

ML6 USE CASE

WELCOME

Text extraction from PDF to Json

THE TEAM



MARYAME



ALEX



MAARTEN



JONATHAN

TABLE OF CONTENTS

SDS
DOCUMENTATION

WORKFLOW
AND
TECHNOLOGIES

PIPELINE

- Text extraction
- Parsing
- REGEX
- Functions
- Json
- batch conversion

ADD-ONS
AND
FLAWS

SDS DOCUMENTATION

SAFETY DATA SHEET - ANNEX II

SAFETY DATA SHEET (SDS)

The Annex II of EU Regulation (EC) No 1907/2006 (REACH) has mandated what information should and could be included in each section of the SDS. This annex II is ammended by the COMMISSION REGULATION (EU) No **2015/830** in May 2015.

All SDS shall meet the latest requirements of REACH regulation.

The Commission has adopted a Regulation amending the REACH Annexes relevant for SDS – Annex II and, to a lesser extent, Annex VI. The revision brings the SDS requirements into line with the Regulation on Classification, Labelling and Packaging (Regulation (EC) No 1272/2008) and with the guidance on the preparation of SDSs as laid down in the Globally Harmonised System of Classification and Labelling of Chemicals (GHS) of the United Nations. A three column reference document compares Annex II as amended from 1 December 2010 onwards, with the relevant text of the GHS and with the original version of Annex II.

The safety data sheet shall include the following 16 headings

SECTION 1: Identification of the substance/mixture and of the company/undertaking

- 1.1. Product identifier
- 1.2. Relevant identified uses of the substance or mixture and uses advised against
- 1.3. Details of the supplier of the safety data sheet
- 1.4. Emergency telephone number

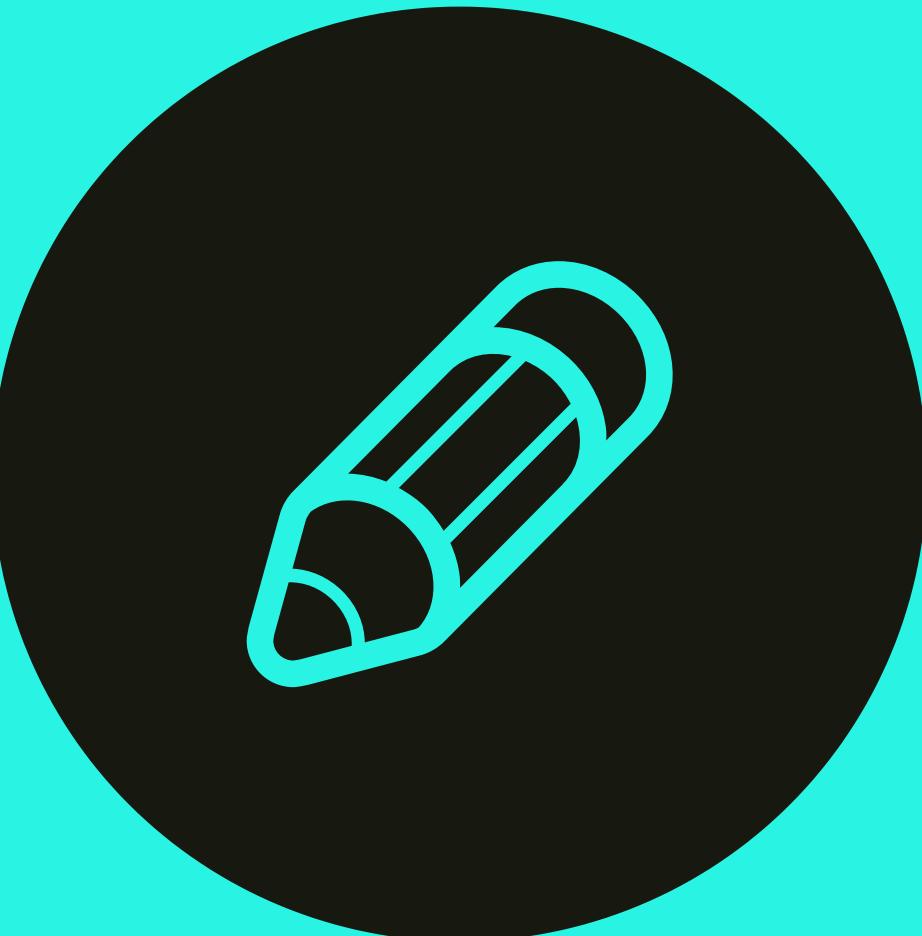
SECTION 2: Hazards identification

- 2.1. Classification of the substance or mixture
- 2.2. Label elements
- 2.3. Other hazards

