Use Case : Rental Agreement Extraction

Approach Document :

1.> Extraction:

Structure the data in the form of python dictionary with all the attributes needed for extraction.

from docx.api import Document

import spacy

import pandas as pd

nlp = spacy.load('en\_core\_web\_sm')

class NlpDataStruct:

def \_\_init\_\_(self, path):

self.path = path

self.para\_dic = {}

self.table\_dic = {}

def run\_nlp(self):

document = Document(path)

tables = document.tables

paragraphs = document.paragraphs

self.para\_dic = {}

counter = 0

for para in paragraphs:

counter += 1

if para.text != '':

doc = nlp(para.text)

tokens = []

for token in doc:

tokens.append({'text': token.text, 'lemma': token.lemma\_, 'pos': token.pos\_, 'tag': token.tag\_,

'dep': token.dep\_, 'shape': token.shape\_, 'alpha': token.is\_alpha,

'is\_stop': token.is\_stop})

ner = []

for ent in doc.ents:

ner.append([ent.text, ent.label\_])

is\_ner = True if len(ner) > 0 else False

self.para\_dic[counter] = {

'text': para.text, 'ner': ner, 'tokens': tokens, 'is\_ner': is\_ner,

'alignment': para.alignment, 'indent\_first\_line': para.paragraph\_format.first\_line\_indent,

'page\_break': para.paragraph\_format.page\_break\_before, 'allcaps': para.style.font.all\_caps,

'is\_bold': para.style.font.bold,'is\_italic': para.style.font.italic,

'color': para.style.font.color.rgb,'underline': para.style.font.underline,

'fontname':[para.style.font.name](http://para.style.font.name/)}

self.table\_dic = {}

counter = 0

for table in tables:

data = []

counter += 1

keys = None

table\_frame = []

for i, row in enumerate(table.rows):

r = []

for j in range(len(row.cells)):

r.append(row.cells[j].text)

if i == 0:

keys = tuple(row.cells[j].text)

continue

doc = nlp(row.cells[j].text)

tokens = []

for token in doc:

tokens.append(

{'text': token.text, 'lemma': token.lemma\_, 'pos': token.pos\_, 'tag': token.tag\_,

'dep': token.dep\_, 'shape': token.shape\_, 'alpha': token.is\_alpha, 'is\_stop': token.is\_stop})

ner = []

for ent in doc.ents:

ner.append([ent.text, ent.label\_])

is\_tab\_ner = True if len(ner) > 0 else False

self.table\_dic[counter] = {}

self.table\_dic[counter][i] = {}

self.table\_dic[counter][i][j] = {'row': data, 'ner': ner, 'tokens': tokens,

'is\_ner': is\_tab\_ner}

table\_frame.append(r)

self.table\_dic[counter]['frame'] = pd.DataFrame(table\_frame)

self.table\_dic[counter]['alignment'] = table.alignment

self.table\_dic[counter]['style'] = table.style

self.table\_dic[counter]['alignment'] = table.alignment

path = '/home/nitin/Downloads/Sample-Rental-Agreement.docx'

doku = NlpDataStruct(path)

doku.run\_nlp()

2> Create a configuration for NER and POS tag per field:

ner\_config ={

“Agreement Value” : [ ‘CARDINAL’]

……….}

3> Ner and Pos tag paragraph Classifier (per field):

Classify the paragraph as relevant or not with respect to ner and pos configuration.

Result : we will get the pragraphs in which the fields are. On data provided it was 3 paragraphs per field average.

4> Emphasis Classifier (per field):

Checks if the words or paragraph has emphasis or not such as bold ,underline , Font name etc. change.

Result : It removed few of the paragraph which were False positive of NER and Pos Classification stage.

5> Clean the paragraph text (per field): Remove string punctuation and stop words.

6> Combine the tokens with same pos tags or ner together:

Pos ,like PROPN , is same for three words continuously we club them.

This enhance the result as some places NER misses to classify the date as ‘DATE’

We assemble all the date part together with.

7> Value Partitioning:

We identify the multipliers for ex : 11 months,

11 is a multiplier in Date value.

8> Data Transformation :

We change the format of the values to required value.

9> Rule:

1.As the Agreement Start Date < Agreement End Date.

2. Occurence of PERSON NER. (Usually the owner is mentioned before the tenants so the Person ner will be before )

10> Record building: Take multiple combination of the values and form the records.