoding
- train using the training features and NERC labels

In [9]:

```
import pandas
from sklearn.feature_extraction import DictVectorizer
```

In [11]:

```
##### Adapt the path to point to your local copy of NERC_datasets
path = 'nerc_datasets/kaggle/ner_v2.csv'
kaggle_dataset = pandas.read_csv(path, error_bad_lines=False)
```

```
b'Skipping line 281837: expected 25 fields, saw 34\n'
```

In [12]:

```
len(kaggle_dataset)
```

Out[12]:

```
1050795
```

In [13]:

```
df_train = kaggle_dataset[:100000]
df_test = kaggle_dataset[100000:120000]
print(len(df_train), len(df_test))
```

```
100000 20000
```

In [14]:

```
vec = DictVectorizer()

training2_features = []
training2_gold_labels = []
test2_features = []
test2_gold_labels = []
for index, instance in df_train.iterrows():
    a_dict = {
        'id': index,
        'lemma':instance['lemma'],
        'pos':instance['pos'],
        'shape':instance['shape'],
        'word':instance['word'],
        'next-lemma': instance['next-lemma'],
        'next-pos': instance['next-pos'],
        'next-shape': instance['next-shape'],
        'next-word': instance['next-word'],
        'next-next-lemma': instance['next-next-lemma'],
        'next-next-pos': instance['next-next-pos'],
        'next-next-shape': instance['next-next-shape'],
        'next-next-word': instance['next-next-word'],
        'prev-iob':instance['prev-iob'],
        'prev-lemma':instance['prev-lemma'],
        'prev-pos':instance['prev-pos'],
        'prev-shape':instance['prev-shape'],
        'prev-word':instance['prev-word'],
        'prev-prev-iob':instance['prev-prev-iob'],
        'prev-prev-lemma':instance['prev-prev-lemma'],
        'prev-prev-pos':instance['prev-prev-pos'],
        'prev-prev-shape':instance['prev-prev-shape'],
        'prev-prev-word':instance['prev-prev-word'],
        'sentence_idx':instance['sentence_idx']
    }
    training2_features.append(a_dict)
    training2_gold_labels.append(instance['tag'])

for index, instance in df_test.iterrows():
    a_dict = {
        'id': index,
        'lemma':instance['lemma'],
        'pos':instance['pos'],
        'shape':instance['shape'],
        'word':instance['word'],
        'next-lemma': instance['next-lemma'],
        'next-pos': instance['next-pos'],
        'next-shape': instance['next-shape'],
        'next-word': instance['next-word'],
        'next-next-lemma': instance['next-next-lemma'],
        'next-next-pos': instance['next-next-pos'],
        'next-next-shape': instance['next-next-shape'],
        'next-next-word': instance['next-next-word'],
        'prev-iob':instance['prev-iob'],
        'prev-lemma':instance['prev-lemma'],
        'prev-pos':instance['prev-pos'],
        'prev-shape':instance['prev-shape'],
        'prev-word':instance['prev-word'],
        'prev-prev-iob':instance['prev-prev-iob'],
        'prev-prev-lemma':instance['prev-prev-lemma'],
        'prev-prev-pos':instance['prev-prev-pos'],
```

```
        'prev-prev-shape':instance['prev-prev-shape'],
        'prev-prev-word':instance['prev-prev-word'],
        'sentence_idx':instance['sentence_idx']
    }
    test2_features.append(a_dict)
    test2_gold_labels.append(instance['tag'])
```

In [15]:

```
print("\033[1m First  5 training feature: \033[0m", training2_features[:5])
print(" \033[1mand the training NERC labels: \033[0m", training2_gold_labels[:5]
)
print()
print("\033[1mFirst 5 test feature: \033[0m", training2_features[:5])
print(" \033[1mand the test NERC labels:\033[0m", test2_gold_labels[:5] )
```