SECTION 3: Composition/information on ingredients

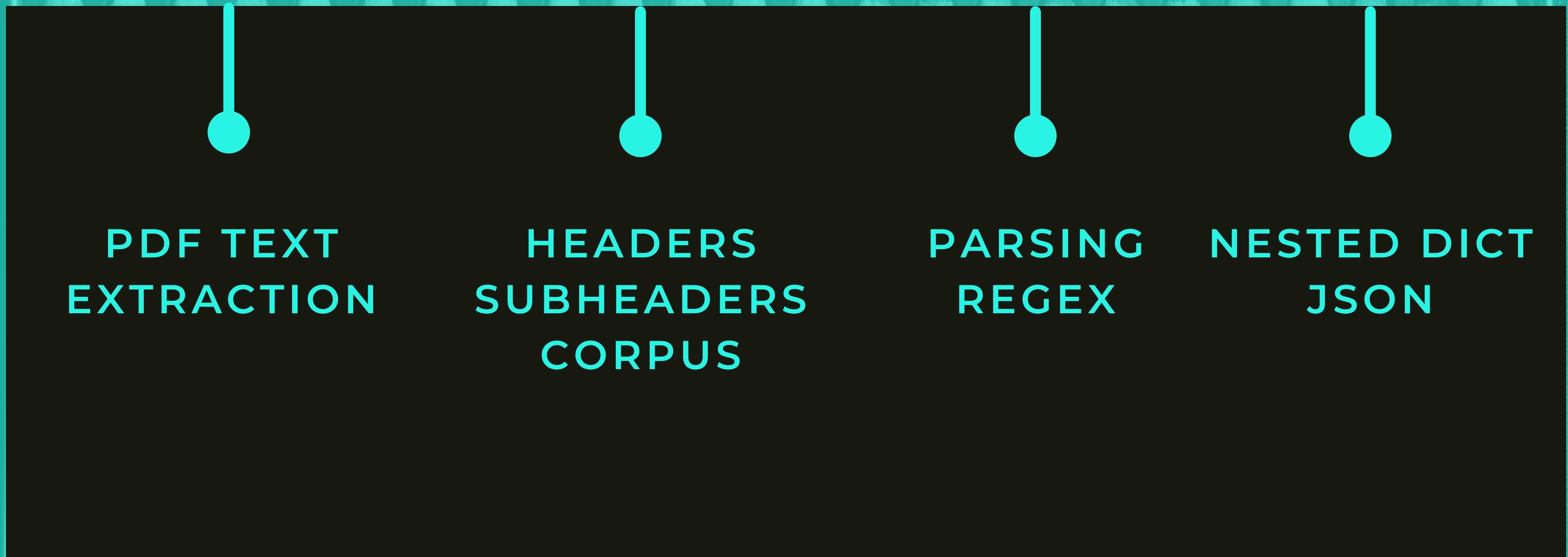
- 3.1. Substances
- 3.2. Mixtures

SECTION 4: First aid measures



```
dict_structure = {"SECTION 1: Identification of the substance":  
    {"1.1. Product identifier":"",
     "1.2. Relevant identified uses of the substance or mixture and uses advised by the manufacturer or importer",
     "1.3. Details of the supplier of the safety data sheet":"",
     "1.4. Emergency telephone number":""},  
"SECTION 2: Hazards identification":  
    {"2.1. Classification of the substance or mixture":"",
     "2.2. Label elements",
     "2.3. Other hazards"},  
"SECTION 3: Composition/information on ingredients":  
    {"3.1. Substances",
     "3.2. Mixtures"},  
"SECTION 4: First aid measures":  
    {"4.1. Description of first aid measures",
     "4.2. Most important symptoms and effects, both acute and delayed",
     "4.3. Indication of any immediate medical attention and special treatment needed"},  
"SECTION 5: Firefighting measures":  
    {"5.1. Extinguishing media",
     "5.2. Special hazards arising from the substance or mixture",
     "5.3. Advice for firefighters"},
```

