**NLP ACTION VERBS FUNCTION**

import spacy  
nlp = spacy.load("en\_core\_web\_lg")  
  
pre\_def\_verbs = ['click', 'submit', 'review', 'edit', 'enter', 'view', 'check', 'access', 'select', 'authorize','modify','upload','login','validate','authorise']  
def\_verbs = ['click','enter','select','login']  
  
def extract\_action\_verbs(sentence):  
 try:  
 doc = nlp(sentence)  
 action\_verbs = []  
  
 rootToken = next(token for token in doc if token.dep\_ == "ROOT")  
  
 for index, token in enumerate(doc):  
 # print(token.text , token.pos\_, token.dep\_) # i want to play # i sing  
  
 if token.dep\_ == "ROOT":  
  
 # Check if ROOT and does not have xcomp children  
  
 if not any(child.dep\_ in ["ccomp", "xcomp", "pcomp"] for child in token.children):  
  
 if token.pos\_ == "VERB":  
 action\_verbs.append(token.text)  
 if token.text.lower() not in pre\_def\_verbs:  
 pre\_def\_verbs.append(token.text.lower())  
 elif token.text.lower in pre\_def\_verbs:  
 action\_verbs.append(token.text)  
  
 # If ROOT and has xcomp children, skip ROOT but consider xcomp  
  
 else:  
  
 for child in token.children:  
  
 if child.dep\_ in ["ccomp", "pcomp", "xcomp"] and child.pos\_ == "VERB":  
 action\_verbs.append(child.text)  
 if child.text.lower() not in pre\_def\_verbs:  
 pre\_def\_verbs.append(child.text.lower())  
  
 elif token.dep\_ in ["xcomp", "ccomp", "pcomp"]:  
  
 # print("Insidexcomp", token.text)  
 try:  
  
 if token.head.text != rootToken.text:  
 if token.pos\_ == "VERB":  
 action\_verbs.append(token.text)  
 if token.text.lower() not in pre\_def\_verbs:  
 pre\_def\_verbs.append(token.text.lower())  
 except:  
 pass  
  
 elif token.dep\_ == "conj":  
  
 # Check if conj has xcomp children and the child is a verb  
  
 if any(child.dep\_ in ["xcomp", "pcomp", "ccomp"] and child.pos\_ == "VERB" for child in token.children):  
  
 for child in token.children:  
  
 if child.dep\_ in ["xcomp", "pcomp", "ccomp"] and child.pos\_ == "VERB":  
 action\_verbs.append(child.text)  
 if child.text.lower() not in pre\_def\_verbs:  
 pre\_def\_verbs.append(child.text.lower())  
  
 # If conj itself is a verb and has no xcomp children, append conj  
  
 elif token.pos\_ == "VERB":  
 action\_verbs.append(token.text)  
 if token.text.lower() not in pre\_def\_verbs:  
 pre\_def\_verbs.append(token.text.lower())  
  
 if token.text.lower() in pre\_def\_verbs and token.pos\_ == "VERB" and token.text not in action\_verbs:  
 action\_verbs.append(token.text)  
  
 if token.text.lower() in def\_verbs and token.text not in action\_verbs:  
 action\_verbs.append(token.text)  
  
 except Exception as e:  
 print(str(e))  
 return []  
  
 return action\_verbs  
  
# pr = extract\_action\_verbs("On delete beneficiary screen search the benefiary using seach option with feilds customer name/ cust ID/ bene name/status")  
# print(pr)

**ACTION EXTRACTION**

def actionExtraction(self,df):  
 global\_action\_list =[]  
 combined\_list = []  
  
 step\_summary\_rows = df['DATA'][1:10]  
  
 for row in step\_summary\_rows:  
 try:  
 result = ast.literal\_eval(row)  
 for step in result['STEP\_SUMMARY']:  
 step = utils.remove\_punctuation(step)  
 action\_verbs = action.extract\_action\_verbs(step)  
 global\_action\_list.extend(action\_verbs)  
 except Exception as e:  
 print("Exception occured : ", str(e))  
  
 # Filtering Unique Verbs  
  
 global\_action\_list = list(set(action.lower() for action in global\_action\_list))