First 5 training feature: [{'id': 0, 'lemma': 'thousand', 'pos': 'NNS', 'shape': 'capitalized', 'word': 'Thousands', 'next-lemma': 'of', 'next-pos': 'IN', 'next-shape': 'lowercase', 'next-word': 'of', 'next-next-lemma': 'demonstr', 'next-next-pos': 'NNS', 'next-next-shape': 'lowercase', 'next-next-word': 'demonstrators', 'prev-iob': '__START1__', 'prev-lemma': '__start1__', 'prev-pos': '__START1__', 'prev-shape': 'wildcard', 'prev-word': '__START1__', 'prev-prev-iob': '__START2__', 'prev-prev-lemma': '__start2__', 'prev-prev-pos': '__START2__', 'prev-prev-shape': 'wildcard', 'prev-prev-word': '__START2__', 'sentence_idx': 1.0}, {'id': 1, 'lemma': 'of', 'pos': 'IN', 'shape': 'lowercase', 'word': 'of', 'next-lemma': 'demonstr', 'next-pos': 'NNS', 'next-shape': 'lowercase', 'next-word': 'demonstrators', 'next-next-lemma': 'have', 'next-next-pos': 'VBP', 'next-next-shape': 'lowercase', 'next-next-word': 'have', 'prev-iob': 'O', 'prev-lemma': 'thousand', 'prev-pos': 'NNS', 'prev-shape': 'capitalized', 'prev-word': 'Thousands', 'prev-prev-iob': '__START1__', 'prev-prev-lemma': '__start1__', 'prev-prev-pos': '__START1__', 'prev-prev-shape': 'wildcard', 'prev-prev-word': '__START1__', 'sentence_idx': 1.0}, {'id': 2, 'lemma': 'demonstr', 'pos': 'NNS', 'shape': 'lowercase', 'word': 'demonstrators', 'next-lemma': 'have', 'next-pos': 'VBP', 'next-shape': 'lowercase', 'next-word': 'have', 'next-next-lemma': 'march', 'next-next-pos': 'VBN', 'next-next-shape': 'lowercase', 'next-next-word': 'marched', 'prev-iob': 'O', 'prev-lemma': 'of', 'prev-pos': 'IN', 'prev-shape': 'lowercase', 'prev-word': 'of', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'thousand', 'prev-prev-pos': 'NNS', 'prev-prev-shape': 'capitalized', 'prev-prev-word': 'Thousands', 'sentence_idx': 1.0}, {'id': 3, 'lemma': 'have', 'pos': 'VBP', 'shape': 'lowercase', 'word': 'have', 'next-lemma': 'march', 'next-pos': 'VBN', 'next-shape': 'lowercase', 'next-word': 'marched', 'next-next-lemma': 'through', 'next-next-pos': 'IN', 'next-next-shape': 'lowercase', 'next-next-word': 'through', 'prev-iob': 'O', 'prev-lemma': 'demonstr', 'prev-pos': 'NNS', 'prev-shape': 'lowercase', 'prev-word': 'demonstrators', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'of', 'prev-prev-pos': 'IN', 'prev-prev-shape': 'lowercase', 'prev-prev-word': 'of', 'sentence_idx': 1.0}, {'id': 4, 'lemma': 'march', 'pos': 'VBN', 'shape': 'lowercase', 'word': 'marched', 'next-lemma': 'through', 'next-pos': 'IN', 'next-shape': 'lowercase', 'next-word': 'through', 'next-next-lemma': 'london', 'next-next-pos': 'NNP', 'next-next-shape': 'capitalized', 'next-next-word': 'London', 'prev-iob': 'O', 'prev-lemma': 'have', 'prev-pos': 'VBP', 'prev-shape': 'lowercase', 'prev-word': 'have', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'demonstr', 'prev-prev-pos': 'NNS', 'prev-prev-shape': 'lowercase', 'prev-prev-word': 'demonstrators', 'sentence_idx': 1.0}]
and the training NERC labels: ['O', 'O', 'O', 'O', 'O']