WORKFLOW



PDFMINER

TEXTRACT

TIKA

REGEX

JSON

MAIN
LIBRARIES
USED

PIPELINE

TEXT EXTRACTION

```
laparams = pdfminer.layout.LAParams()
setattr(laparams, 'all_texts', True)

def miner_extract_text_from_pdf(pdf_path):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    converter = TextConverter(resource_manager, fake_file_handle, laparams=laparams)
    page_interpreter = PDFPageInterpreter(resource_manager, converter)

    with open(pdf_path, 'rb') as fh:
        for page in PDFPage.get_pages(fh,
                                      caching=True,
                                      check_extractable=True):
            page_interpreter.process_page(page)

    text = fake_file_handle.getvalue()

    # close open handles
    converter.close()
    fake_file_handle.close()

    if text:
        return text
```

```
from tika import parser

def tika_extract_text_from_pdf(pdf_path):
    # opening pdf file
    parsed_pdf = parser.from_file(pdf_path)
    # saving content of pdf
    # you can also bring text only, by parsed_pdf['text']
    # parsed_pdf['content'] #returns string
    text = parsed_pdf['content']

    if text:
        return text
```

OUTPUT

10N_Sodium_Hydroxide_(NaOH_40%)_ (6)_(US)_EN_sds

Page 1/11
Safety Data Sheet (SDS)

OSHA HazCom Standard 29 CFR 1910.1200(g) and GHS Rev 03.

Issue date 02/09/2017 Reviewed on 02/09/2017

44.2.1

- * 1 Identification
 - Product Identifier
 - Trade name: 10N Sodium Hydroxide (NaOH 40%)
 - Product Number: NGT-10N NaOH
 - Relevant identified uses of the substance or mixture and uses advised against:
No further relevant information available.

- Product Description PC21 Laboratory chemicals
- Application of the substance / the mixture: Laboratory chemicals
- Details of the Supplier of the Safety Data Sheet:
- Manufacturer/Supplier:
NuGeneration Technologies, LLC (dba NuGenTec)
1155 Park Avenue, Emeryville, CA 94608
salesteam@nugentec.com www.nugentec.com
1-888-996-8436 or 1-707-820-4080 for product information
- Emergency telephone number:
PERS Emergency Response: Domestic and Canada - 1-800-633-8253, International 1-801-629-0667

- * 2 Hazard(s) Identification
- Classification of the substance or mixture:

d~ GHS05 Corrosion
Skin Corr. 1A H314 Causes severe skin burns and eye damage.
Eye Dam. 1 H318 Causes serious eye damage.

- Label elements:
 - GHS label elements
- The product is classified and labeled according to the Globally Harmonized System (GHS).
- Hazard pictograms:

SUBDIVISION AND PARSING

```
import re

def FindOccurrences(Text, RegularExpression):
    # Find all Occurrences of SubString in text.
    # Return dict with begin and end position in Text for each occurrence
    # If not found, returns empty dictionary

    count = 1
    Dict = {}

    for m in re.finditer(RegularExpression, Text, flags= re.MULTILINE | re.IGNORECASE):
        Dict[count] = (m.start(), m.end())
        count += 1

    return(Dict)
```

```
HeaderDict = { 0 : [""], \
    1 : ["SECTION 1: Identification of the substance", "1. PRODUCT AND COMPANY IDENTIFICATION"], \
    2 : ["SECTION 2: Hazards identification", "Hazards Identification", "2. HAZARDS IDENTIFICATION"], \
    3 : ["SECTION 3: Composition/information on ingredients", "3. COMPOSITION/INFORMATION ON INGREDIENTS"], \
    4 : ["SECTION 4: First aid measures", "4. FIRST AID MEASURES", "4 First-Aid Measures"], \
    5 : ["SECTION 5: Firefighting measures", "5. FIRE-FIGHTING MEASURES", "5 Fire-Fighting Measures"], \
    6 : ["SECTION 6: Accidental release measures", "6. ACCIDENTAL RELEASE MEASURES", "6 Accidental Release Measures"], \
    7 : ["SECTION 7: Handling and storage", "7. HANDLING AND STORAGE", "7 Handling and Storage"], \
    8 : ["SECTION 8: Exposure controls/personal protection", "8. EXPOSURE CONTROLS/PERSONAL PROTECTION", "8 Exposure Controls/Personal Protection"], \
    9 : ["SECTION 9: Physical and chemical properties", "9. PHYSICAL AND CHEMICAL PROPERTIES", "9 Physical and Chemical Properties"], \
    10 : ["SECTION 10: Stability and reactivity", "10. STABILITY AND REACTIVITY", "10 Stability and Reactivity"], \
    11 : ["SECTION 11: Toxicological information", "11. TOXICOLOGICAL INFORMATION", "11 Toxicological Information"], \
    12 : ["SECTION 12: Ecological information", "12. ECOLOGICAL INFORMATION", "12 Ecological Information"], \
    13 : ["SECTION 13: Disposal considerations", "13. DISPOSAL CONSIDERATIONS", "13 Disposal Considerations"], \
    14 : ["SECTION 14: Transport information", "14. TRANSPORT INFORMATION", "14 Transport Information"], \
    15 : ["SECTION 15: Regulatory information", "15. REGULATORY INFORMATION", "15 Regulatory Information"], \
    16 : ["SECTION 16: Other information", "16. OTHER INFORMATION", "16 Other Information"] \
}
```

