2. Group Classification ¶

After having understood the best subdivision of scenarios into groups by taking the *six most "distant" clusters in the multidimensional space* (in Scenario-Grouping.ipynb), we are ready to train the model to make it classify **questions** into **groups**.

In [1]:

```
import spacy
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline, make_pipeline, FeatureUnion
from sklearn.compose import ColumnTransformer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import Normalizer
```

After having imported the needed libraries we load our train dataframe and the **spaCy** model (https://spacy.io/models/en#en_core_web_lg) we will use. We use the *large model* because we will need vectors for *word embedding*.

In [2]:

```
nlp = spacy.load('en_core_web_lg')
train_df = pd.read_csv('dataset_intent_train.csv', sep=';')
```

We create a new column "group", that is going to be our label, by clustering scenarios.

```
In [ ]: def grouping(df):
            groups = []
            for i in df['scenario']:
                if i in ['weather', 'cooking', 'transport', 'general', 'social',
                                  'news', 'takeaway', 'qa']:
                    groups.append('a')
                elif i in ['music', 'audio', 'play']:
                    groups.append('b')
                elif i in ['recommendation', 'lists', 'datetime', 'calendar']:
                    groups.append('c')
                elif i == 'alarm':
                    groups.append('d')
                elif i == 'iot':
                    groups.append('e')
                elif i == 'email':
                    groups.append('f')

            df['group'] = groups
            return df

        grouping(train_df)
```

We then vectorize the questions, creating a *300 dimensions word embedding*.

```
In [4]: train_df['vector'] = [nlp(text).vector for text in train_df.question]
```

We define our **X** and **y**.

```
In [6]: X = train_df[['question', 'vector']]
        y = train_df['group']
```

We don't want our "vector" column to be a Series of length 300, but rather to add 300 new columns (**features**).

```
In [7]: for i, row in X.iterrows():
            for j, vec in enumerate(X.loc[i, 'vector']):
                X.loc[i, f'Vec_{j+1}'] = vec
        X = X.drop('vector', axis=1)
```

We define our:

> - Term Frequency - Inverse Document Frequency analyzer: proceding "hunder the hood" through a Bag-of-Words
> - Preprocessor: tfidf on question and normalizing the question-vector dimensions
> - Classifier: Linear Support Vector Classifier

In [8]:

```
tfidf = TfidfVectorizer(ngram_range=(1, 2))
preproc = ColumnTransformer([('tfidf', tfidf, 'question'),
                             ('scaler', Normalizer(), [i for i in X.columns[1:]])])
lsvc = LinearSVC(C=1.7, loss='hinge', max_iter=10000, class_weight='balanced')
```

We now check our accuracy cross-validating via 10 different train_test_split

In [9]:

```
acc = []

for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    pipe = make_pipeline(preproc, lsvc).fit(X_train, y_train)
    pred = pipe.predict(X_test)
    acc.append(accuracy_score(y_test, pred))
print(np.array(acc).mean())
```

```
0.959550561797753
```

We fit our entire train dataframe to our Pipeline.

In [ ]:

```
pipe_t = make_pipeline(preproc, lsvc).fit(X, y)
```

We load the test dataframe and repeat the previous vectorization processes.

In [11]:

```
df_test = pd.read_csv('testset_notarget.csv').drop('Unnamed: 0', axis=1)
df_test['vector'] = [nlp(text).vector for text in df_test.question]
Xt = df_test[['question', 'vector']]

for i, row in Xt.iterrows():
    for j, vec in enumerate(Xt.loc[i, 'vector']):
        Xt.loc[i, f'Vec_{j+1}'] = vec
Xt = Xt.drop('vector', axis=1)
```

And, finally, we predict the test questions groups.

```
In [16]: pred_t = pipe_t.predict(Xt)
```

```
In [24]: df_out = pd.concat([df_test, pd.Series(pred_t)], axis=1).drop('vector', axis=1).rename({0: 'pred_group'}, axis=1)
         df_out.head()
```

Out[24]:

	question	pred_group
0	delete item on list	c
1	what brand hair spray does donald trump use	a
2	play the song by michael jackson	b
3	what events are near me	c
4	can you reserve a ticket to grand rapids by train	a

We are now ready to proceed to the intent classifcation through BERT.