First 5 test feature: [{'id': 0, 'lemma': 'thousand', 'pos': 'NNS', 'shape': 'capitalized', 'word': 'Thousands', 'next-lemma': 'of', 'next-pos': 'IN', 'next-shape': 'lowercase', 'next-word': 'of', 'next-next-lemma': 'demonstr', 'next-next-pos': 'NNS', 'next-next-shape': 'lowercase', 'next-next-word': 'demonstrators', 'prev-iob': '__START1__', 'prev-lemma': '__start1__', 'prev-pos': '__START1__', 'prev-shape': 'wildcard', 'prev-word': '__START1__', 'prev-prev-iob': '__START2__', 'prev-prev-lemma': '__start2__', 'prev-prev-pos': '__START2__', 'prev-prev-shape': 'wildcard', 'prev-prev-word': '__START2__', 'sentence_idx': 1.0}, {'id': 1, 'lemma': 'of', 'pos': 'IN', 'shape': 'lowercase', 'word': 'of', 'next-lemma': 'demonstr', 'next-pos': 'NNS', 'next-shape': 'lowercase', 'next-word': 'demonstrators', 'next-next-lemma': 'have', 'next-next-pos': 'VBP', 'next-next-shape': 'lowercase', 'next-next-word': 'have', 'prev-iob': 'O', 'prev-lemma': 'thousand', 'prev-pos': 'NNS', 'prev-shape': 'ca pitalized', 'prev-word': 'Thousands', 'prev-prev-iob': '__START1__', 'prev-prev-lemma': '__start1__', 'prev-prev-pos': '__START1__', 'prev-prev-shape': 'wildcard', 'prev-prev-word': '__START1__', 'sentence_idx': 1.0}, {'id': 2, 'lemma': 'demonstr', 'pos': 'NNS', 'shape': 'lowercase', 'word': 'demonstrators', 'next-lemma': 'have', 'next-pos': 'VBP', 'next-shape': 'lowercase', 'next-word': 'have', 'next-next-lemma': 'march', 'next-next-pos': 'VBN', 'next-next-shape': 'lowercase', 'next-next-word': 'marched', 'prev-iob': 'O', 'prev-lemma': 'of', 'prev-pos': 'IN', 'prev-shape': 'lowercase', 'prev-word': 'of', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'thousand', 'prev-prev-pos': 'NNS', 'prev-prev-shape': 'capitalized', 'prev-prev-word': 'Thousands', 'sentence_idx': 1.0}, {'id': 3, 'lemma': 'have', 'pos': 'VBP', 'shape': 'lowercase', 'word': 'have', 'next-lemma': 'march', 'next-pos': 'VBN', 'next-shape': 'lowercase', 'next-word': 'marched', 'next-next-lemma': 'through', 'next-next-pos': 'IN', 'next-next-shape': 'lowercase', 'next-next-word': 'through', 'prev-iob': 'O', 'prev-lemma': 'demonstr', 'prev-pos': 'NNS', 'prev-shape': 'lowercase', 'prev-word': 'demonstrators', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'of', 'prev-prev-pos': 'IN', 'prev-prev-shape': 'lowercase', 'prev-prev-word': 'of', 'sentence_idx': 1.0}, {'id': 4, 'lemma': 'march', 'pos': 'VBN', 'shape': 'lowercase', 'word': 'marched', 'next-lemma': 'through', 'next-pos': 'IN', 'next-shape': 'lowercase', 'next-word': 'through', 'next-next-lemma': 'london', 'next-next-pos': 'NNP', 'next-next-shape': 'capitalized', 'next-next-word': 'London', 'prev-iob': 'O', 'prev-lemma': 'have', 'prev-pos': 'VBP', 'prev-shape': 'lowercase', 'prev-word': 'have', 'prev-prev-iob': 'O', 'prev-prev-lemma': 'demonstr', 'prev-prev-pos': 'NNS', 'prev-prev-shape': 'lowercase', 'prev-prev-word': 'demonstrators', 'sentence_idx': 1.0}]

and the test NERC labels: ['O', 'O', 'O', 'B-geo', 'I-geo']

In [35]:

```
train2 = vec.fit_transform(training2_features[:]).toarray()
test2 = vec.fit_transform(test2_features).toarray()
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent ca
ll last)
<ipython-input-35-4d34981553c8> in <module>
----> 1 train2 = vec.fit_transform(training2_features[:]).toarray()
      2 test2 = vec.fit_transform(test2_features).toarray()

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/compressed.py
 in toarray(self, order, out)
   1022         if out is None and order is None:
   1023             order = self._swap('cf')[0]
-> 1024         out = self._process_toarray_args(order, out)
   1025         if not (out.flags.c_contiguous or out.flags.f_conti
guous):
   1026             raise ValueError('Output array must be C or F c
ontiguous')

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/base.py in _pr
ocess_toarray_args(self, order, out)
   1184             return out
   1185         else:
-> 1186             return np.zeros(self.shape, dtype=self.dtype, o
rder=order)
   1187 
   1188 

MemoryError: Unable to allocate array with shape (100000, 90530) an
d data type float64
```

In [ ]:

```
# combine train and test features in a list and represent them using one hot enc
oding
all_features = training2_features[:5] + test2_features[:5]
combined_features = vec.fit_transform(all_features).toarray()
print("\033[1m first 5 train and test features combined: \033[0m \n", combined_f
eatures)
```

[4 points] b. Train and evaluate the model and provide the classification report:

- use the SVM to predict NERC labels on the test data
- evaluate the performance of the SVM on the test data

Analyze the performance per NERC label.

Entity Linking

Excersizes 3 and 4 focus on Entity linking

Excersize 3 (NEL): Quantitative analysis [Points: 15]

In this assignment, you are going to work with two systems for entity linking: AIDA and DBpedia Spotlight. You will run them on an entity linking dataset and evaluate their performance. You will perform both quantitative and qualitative analysis of their output, and run one of these systems on your own text. We will reflect on the results of these tasks.