REGEX

```
import re

regex1 = "^[^a-zA-Z\n]*?(SECTION)?[\n\s]+?(\?<!\.).)"
regex2 = "[\n\s]+?[\.\:-]?[\n\s]+*?(\w+[\n\s/]+)*?"

sections = {
    1: "(IDENTIFICATION|Identification|identification)",
    2: "(HAZARDS|Hazard\\(s\\)|INGREDIENTS)",
    3: "(COMPOSITION|INGREDIENTS|HAZARDS|Hazard\\(s\\))",
    4: "(First[\\s-]aid)",
    5: "(Fire[\\s-]*Fighting)",
    6: "(Accidental\\sRelease)",
    7: "(Handling\\sAnd\\sStorage)",
    8: "(Exposure\\s(Controls\\s*)?[And|/]\\s*?Personal\\s(Protection)*)",
    9: "(Physical\\s(And|&)\\sChemical\\sProperties)",
    10: "(Stability\\s(And|&)\\sReactivity)",
    11: "(Toxicological\\sInformation)",
    12: "(Ecological\\sInformation)",
    13: "(Disposal\\sConsiderations*)",
    14: "(Transport(ation)*\\sInformation)",
    15: "(Regulatory\\sInformation)",
    16: "(Other\\sInformation)"
}
```

```
def GenerateHeaderRE(SectionNbr):  
    # Output the Regular expression to search for the Section Headers of section SectionNbr.  
    RE = regex1 + str(SectionNbr) + regex2 + f"{{sections[SectionNbr]}}"  
    return(RE)
```

```
SubHeaderDict = { 0: {}, \
    1: { 1 : ["Product identifier","1.1. Identification","PRODUCT NAME:", "Product Name"], \
          2 : ["Relevant identified uses of the substance or mixture and uses advised"], \
          3 : ["Details of the supplier of the safety data sheet","Supplier's details"], \
          4 : ["Emergency telephone number","Emergency", "Emergency telephone\nnumber ("], \
                }, \
    2: { 1 : ["Classification of the substance or mixture","Classification", "Emergency"], \
          2 : ["Label elements"], \
          3 : ["Other hazards", "Hazards not otherwise classified", "Potential Health Effects"], \
                }, \
    3: { 1 : ["Substances", "components"], \
          2 : ["Mixtures"] \
                }, \
    4: { 1 : ["Description of first aid measures", "4.1. Description of first aid measures"], \
          2 : ["Most important symptoms and effects, both acute and delayed", "Most important"], \
          3 : ["Indication of any immediate medical attention and special treatment needed"], \
                }, \
    5: { 1 : ["Extinguishing media", "\u2022 Suitable extinguishing agents:"], \
          2 : ["Special hazards arising from the substance or mixture", "Hazardous Combustion"], \
          3 : ["Advice for firefighters", "Special protective actions \nfor fire-fighters"], \
                }, \
    6: { 1 : ["Personal precautions, protective equipment and emergency procedures", "Personal Precautions"], \
          2 : ["Environmental precautions", "Containment Procedures"], \
          3 : ["Methods and material for containment and cleaning up", "Clean-Up Procedures"], \
          4 : ["Reference to other sections", "Special Procedures"] \
                }, \
    7: { 1 : ["Precautions for safe handling", "Handling"], \
          2 : ["Conditions for safe storage, including any incompatibilities", "Conditions for Storage"], \
          3 : ["Specific end use(s)", "Specific end use\s\)", "Specific Use"] \
                }, \
    8: { 1 : ["Control parameters", "Engineering Controls"], \
          2 : ["Exposure controls", "Substance Exposure Limits", "Exposure Guidelines"], \
                }, \
    9: { 1 : ["Information on basic physical and chemical properties", "Appearance"], \
          2 : ["Other information", "Other safety information", "NOTE:"] \
                }
```

```
from fuzzywuzzy import fuzz

def get_starting_section(first_header_pos, text):
    section = text[:first_header_pos]
    section_list = section.split("\n")
    return section_list

def RemovePageHeaders(FileText):
    # Clean the file text from headers
    # FileText: full text of PDF file
    # zero_section_list : list of all lines of the header, one line per list item

    # locate header on first page of PDF:
    AllHeaders = FindOccurrences(FileText, GenerateHeaderRE(1))
    # Pageheader = all lines before first header:
    zero_section_list = get_starting_section(AllHeaders[1][0], FileText)

    CleanFile = ""

    for division in FileText.split("\n"):
        if not(division == ""):
            PageHeaderMatch = False
            for zero in zero_section_list:
                ratio = fuzz.token_sort_ratio(zero, division)
                if ratio >= 85: # match with header, skip line
                    PageHeaderMatch = True
                    break
            if not PageHeaderMatch:
                CleanFile += (division + "\n")

    return(CleanFile)
```