**result = ast.literal\_eval(row)**

**to convert json into dict int python**

**NLP PREDICTION CODE**

rt re  
  
def steps\_to\_list(step\_summary):  
 steps\_list = re.split(r'\d+[\.)]\s\*', step\_summary)  
 steps\_list = [step.replace('\n', '').strip() for step in steps\_list if step.strip()]  
  
 return steps\_list  
  
count = 0  
final\_dict =[]  
for id,data,expected in zip(df\_imps['Test Case id'],df\_imps['Step Summary'],df\_imps['Expected Result']):  
 index = 0  
 print(count + 1)  
 count = count + 1  
 try:  
 steplist = steps\_to\_list(data)  
 expected\_list = steps\_to\_list(expected)  
 for index,step in enumerate(steplist):  
 if index == len(steplist) - 1:  
 expected\_data = {}  
 expected\_data["TestCaseNum"] = str(id) if id is not None else None  
 expected\_data['Sequence'] = index + 2  
 ex\_Sentence = str(expected\_list[len(expected\_list)-1])  
 exmysen = ex\_Sentence  
 ex\_Sentence = utils.remove\_punctuation(ex\_Sentence)  
 ex\_Sentence = utils.cleansen(ex\_Sentence)  
 action\_verbs = action.extract\_action\_verbs(ex\_Sentence)  
 expected\_data['Action'] = ','.join(action\_verbs) if action\_verbs is not None else None  
 # Predicting remaining  
 page\_str = ""  
 field\_str = ""  
 data\_str = ""  
 ex\_Sentence = ex\_Sentence.lower()  
 ex\_Sentence = utils.removestopfrom\_sentence(ex\_Sentence)  
 doc = nlpner(ex\_Sentence)  
 for ent in doc.ents:  
 if ent.label\_ == "PAGE":  
 if page\_str == "":  
 page\_str = ent.text  
 else:  
 page\_str += ", " + ent.text  
 elif ent.label\_ == "FIELD":  
 if field\_str == "":  
 field\_str = ent.text  
 else:  
 field\_str += ", " + ent.text  
 elif ent.label\_ == "DATA":  
 if data\_str == "":  
 data\_str = ent.text  
 else:  
 data\_str += ", " + ent.text  
  
 expected\_data['ScreenName'] = page\_str  
 expected\_data['FieldName'] = field\_str  
 expected\_data['DataField'] = data\_str  
 expected\_data['Sentence'] = exmysen  
  
 row\_data = {}  
 row\_data["TestCaseNum"] = str(id) if id is not None else None  
 row\_data['Sequence'] = index+1  
 mysen = step  
 step = utils.remove\_punctuation(step)  
 step = utils.cleansen(step)  
 action\_verbs = action.extract\_action\_verbs(step)  
 row\_data['Action'] = ','.join(action\_verbs) if action\_verbs is not None else None  
  
 # Predicting remaining  
 page\_str = ""  
 field\_str = ""  
 data\_str = ""  
 step = step.lower()  
 step = utils.removestopfrom\_sentence(step)  
 doc = nlpner(step)  
 for ent in doc.ents:  
 if ent.label\_ == "PAGE":  
 if page\_str == "":  
 page\_str = ent.text  
 else:  
 page\_str += ", " + ent.text  
 elif ent.label\_ == "FIELD":  
 if field\_str == "":  
 field\_str = ent.text  
 else:  
 field\_str += ", " +ent.text  
 elif ent.label\_ == "DATA":  
 if data\_str == "":  
 data\_str = ent.text  
 else:  
 data\_str += ", " + ent.text  
  
 row\_data['ScreenName']=page\_str  
 row\_data['FieldName']=field\_str  
 row\_data['DataField']=data\_str  
 row\_data['Sentence'] = mysen  
 final\_dict.append(row\_data)  
 if index == len(steplist) - 1:  
 final\_dict.append(expected\_data)  
  
 index +=1  
 except Exception as e:  
 print("Exception occured : ", str(e))  
  
try:  
 df = pd.DataFrame(final\_dict)  
 with pd.ExcelWriter('IMPS.xlsx', 'openpyxl') as writer:  
 df.to\_excel(writer, sheet\_name='Payments\_Testcases', index=False)  
except Exception as e:  
 print(str(e))