Note: We will use the dataset Reuters-128 in this assignment. This dataset was introduced in the notebook 'Lab4.3-Entity-linking-tools', so you probably have it already (in case you do not have it make sure you download it from Canvas first and put it in the same location as this notebook).

Exercise 1a Write code that runs both systems on the full Reuters-128 dataset. (5 points)

In [3]:

```
# Run both systems on the full Reuters-128 dataset
from rdflib import Graph, URIRef
from tqdm import tqdm
import sys
import requests
import urllib
import urllib.parse
from urllib.request import urlopen, Request
from urllib.parse import urlencode
import xml.etree.cElementTree as ET
from lxml import etree
import time
import json

# import our own utility functions and classes
import lab4_utils as utils
import lab4_classes as classes
```

In [4]:

```
aida_disambiguation_url = "https://gate.d5.mpi-inf.mpg.de/aida/service/disambigu
ate"
spotlight_disambiguation_url="http://model.dbpedia-spotlight.org/en/disambiguat
e"

def aida_disambiguation(articles, aida_url):

    with tqdm(total=len(articles), file=sys.stdout) as pbar: #use of progress ba
r
        for i, article in enumerate(articles):

            #premark entities in text
            original_content = article.content
            new_content=original_content
            for entity in reversed(article.entity_mentions):
                entity_span=new_content[entity.begin_index: entity.end_index]
                new_content=new_content[:entity.begin_index] + '[[' + entity_spa
n + ']]' + new_content[entity.end_index:]

            # Send request to aida library
            params={"text": new_content, "tag_mode": 'manual'}
            request = Request(aida_url, urlencode(params).encode())
            this_json = urlopen(request).read().decode('unicode-escape')
            try:
                results=json.loads(this_json)
            except:
                continue

            #normalize and clean the json response of aida
            dis_entities={}
            for dis_entity in results['mentions']:
                if 'bestEntity' in dis_entity.keys():
                    best_entity=dis_entity['bestEntity']['kbIdentifier']
                    clean_url=best_entity[5:] #SKIP YAGO:
                else:
                    clean_url='NIL'
                dis_entities[str(dis_entity['offset'])] = clean_url

            # store the entity with link
            for entity in article.entity_mentions:
                start = entity.begin_index
                try:
                    dis_url = str(dis_entities[str(start)])
                except:
                    dis_url='NIL'
                entity.aida_link = dis_url

            # update progress bar
            pbar.set_description('processed: %d' % (1 + i))
            pbar.update(1)
    return articles

def spotlight_disambiguate(articles, spotlight_url):

    with tqdm(total=len(articles), file=sys.stdout) as pbar:
        for i, article in enumerate(articles):

            # initiate xml structure
            annotation = etree.Element("annotation", text=article.content)
```

```
            # create surface form elements of the article data for our xml
            for mention in article.entity_mentions:
                sf = etree.SubElement(annotation, "surfaceForm")
                sf.set("name", mention.mention)
                sf.set("offset", str(mention.begin_index))
            my_xml=etree.tostring(annotation, xml_declaration=True, encoding='UT
F-8')

            # send request to spotlight and process json response
            results=requests.post(spotlight_url, urllib.parse.urlencode({'text':
my_xml, 'confidence': 0.5}),
                                  headers={'Accept': 'application/json'})

            j=results.json()
            dis_entities={}
            if 'Resources' in j:
                resources=j['Resources']
            else:
                resources=[]
            for dis_entity in resources:
                dis_entities[str(dis_entity['@offset'])] = utils.normalizeURL(di
s_entity['@URI'])

            # store spotlight link for the article
            for entity in article.entity_mentions:
                start = entity.begin_index
                if str(start) in dis_entities:
                    dis_url = dis_entities[str(start)]
                else:
                    dis_url = 'NIL'
                entity.spotlight_link = dis_url

            # update progress bar
            pbar.set_description('processed: %d' % (1 + i))
            pbar.update(1)

            # Pause for 1s to prevent overloading the server
            time.sleep(1)
    return articles

def process_both(article_set):
    processed_aida= aida_disambiguation(article_set, aida_disambiguation_url)
    processed_spotlight=spotlight_disambiguate(processed_aida, spotlight_disambi
guation_url)
    return processed_spotlight
```

In [5]:

```
reuters_file='Reuters-128.ttl'
articles=utils.load_article_from_nif_file(reuters_file)
```

In [34]:

```
# run if server always overloads when passing all articles to spotlight
set1 = articles[0:32]
set2 = articles[32:64]
set3 = articles[64:96]
set4 = articles[96:128]
```