FUNCTIONS

```
def PDF2SubSections(PDFfile):
    # Convert a PDF file in a Dict of Sections
    # Locate all Headers:
    HeaderIndexes = {}
    AllHeaders = {}
    for Header in range(1,17):
        AllHeaders = FindOccurrences(PDFfile, GenerateHeaderRE(Header) )
        if AllHeaders:
            HeaderIndexes[Header] = AllHeaders[1]
            #VerifyOccurrence(AllHeaders)
        else:
            print("\nHeader ", HeaderDict[Header], " not found")
            return(None)
    # Determine begin and end position of each header:
    SectionIndexes = {}
    SectionIndexes[0] = (0, HeaderIndexes[1][0] - 1)    # Section 0 is to contain PDF file intro
    for Section in range(1,16):
        SectionIndexes[Section] = (HeaderIndexes[Section][1] + 1, HeaderIndexes[Section + 1][0] - 1)
    SectionIndexes[16] = (HeaderIndexes[16][1] + 1, len(PDFfile) )
    # Extract the text of each section:
    SectionTextDict = {}
    for Section in range(0,17):
        Hlp = ExtractText(PDFfile, SectionIndexes[Section][0], SectionIndexes[Section][1])
        SubS = SubSectionsStr(Section, Hlp)
        if (SubS):
            SectionTextDict[Section] = SubS
        else: # no subsections
            SectionTextDict[Section] = Hlp
    return(SectionTextDict)
```

```
# Removing Extra Whitespaces

import re

def RemoveExtraSpaces(Text):
    """
    Return : string, text after removing duplicate whitespaces between words and leading and trailing
    Input : string, text to be cleaned
    Output : string
    """

    # Duplicate spaces and newlines between words:
    Pattern = r'\s+'
    SpaceRemoved = re.sub(pattern=Pattern, repl=" ", string=Text)

    #Remove leading and trailing whitespaces:
    return(SpaceRemoved.strip(" \n"))
```

```
def SortHeaders(HeaderIndexes):  
    # indicate which is the order the headers occur in the text  
  
    val = [item[0] for item in list(HeaderIndexes.values())]  
    SortList = sorted(val)  
    SortedHeaders = []  
    for count in range(len(SortList)):  
        for k,v in HeaderIndexes.items():  
            if v[0] == SortList[count]:  
                SortedHeaders.append(k)  
                break;  
  
    return(SortedHeaders)
```

```
def SubSectionsStr(SectionNbr, SectionText):
    # Split text of section SectionNbr from PDFSectionText into subsections
    # Correct order of subsections
    SubSectionTextDict = {}
    # Verify there is a section:
    if len(SectionText) < 8:
        return({})
    # Check if there are subsections:
    if SubHeaderDict[SectionNbr]:
        # Locate all Headers:
        HeaderIndexes = {}
        AllHeaders = {}
        # loop through all possible subheaders of the section:
        for sh in range(1,len(SubHeaderDict[SectionNbr])+1):      # indexing of subsections starts at
            # loop through all possible regular expressions for that subheader:
            for i in range(len(SubHeaderDict[SectionNbr][sh])):
                AllHeaders = FindOccurrences(SectionText, SubHeaderDict[SectionNbr][sh][i])
                if AllHeaders:
                    HeaderIndexes[sh] = AllHeaders[1] # Assume first subheader occurrence is the right one
        if not HeaderIndexes: # no sub headers found
            return({})
```

```

# Subheaders found are not necessarily in sequential order in the text.
# To allow for sequential cutting of the text body, sort the found headers
HeaderOrder = SortHeaders(HeaderIndexes)

# Determine begin and end position of each header:
SectionIndexes = {}

for SubSection in range(1, len(SubHeaderDict[SectionNbr])+1):
    # Subsection headers are optional, apply HeaderIndexes in order as in the text:
    SSlist = []
    for SubSection in HeaderOrder:
        SSlist.append(SubSection)
    if SSlist:
        if len(SSlist) == 1:
            SectionIndexes[SSlist[0]] = (HeaderIndexes[SSlist[0]][1] + 1, len(SectionText))
        else:
            for i in range(len(SSlist) - 1):
                SectionIndexes[SSlist[i]] = (HeaderIndexes[SSlist[i]][1] + 1, HeaderIndexes[SSlist[-1]][1])
            SectionIndexes[SSlist[-1]] = (HeaderIndexes[SSlist[-1]][1] + 1, len(SectionText))

    # now switch to official section order again:
    if (SSlist):
        for SubSection in HeaderIndexes:
            SubSectionTextDict[SubSection] = ExtractText(SectionText, SectionIndexes[SubSection])

return(SubSectionTextDict)

