Task 5

In respect of the fifth task of the assignment, the goal is to categorize names of parties involved in patent litigation. Each name has to be classified as either "individual", "organization" or "unknown". In the field of NLP, this subtask of information extraction is generally known as named entity recognition (NER).

In [1]:

```
# import relevant packages
import numpy as np
from collections import Counter
import pandas as pd
import en_core_web_sm
```

First, the downloaded patent litigation file is loaded into the Jupyter Notebook environment. To get a quick overview of the data, the pandas info() method is used along with the sample() method to print out ten random rows. For the task at hand, the only important column is the name column. From the output below, it can be concluded that there are no missing values. It looks like the entries have some sort of underlying structure, since there are multiple records with company names, followed by their legal form, though their casing is inconsistent.

In [2]:

```
# specify file path and read in data set as a dataframe
data_file_path = "./task5_data/patent_litigation_parties.csv"
patent_litigation_df = pd.read_csv(data_file_path)
df = patent_litigation_df
print(patent_litigation_df.info())
patent_litigation_df.sample(10, random_state=42)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 6 columns):
case_row_id        4000 non-null int64
case_number        4000 non-null object
party_row_count    4000 non-null int64
party_type         4000 non-null object
name_row_count     4000 non-null int64
name               4000 non-null object
dtypes: int64(3), object(3)
memory usage: 187.6+ KB
None
```

Out[2]:

	case_row_id	case_number	party_row_count	party_type	name_row_count	name
555	26338	1:15-cv-02350-RMB-JS	139925	Counter Defendant	172623	GILEAD SCIENCES, INC.
3491	23011	1:13-cv-02778	124256	Defendant	154742	Zoje Kitchen & Bath Co., Ltd.
527	54146	3:07-cv-00710	300945	Third Party Defendant	392833	Third Party DefendantTERMINATED: 11/20/2008
3925	38014	2:09-cv-00228-WJM-MF	209893	Plaintiff	267235	TERMINATED: 06/02/2009
2989	16618	1:10-cv-00269-SLR	88610	Counter Defendant	110706	LEO Pharma A/S
70	54706	3:08-cv-01314-WHA	304640	Defendant	397716	A.C.N. 120 786 012 PTY Ltd.
1756	52699	3:03-cv-05697-WHA	292714	Defendant	381608	D&J Sports, Inc.
211	67016	6:95-cv-01574-RTH-MEM	375933	Counter Claimant	490094	Conrad Schatzle
196	36014	2:06-cv-03604-DRH-AKT	191796	Defendant	240918	IDT, Inc.
803	37156	2:08-cv-00006	202653	Counter Defendant	256827	Credit Card Fraud Control Corporation

Description of classification approach

Because of the time constraints, a simple yet effective classification technique is required. There exist rule-based and machine learning approaches for NER. Considering the task at hand, the idea is to develop a hybrid approach that first uses rule-based classification and then a pretrained classifier.

In brief, the idea is to apply a rule-based classifier that detects terms or abbreviations for the most common legal forms, for example Inc., Corp., or Limited. That way, a number of party names can be categorized into organizations with a very high accuracy. As for the remaining unclassified records, a pretrained neural model of the spacy package is imported (https://spacy.io/models/en). It can distinguish between organizations, persons, locations, dates, as well as other entities. Ultimately, the determined entities of both the rule-based and model-based approach are combined to predict one of the three party name categories.

Text preprocessing

The text preparation in this task is done in a few steps. It consists of lowercasing and punctuation removal. The preprocessed names are only needed for the rule-based classifier, so that differently written legal forms map to the same word, for example, "INC", "Inc", and "Inc." map to the same word "inc". Prevalent legal forms, which can be identified in the list of the 50 most frequent word tokens printed below, are used for the rule-based approach. In contrast, the pretrained spacy model is case sensitive and will therefore use the unprocessed names.

In [3]:

```
# create new column for preprocessed names
patent_litigation_df["preprocessed_name"] = patent_litigation_df["name"]

# apply lowercasing
patent_litigation_df["preprocessed_name"] = patent_litigation_df["preprocessed_name"].str.lower()

# remove relevant punctuation marks
for char in [",", ".", "(", ")", "/"]:
    patent_litigation_df["preprocessed_name"] = patent_litigation_df["preprocessed_name"].str.repla
ce(char, "")

# remove redundant whitespaces between words
patent_litigation_df["preprocessed_name"] = patent_litigation_df["preprocessed_name"].str.replace("
\s+", " ", regex=True)
patent_litigation_df["preprocessed_name"] = patent_litigation_df["preprocessed_name"].str.strip()

# print 50 most frequent word tokens to identify common legal forms
party_names_list = patent_litigation_df["preprocessed_name"].to_list()
party_names_list = " ".join(party_names_list).split(" ")
print("List of the 50 most frequent tokens in the form ('word', frequency):")
print(Counter(party_names_list).most_common(50))
```

```
List of the 50 most frequent tokens in the form ('word', frequency):
[('inc', 1231), ('corporation', 549), ('llc', 416), ('terminated:', 372), ('a', 325), ('company',
184), ('ltd', 118), ('technologies', 91), ('of', 90), ('delaware', 89), ('co', 89),
('international', 88), ('corp', 86), ('as', 85), ('usa', 83), ('america', 78), ('products', 76), (
'systems', 73), ('business', 68), ('incorporated', 64), ('limited', 64), ('&', 62), ('an', 57), ('
the', 56), ('doing', 55), ('industries', 54), ('california', 52), ('group', 50), ('consolidated',
48), ('electronics', 46), ('technology', 42), ('and', 39), ('pharmaceuticals', 37),
('manufacturing', 36), ('civil', 32), ('lp', 32), ('solutions', 32), ('liability', 32),
('communications', 31), ('laboratories', 30), ('action', 29), ('new', 29), ('known', 28),
('medical', 27), ('individual', 27), ('us', 26), ('north', 26), ('holdings', 26), ('united', 25),
('sa', 24)]
```