In [37]:

```
process1 = process_both(set1)
```

```
  0%|                                                    | 0/32 [00:
00<?, ?it/s]

C:\Users\Grietje001\Anaconda3\lib\site-packages\ipykernel_launcher.
py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 32:  97%|██████████████████████████▏| 31/32 [01:01<00:0
1,  1.97s/it]
processed: 32: 100%|███████████████████████████| 32/32 [00:57<00:0
0,  1.80s/it]
```

In [39]:

```
process2 = process_both(set2)
```

```
  0%|                                                    | 0/32 [00:
00<?, ?it/s]

C:\Users\Grietje001\Anaconda3\lib\site-packages\ipykernel_launcher.
py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 32:  94%|█████████████████████████▎ | 30/32 [01:10<00:0
4,  2.34s/it]
processed: 32: 100%|███████████████████████████| 32/32 [00:58<00:0
0,  1.82s/it]
```

In [44]:

```
process3 = process_both(set3)
```

```
  0%|                                                    | 0/32 [00:
00<?, ?it/s]

C:\Users\Grietje001\Anaconda3\lib\site-packages\ipykernel_launcher.
py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 32:  97%|██████████████████████████▏| 31/32 [00:27<00:0
0,  1.15it/s]
processed: 32: 100%|███████████████████████████| 32/32 [00:56<00:0
0,  1.78s/it]
```

In [52]:

```
process4 = process_both(set4)
```

```
processed: 1:   3%|▊                          | 1/32 [00:00<00:0
5,  5.49it/s]

C:\Users\Grietje001\Anaconda3\lib\site-packages\ipykernel_launcher.
py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 32:  97%|██████████████████████████▏| 31/32 [00:45<00:0
1,  1.47s/it]
processed: 32: 100%|███████████████████████████| 32/32 [00:58<00:0
0,  1.83s/it]
```

In [53]:

```
all_processed = process1 + process2 + process3 + process4
```

Exercise 1b Write code that evaluates the two systems on this dataset by computing their overall precision, recall, and F1-score. (5 points)

In [56]:

```
# Write a function to compute the precision, recall, and F1-score for each of th
e systems on this dataset
def evaluate_entity_linking(system_decisions, gold_decisions):

    tp=0
    fp=0
    fn=0

    for gold_entity,system_entity in zip(gold_decisions,system_decisions):
        if gold_entity=='NIL' and system_entity=='NIL': continue
        if gold_entity==system_entity:
            tp+=1
        else:
            if gold_entity!='NIL':
                fn+=1
            if system_entity!='NIL':
                fp+=1

    print('TP: %d; \nFP: %d, \nFN: %d' % (tp, fp, fn))

    precision=tp/(tp+fp)
    recall=tp/(tp+fn)
    f1=2*precision*recall/(precision+recall)

    return precision, recall, f1
```

In [57]:

```
gold_link = []
aida_link = []
spot_link = []

for article in all_processed:
    for mention in article.entity_mentions:
        gold_link.append(mention.gold_link)
        aida_link.append(mention.aida_link)
        spot_link.append(mention.spotlight_link)
```

In [58]:

```
aida_evaluate = evaluate_entity_linking(aida_link, gold_link)
print(aida_evaluate)
```

```
TP: 297;
FP: 183,
FN: 353
(0.61875, 0.45692307692307693, 0.5256637168141594)
```

In [59]:

```
spotlight_evaluate = evaluate_entity_linking(spot_link, gold_link)
print(spotlight_evaluate)
```

```
TP: 300;
FP: 228,
FN: 350
(0.5681818181818182, 0.46153846153846156, 0.5093378607809848)
```

The acquired precision, recall and f1 score from the evaluations of spotlight and aida are presented in the table below

	precision	recall	f1
aida	0.61875	0.45692307692307693	0.5256637168141594
spotlight	0.5681818181818182	0.46153846153846156	0.5093378607809848

Question 1c What is the F1-score per system? Which system performs better? Is that also the better system in terms of precision and recall? Which is higher and what does that mean (hint: think of NIL entities)?(5 points)

The F1 score for aida is 0.526 and for spotlight is 0.509, aida has a slightly higher performance than spotlight. In terms of precision aida also takes the lead, this is because the number of false positives is lower for aida. A false positive in this case would entail aida providing a link for the entity whilst the gold link is NIL. The precision is influenced by the number of false positives as it takes the number of true positives and divides it by true positives plus false positives.

However, the recall for aida is slightly lower than the recall for spotlight. This is because the number of false negatives is lower for spotlight. A false negative here is when spotlight gives back a wrong link or NIL whilst the gold value is non-NIL link. The recall is affected by the number of false negatives as it divides the number of true positives by the number of true positives plus false negatives.