```

```
import sys

def PrintSubSections(SubSectionDict, FileName = None):
    # Print out the PDF converted to Dictionary
    # If FileName given, will write to that file, otherwise on screen display.
    if FileName:
        original_stdout = sys.stdout # Save a reference to the original standard output
        f = open(FileName, 'w')
        sys.stdout = f # Change the standard output to the file we created.
    for Section in range(0,17):
        print("-----\nSection ", Section, ":", HeaderDict[Section][0], "\n-----\r")
        if SubHeaderDict[Section]:
            if isinstance(SubSectionDict[Section], str):
                print(SubSectionDict[Section])
            elif isinstance(SubSectionDict[Section], dict):
                for sh in SubSectionDict[Section]:
                    print("-----\nSubSection ", sh, ":", SubHeaderDict[Section][sh][0], "\r")
                    print(SubSectionDict[Section][sh])
            else:
                if SubSectionDict[Section]:
                    print(SubSectionDict[Section])
        if FileName:
            sys.stdout = original_stdout # Reset the standard output to its original value
    return(None)
```

```
InputFile = './data/10N_Sodium_Hydroxide_NaOH_40_6_US_EN_sds (1).pdf'
PDFtext = tika_extract_text_from_pdf(InputFile)
CleanFile = RemovePageHeaders(PDFtext)
SubSdict = PDF2SubSections(CleanFile)
PrintSubSections(SubSdict)
```

```
=====
Section 1 : SECTION 1: Identification of the substance
=====

-----
SubSection 1 : Product identifier
-----

· Trade name: 10N Sodium Hydroxide (NaOH 40%)
· Product Number: NGT-10N NaOH
·

-----
SubSection 2 : Relevant identified uses of the substance or mixture and uses advised against
-----

No further relevant information available.

· Product Description PC21 Laboratory chemicals
· Application of the substance / the mixture: Laboratory chemicals
·

-----
SubSection 3 : Details of the supplier of the safety data sheet
-----

· Manufacturer/Supplier:
NuGeneration Technologies, LLC (dba NuGenTec)
1155 Park Avenue, Emeryville, CA 94608
salesteam@nugentec.com www.nugentec.com
1-888-996-8436 or 1-707-820-4080 for product information
·

-----
SubSection 4 : Emergency telephone number
-----


telephone number:
PERS Emergency Response: Domestic and Canada - 1-800-633-8253, International 1-801-629-0667
```