Rule-based classification

In [4]:

```
def detect_legal_form(string_arg):
    """A function that takes in a string, tokenizes it, and checks if each word token represents a
specified legal form."""
    legal_form_list = ["inc", "incorporated", "llc", "corp", "corporation",
                       "ltd", "limited", "company", "lp", "holdings", "sa", "gmbh"]
    string_arg_list = string_arg.split()
    for i in string_arg_list:
        if i in legal_form_list:
            return [("ORG", i)]
        elif i == string_arg_list[-1]:
            return []
```

```
        else: pass
```

In [5]:

```
# map detect_legal_form() function over series of preprocessed party names
patent_litigation_df["rule_based_named_entities"] = None
patent_litigation_df["rule_based_named_entities"] = patent_litigation_df["preprocessed_name"].map(
detect_legal_form)
```

Model-based classification

In [6]:

```
# load spacy entity recognition model and apply it on series of party names
spacy_ner_model = en_core_web_sm.load()
patent_litigation_df["model_based_named_entities"] = None
patent_litigation_df["model_based_named_entities"] = patent_litigation_df["name"].map(lambda x: [(y
.label_, y.text) for y in spacy_ner_model(x).ents])
```

Final prediction of class labels

In the last step, the results of the rule-based and model-based classifications are combined into a list of named entities to determine the class label. If an organizational entity "ORG" is included in the list, then the label "organization" is predicted. The same holds true if names of countries "GPE"/Geopolitical entity" are recognized, since every country by itself represents an organization of a nation. The label "individual" is predicted if, and only if, a detected name leads the model to output a "PERSON" entity without any other entity. This is important, as sometimes "ORG" and "PERSON" entities can occur together when a person's name is part of a company's name. If the prediction falls not into one of the former cases, the label "unknown" is assigned.

In [7]:

```
def classify_named_entity(list_arg):
    """A function that takes in recognized entities and outputs the final class prediction."""
    if len(list_arg) > 0:
        ne_list = list(zip(*list_arg))[0]
        if "ORG" in ne_list:
            return "organization"
        elif "GPE" in ne_list:
            return "organization"
        elif ("PERSON" in ne_list and "ORG" not in ne_list):
            return "individual"
        else: return "unknown"
    else: return "unknown"
```

In [8]:

```
# determine class of named entitiy predictions and store them in a new CSV file
patent_litigation_df["class_prediction"] = patent_litigation_df["rule_based_named_entities"] +
patent_litigation_df["model_based_named_entities"]
patent_litigation_df["class_prediction"] =
patent_litigation_df["class_prediction"].map(classify_named_entity)
patent_litigation_df.to_csv("./task5_data/patent_litigation_parties_classified.csv",
encoding="utf-8", index=False)
```

In order to get a rough idea about the classification accuracy without too much effort, 200 randomly selected party names are classified manually - the sampling was done with the command `patent_litigation_df.sample(200, random_state=42)`. The comparison with the algorithmic classifications indicated an accuracy score of 0.955.

In [9]:

```
# load manually labeled party names
data_file_path = "./task5_data/party_names_validation_data.csv"
validation_df = pd.read_csv(data_file_path)
validation_df = pd.merge(patent_litigation_df, validation_df)

# determine accuracy of classifier with numpy vector
accuracy_vector = np.where((validation_df["class_prediction"] == validation_df["class"]), 1, 0)
print("From a total of {} instances, exactly {} instances are classified correctly. The accuracy s
core is {}.".format(len(accuracy_vector), np.sum(accuracy_vector), np.mean(accuracy_vector)))
```

```
core is {}.".format(len(accuracy_vector), np.sum(accuracy_vector), np.mean(accuracy_vector)))
validation_df.to_csv("./task5_data/classifier_accuracy_data.csv", encoding="utf-8", index=False)
```

```
From a total of 200 instances, exactly 191 instances are classified correctly. The accuracy score
is 0.955.
```