Excersize 4 (NEL): Qualitative analysis [Points: 15]

Exercise 2a Check the entity disambiguation by AIDA against the gold entities on the document with identifier "http://aksw.org/N3/Reuters-128/82#char=0,1370 (http://aksw.org/N3/Reuters-128/82#char=0,1370)" (write code to print the entity mentions, gold links and AIDA links). (2 points)

In [6]:

```
for index, item in enumerate(articles):
    if item.identifier == 'http://aksw.org/N3/Reuters-128/82#char=0,1370':
        break
else:
    index = -1
test_items=articles[index:index+1]
processed_aida=aida_disambiguation(test_items, aida_disambiguation_url)
processed_both=spotlight_disambiguate(processed_aida, spotlight_disambiguation_u
rl)
an_article=processed_both[0]
doc_id=an_article.identifier
print(an_article.content)
print(doc_id)
for m in an_article.entity_mentions:
    print('|mention: %s\t|gold:\t%s\t|aida:\t%s\t|' % (m.mention, m.gold_link, m
.aida_link))
```

```
processed: 1: 100%|██████████| 1/1 [00:04<00:00,  4.42s/it]
  0%|          | 0/1 [00:00<?, ?it/s]

/home/maumau/anaconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 1: 100%|██████████| 1/1 [00:02<00:00,  2.53s/it]
Exchanges and telecommunications authorities should abolish their r
estrictions on full and free dissemination of information to the in
vestment and banking communities, Reuters Holdings Plc RTRS.L chair
man Sir Christopher Hogg said. In the 1986 annual repoprt, he said 
lengthy negotiations had brought agreement with the Tokyo and Londo
n Stock Exchanges for fuller, but still not complete, access to mar
ket data through Reuter services. Many other markets maintain restr
ictions, he added. Hogg said members of some markets appear to beli
eve that information restrictions protected their interests. In oth
er cases, exchanges seem to be limiting the distribution of data in
order to provide competitive advantage to their own commercial info
rmation businesses. He also noted that despite increasing liberalis
ation in the telecommunications field, some countries continue to p
rotect their state monopolies at the expense of other economic sect
ors. Reuter dealing services remain excluded from such countries. A
s a result, banking communities serving entire economies are put at
a competitive disadvantage, he added. Reuters increased its 1986 pr
e-tax profit by 39 pct from the previous year to 130.1 mln stg on a
43 pct rise in revenues to 620.9 mln stg. Earnings per ordinary sha
re were up 47 pct to 19.4p. The annual shareholder meeting will be
held in London on April 29.
http://aksw.org/N3/Reuters-128/82#char=0,1370
|mention: Reuters Holdings Plc  |gold:  Reuters |aida:  Reuters_Gro
up      |
|mention: Christopher Hogg      |gold:  NIL     |aida:  NIL     |
|mention: Tokyo |gold:  Tokyo_Stock_Exchange    |aida:  Tokyo   |
|mention: London Stock Exchanges        |gold:  London_Stock_Exchan
ge      |aida:  NIL     |
|mention: Hogg  |gold:  NIL     |aida:  Edward_Hogg     |
|mention: Reuters       |gold:  Reuters |aida:  Reuters |
|mention: London        |gold:  London  |aida:  London  |
```

You can see in this document that one of the mentions of "Tokyo" is disambiguated wrongly by AIDA as `Tokyo` (it should be `Tokyo_Stock_Exchange`). Knowing how AIDA works, what would be your explanation for this error? (4 points)

Answer 2a

AIDA uses two types of connections in their algorithm:

The first type is between a mention and an entity instance and tells us how often is an instance is referred to by a mention. The second connection type is between two entity instances; it tells us how well-connected are two entities.

When looking why Tokyo was classified as tokyo instead of Tokyo_Stock_Exchange. It is ovious that Tokyo "the city" is mentioned more often then Tokyo_Stock_Exchange. And when looking at the second type spotlight will have looked at london and Tokyo and seeing that these two city will be be well connected the algorithm will incorrectly assign Tokyo "the city" to Tokyo. # Your answer here...