OUTPUT

JSON

```
# Create a template dictionary based on the SDS standard:  
# Top level  
SDSTemplate = { \\\n    "identification" : {"product_identifier" : "", \\\n        "uses" : "", \\\n        "supplier_details" : "", \\\n        "emergency_telephone_number" : "" \\\n    }, \\\n    "hazards" : {"classification" : "", \\\n        "label" : "", \\\n        "other" : "" \\\n    }, \\\n    "ingredients" : {"substances" : "", \\\n        "mixtures" : "" \\\n    }, \\\n    "First-Aid" : {"measures" : "", \\\n        "symptoms" : "", \\\n        "treatment" : "" \\\n    }, \\\n    "Firefighting" : {"extinguishing_media" : "", \\\n        "special_hazards" : "", \\\n        "advice" : "" \\\n    }, \\\n    "Accidents" : {"procedures" : "", \\\n        "environmental_precautions" : "", \\\n        "containment_and_cleaning" : "", \\\n        "section_reference" : "" \\\n    }, \\\n    "Handling" : {"precautions" : "", \\\n        "safe_storage" : "", \\\n        "specific_end_use" : "" \\\n    }, \\\n    "Storage" : {"precautions" : "", \\\n        "compatibility" : "", \\\n        "incompatibilities" : "" \\\n    } \\\n}
```

```
def CreateJSONdict(InDict):
    # Create a dictionary with values according to the template:
    #InDict is the processed dictionary, with heading numbers as keys
    OutputDict = SDSTemplate.copy()
    TopCount = 1
    for kt in OutputDict.keys():
        if TopCount in InDict.keys():
            #print(kt, ":", TopCount)
            if isinstance(InDict[TopCount], str):
                OutputDict[kt] = InDict[TopCount]
            else:
                SubCount = 1
                for ks in OutputDict[kt].keys():
                    #print("\t", ks, ":", SubCount)
                    if SubCount in InDict[TopCount].keys():
                        OutputDict[kt][ks] = InDict[TopCount][SubCount]
                    else:
                        OutputDict[kt][ks] = ""
                    SubCount += 1
                TopCount += 1

    return (OutputDict)
```

```
import json

def write_json(dictionary, filename):

    JsonDict = CreateJSONdict(dictionary)

    with open(filename, "w") as outfile:
        json.dump(JsonDict, outfile, indent=4)

    return(JsonDict)
```

OUTPUT

```
json_dict = write_json(SubSdict, 'json_out.txt')
```

```
json_dict
```

```
{'identification': {'product_identifier': '· Trade name: 10N Sodium Hydroxide (NaOH 40%) · Product Number: NGT-10N NaOH ·',  
    'uses': 'No further relevant information available. · Product Description PC21 Laboratory chemicals · Application of the su  
bstance / the mixture: Laboratory chemicals ·',  
    'supplier_details': '· Manufacturer/Supplier: NuGeneration Technologies, LLC (dba NuGenTec) 1155 Park Avenue, Emeryville, C  
A 94608 salesteam@nugentec.com www.nugentec.com 1-888-996-8436 or 1-707-820-4080 for product information ·',  
    'emergency_telephone_number': 'telephone number: PERS Emergency Response: Domestic and Canada - 1-800-633-8253, Internation  
al 1-801-629-0667'},  
    'hazards': {'classification': 'of the substance or mixture: d~\uf082 GHS05 Corrosion Skin Corr. 1A H314 Causes severe skin b  
urns and eye damage. Eye Dam. 1 H318 Causes serious eye damage. ·',  
        'label': '· GHS label elements The product is classified and labeled according to the Globally Harmonized System (GHS). · H  
azard pictograms: d~\uf082 GHS05 · Signal word: Danger · Hazard-determining components of labeling: Sodium Hydroxide · Hazard  
statements: H314 Causes severe skin burns and eye damage. · Precautionary statements: P260 Do not breathe dusts or mists. P28  
0 Wear protective gloves/protective clothing/eye protection/face protection. P264 Wash thoroughly after handling. P303+P361+P  
353 If on skin (or hair): Take off immediately all contaminated clothing. Rinse skin with water/shower. P305+P351+P338 If in  
eyes: Rinse cautiously with water for several minutes. Remove contact lenses, if present and easy to do. Continue rinsing. (C  
ontd. on page 2) US Page 2/11 Safety Data Sheet (SDS) OSHA HazCom Standard 29 CFR 1910.1200(g) and GHS Rev 03. Issue date 02/  
09/2017 Reviewed on 02/09/2017 Trade name: 10N Sodium Hydroxide (NaOH 40%) 44.2.1 P310 Immediately call a POISON CENTER/docto  
r. P321 Specific treatment (see supplementary first aid instructions on this Safety Data Sheet). P304+P340 IF INHALED: Remove  
person to fresh air and keep comfortable for breathing. P363 Wash contaminated clothing before reuse. P301+P330+P331 If swall  
owed: Rinse mouth. Do NOT induce vomiting. P405 Store locked up. P501 Dispose of contents/container in accordance with local/  
regional/national/ international regulations. · Unknown acute toxicity: This value refers to knowledge of known, established  
toxicological or ecotoxicological values. 0 % of the mixture consists of component(s) of unknown toxicity. · Classification s  
ystem: NFPA/HMIS Definitions: 0-Least, 1-Slight, 2-Moderate, 3-High, 4-Extreme · NFPA ratings (scale 0 - 4) 3 0 0 Health = 3  
Flame = 0 Reactivity = 0 Toxicity = 0 HMIS Rating: 3 0 0 Health = 3 Flammability = 0 Reactivity = 0 Toxicity = 0 HMIS Rating: 1 0 0
```

BATCH

```
# PDFminer based
# Only top level - TODO sublevel

import os

def MinerProcessAllPDFs(Directory):
    # Converts to sections all PDF files in Directory

    # test for existence of JSON subdirectory, otherwise create it
    if not os.path.isdir(Directory + '/JSON'):
        os.makedirs(Directory + '/JSON')

    for filename in os.listdir(Directory):
        if filename.endswith(".pdf"):
            print(f"\n=====\nFile: {filename}\n=====")
            PDFtext = miner_extract_text_from_pdf(Directory + '/' + filename)
            CleanedFile = RemovePageHeaders(PDFtext)
            PDFSectionsText = PDF2SubSections(CleanedFile)
            PrintSections(PDFSectionsText)
            write_json(PDFSectionsText, Directory + '/' + 'JSON/' + filename + '.json')

    return(None)
```

```
# Tika based
# Top level + Sublevel

import os

def TikaProcessAllPDFs(Directory):
    # Converts to sections all PDF files in Directory

    # test for existence of JSON subdirectory, otherwise create it
    if not os.path.isdir(Directory + '/JSON'):
        os.makedirs(Directory + '/JSON')

    for filename in os.listdir(Directory):
        if filename.endswith(".pdf"):
            print(f"\n=====\nFile: {filename}\n====")
            PDFtext = tika_extract_text_from_pdf(Directory + '/' + filename)
            CleanedFile = RemovePageHeaders(PDFtext)
            PDFSectionsText = PDF2SubSections(CleanedFile)
            PrintSubSections(PDFSectionsText)
            write_json(PDFSectionsText, Directory + '/' + 'JSON/' + filename + '.json')

    return(None)
```

ADD ONS & IDEAS

KEY VALUE PAIRS UNDER SUBHEADERS

- NLP ON CORPUS
- SCIENTIFIC NOTATION OF CHEMICALS (SDSPARSER)

TABLE STRUCTURE

- XML FORMAT, RECONSTRUCTION TABLE KEY-VALUE

ONTOLOGY

HAZARD PICTOGRAMS (PDFBOX)

Alexey Shigarov, Andrey Mikhailov, Andrey Altaev:
"Configurable Table Structure Recognition in
Untagged PDF Documents"

Martha O. Perez-Arriaga, Trilce Estrada, and Soraya
Abad-Mota : "TAO: System for Table Detection and
Extraction from PDF Documents"

ADD ONS & IDEAS

KEY VALUE PAIRS UNDER SUBHEADERS

- NLP ON CORPUS
- SCIENTIFIC NOTATION OF CHEMICALS (SDSPARSER)

TABLE STRUCTURE

- XML FORMAT, RECONSTRUCTION TABLE KEY-VALUE

ONTOLOGY

HAZARD PICTOGRAMS (PDFBOX)

Kobkaew Opasjumruskit, Sirko Schindler, Laura
Thiele, Philipp Matthias Schäfer: Towards Learning
from User Feedback for Ontology-based
Information Extraction

Kobkaew Opasjumruskit : "Automatic Enrichment
of Ontology for Engineering Design Process"

ADD ONS & IDEAS

KEY VALUE PAIRS UNDER SUBHEADERS

- NLP ON CORPUS
- SCIENTIFIC NOTATION OF CHEMICALS (SDSPARSER)

TABLE STRUCTURE

- XML FORMAT, RECONSTRUCTION TABLE KEY-VALUE

ONTOLOGY

HAZARD PICTOGRAMS (PDFBOX)



```
import pdfbox  
  
p = pdfbox.PDFBox()  
  
p  
  
<pdfbox.PDFBox at 0x7fa80809fbe0>  
  
InputFile = 'data/gylcol-ether-dpm-sds.pdf'  
  
# extracts images embedded in the PDF file as .png files  
p.extract_images(InputFile) # writes images to /path/to/my_file-1.png, /path/to/my_file-2.png, etc.  
  
#text extraction, in html format:  
p.extract_text(InputFile, '-html') # writes text to /path/to/my_file.txt, i.e. strips the extension
```

FLAWS

CLEAN OUTPUT

- further removal of junk content
- mixed corpus
- still anomaly sensitive

PAGE HEADER AND FOOTER

- occasionally missed

PDF STRUCTURE FLEXIBILITY

- columns
- repetitions
- order of subheaders
- PDF types

THANKS

[HTTPS://GITHUB.COM/JONATHANME658/PDF-INFO-EXTRACTION](https://github.com/jonathanme658/pdf-info-extraction)