Exercise 2b Check the entity disambiguation by Spotlight against the gold entities on the document "[http://aksw.org/N3/Reuters-128/36#char=0,1146 (http://aksw.org/N3/Reuters-128/36#char=0,1146)](http://aksw.org/N3/Reuters-128/36#char=0,1146)" (write code to print the entity mentions, gold links and Spotlight links). (2 points)

In [7]:

```
for index2, item in enumerate(articles):
    if item.identifier == 'http://aksw.org/N3/Reuters-128/36#char=0,1146':
        break
else:
    index2 = -1
items2=articles[index2:index2+1]
processed_aida2=aida_disambiguation(items2, aida_disambiguation_url)
processed_both2=spotlight_disambiguate(processed_aida2, spotlight_disambiguation
_url)
an_article2=processed_both2[0]
doc_id2=an_article2.identifier
print(doc_id2)
for m in an_article2.entity_mentions:
    print('|mention: %s\t|gold:\t%s\t|spotlight:\t%s |' % (m.mention, m.gold_lin
k, m.spotlight_link))
```

```
processed: 1: 100%|██████████| 1/1 [00:04<00:00,  4.54s/it]
  0%|          | 0/1 [00:00<?, ?it/s]

/home/maumau/anaconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:19: DeprecationWarning: invalid escape sequence '\/'

processed: 1: 100%|██████████| 1/1 [00:02<00:00,  2.26s/it]
http://aksw.org/N3/Reuters-128/36#char=0,1146
|mention: U.S. Treasury |gold:  United_States_Department_of_the_Tre
asury   |spotlight:     United_States_Department_of_the_Treasury |
|mention: Group of Five |gold:  Group_of_Five   |spotlight:     Gro
up_of_Five |
|mention: Gerhard Stoltenberg   |gold:  Gerhard_Stoltenberg     |sp
otlight:        Gerhard_Stoltenberg |
|mention: Bundesbank    |gold:  Deutsche_Bundesbank     |spotlight:
German_Federal_Bank |
|mention: Karl Otto Poehl       |gold:  Karl_Otto_Pöhl  |spotlight:
NIL |
|mention: Edouard Balladur      |gold:  Édouard_Balladur        |sp
otlight:        Édouard_Balladur |
|mention: Jacques de Larosiere  |gold:  Jacques_de_Larosière    |sp
otlight:        NIL |
|mention: Kiichi Miyazawa       |gold:  Kiichi_Miyazawa |spotlight:
Kiichi_Miyazawa |
|mention: Satoshi Sumita        |gold:  Satoshi_Sumita  |spotlight:
NIL |
|mention: Robin Leigh Pemberton |gold:  Robin_Leigh-Pemberton,_Baro
n_Kingsdown     |spotlight:     NIL |
|mention: Group of Seven        |gold:  G7      |spotlight:     Gro
up_of_Seven |
|mention: Giovanni Goria        |gold:  Giovanni_Goria  |spotlight:
Giovanni_Goria |
|mention: Treasury      |gold:  United_States_Department_of_the_Tre
asury   |spotlight:     HM_Treasury |
|mention: James Baker   |gold:  James_Baker     |spotlight:     Jam
es_Baker |
|mention: Baker |gold:  James_Baker     |spotlight:     James_Baker
|
|mention: Goria |gold:  Giovanni_Goria  |spotlight:     Giovanni_Go
ria |
|mention: Group of Seven        |gold:  G7      |spotlight:     Gro
up_of_Seven |
|mention: Paris |gold:  Paris   |spotlight:     Paris |
|mention: Italy |gold:  Italy   |spotlight:     Kingdom_of_Italy |
```

You can see in this document that the mention of "Group of Seven" is disambiguated wrongly by Spotlight as `G8` (it should be `G7`). Knowing how Spotlight works, what would be your explanation for this error? (4 points)

Anwser 2b

In the above table you see the gold and spotlight linking of the text [http://aksw.org/N3/Reuters-128/36#char=0,1146 (http://aksw.org/N3/Reuters-128/36#char=0,1146)](http://aksw.org/N3/Reuters-128/36#char=0,1146). It is stated in the question that Spotlight disambiguated wrongly Group of Seven as G8. But when we run the experment the resulting link was Group_of_Seven in both instances of the Group of Seven. In the rest of the table the results seems to be also correct so can't find the reason for this result.

Question 2c In the document with identifier "http://aksw.org/N3/Reuters-128/67#char=0,1627 (http://aksw.org/N3/Reuters-128/67#char=0,1627)":

- both systems correctly decide that "Michel Dufour" is a `NIL` entity with no representation in the English Wikipedia.
- however, Spotlight later decides that "Dufour" refers to `Guillaume-Henri_Dufour`

How would you help Spotlight fix this error? (Hint: think of how you would know that "Dufour" is a NIL entity in that document) (3 points)

In [8]:

```
for index3, item in enumerate(articles):
    if item.identifier == 'http://aksw.org/N3/Reuters-128/67#char=0,1627':
        break
else:
    index3 = -1

items3=articles[index3:index3+1]
processed_aida3=aida_disambiguation(items3, aida_disambiguation_url)
processed_both3=spotlight_disambiguate(processed_aida3, spotlight_disambiguation
_url)

an_article3=processed_both2[0]
doc_id3=an_article3.identifier
print(doc_id2)
print(an_article3.content)
for m in an_article3.entity_mentions:
    print('|mention: %s\t|gold:\t%s\t|spotlight:\t%s |' % (m.mention, m.gold_lin
k, m.spotlight_link))
```

```
processed: 1: 100%|██████████| 1/1 [00:00<00:00,  1.20it/s]
  0%|          | 0/1 [00:00<?, ?it/s]

/home/maumau/anaconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:19: DeprecationWarning: invalid escape sequence '\/'
```

```
processed: 1: 100%|██████████| 1/1 [00:03<00:00,  3.34s/it]
http://aksw.org/N3/Reuters-128/36#char=0,1146
Top officials of leading industrial nations arrived at the U.S. Tre
asury main building to begin a meeting of the Group of Five. Offici
als seen arriving by Reuter correspondents included West German Fin
ance Minister Gerhard Stoltenberg and Bundesbank President Karl Ott
o Poehl, French Finance Minister Edouard Balladur and his central b
anker Jacques de Larosiere. Also seen arriving were Japanese Financ
e Minister Kiichi Miyazawa and Japans central bank governor Satoshi
Sumita and British Chancellor of the Exchequer and central bank gov
ernor Robin Leigh Pemberton. There was no immediate sign of Italian
or Canadian officials. Monetary sources have said a fully blown mee
ting of the Group of Seven is expected to begin around 3 p.m. local
time (1900 gmt) and last at least until 6 p.m. (2200 gmt), when a c
ommunique is expected to be issued. Italian sources said Italian ac
ting Finance Minister Giovanni Goria met Treasury Secretary James B
aker last night. At those talks Baker apparently convinced Goria, w
ho declined to attend the February meeting of the Group of Seven in
Paris, that Italy would participate fully in any meaningful decisio
ns.
|mention: U.S. Treasury |gold:  United_States_Department_of_the_Tre
asury   |spotlight:     United_States_Department_of_the_Treasury |
|mention: Group of Five |gold:  Group_of_Five   |spotlight:     Gro
up_of_Five |
|mention: Gerhard Stoltenberg   |gold:  Gerhard_Stoltenberg     |sp
otlight:        Gerhard_Stoltenberg |
|mention: Bundesbank    |gold:  Deutsche_Bundesbank     |spotlight:
German_Federal_Bank |
|mention: Karl Otto Poehl       |gold:  Karl_Otto_Pöhl  |spotlight:
NIL |
|mention: Edouard Balladur      |gold:  Édouard_Balladur        |sp
otlight:        Édouard_Balladur |
|mention: Jacques de Larosiere  |gold:  Jacques_de_Larosière    |sp
otlight:        NIL |
|mention: Kiichi Miyazawa       |gold:  Kiichi_Miyazawa |spotlight:
Kiichi_Miyazawa |
|mention: Satoshi Sumita        |gold:  Satoshi_Sumita  |spotlight:
NIL |
|mention: Robin Leigh Pemberton |gold:  Robin_Leigh-Pemberton,_Baro
n_Kingsdown     |spotlight:     NIL |
|mention: Group of Seven        |gold:  G7      |spotlight:     Gro
up_of_Seven |
|mention: Giovanni Goria        |gold:  Giovanni_Goria  |spotlight:
Giovanni_Goria |
|mention: Treasury      |gold:  United_States_Department_of_the_Tre
asury   |spotlight:     HM_Treasury |
|mention: James Baker   |gold:  James_Baker     |spotlight:     Jam
es_Baker |
|mention: Baker |gold:  James_Baker     |spotlight:     James_Baker
|
|mention: Goria |gold:  Giovanni_Goria  |spotlight:     Giovanni_Go
ria |
|mention: Group of Seven        |gold:  G7      |spotlight:     Gro
up_of_Seven |
|mention: Paris |gold:  Paris   |spotlight:     Paris |
|mention: Italy |gold:  Italy   |spotlight:     Kingdom_of_Italy |
```

anwsers 2c

The first time dufour is mentioned his first name is given so spotlight looks this full name up and can't find the person to connect it to and so labels it as a NIL. The second and third time dufour is mentioned only his surnamen is used. When spotligt looks this name up he does find a person to link to. This is the wrong person. This could be solved by making spotlight remender that the the dufour in the text is micheal dufour. And only add an another dufour if an other first name is given

End of this notebook