Assignment

March 25, 2021

1 Assignment IENLP 2021

Jarco Van Roest

Francesco Di FLumeri

For the following assignment we used the components of the library “nltk”, “sklearn”, “re”, “os”, “pandas” and “itertools”. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries. PANDAS is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. SCIKIT-LEARN (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. RE module provides regular expression matching operations. OS module provides a portable way of using operating system dependent functionality. ITERTOOLS module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an “iterator algebra” making it possible to construct specialized tools succinctly and efficiently in pure Python.

1.1 First exercise

[1]:
```
import nltk, re, os, pandas
from nltk.corpus import webtext
from nltk.corpus import reuters
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize
from itertools import groupby
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from nltk.stem.porter import *
```

```python
def listToString(s):

    # initialize an empty string
    str1 = " "

    # return string
    return (str1.join(s))


java_path = "C:/Program Files/Java/jdk1.8.0_171/bin/java.exe"
os.environ['JAVAHOME'] = java_path
```

[2]:
```python
#First Part of the assignment

forum = []

for fileid in webtext.fileids():
    #print(fileid)
    forum.append(fileid)

text = webtext.raw(forum[0])
#print(len(text))
#rexp = r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:
 ↪[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:
 ↪\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:'\".,<>?«»“”‘’]))"
#rexp = r"(?i)\b(?:https?://)(?:[^\s()<>]+)"
rexp = r"(?i)\b(?:https?://|www[.])(?!http)(?!www.http)(?![*])(?:[^\s()<>]+)[.](?
 ↪:[^\s()<>]+)"
listURLS = re.findall(rexp, text)
print("Url in the file:", len(listURLS))
print(listURLS)
rexp1 = r"(?i)\b(?:ctrl|alt|win)[+](?![+])(?:[^\s()<>#-;]+)"
listCommand = re.findall(rexp1, text)
print("Commands in the file:", len(listCommand))
print(listCommand)
```

```
Url in the file: 60
['http://www.scripting.com/misc/msswitchad', 'www.foo.com',
'www.localhost.net.au"', 'http://www.watch.impress.co.jp',
'http://bugzilla.mozilla.org', 'www.aol.com', 'www.php.net', 'www.fnac.fr',
'http://mozilla.org', 'www.hvv.de', 'www.petetownshend.co.uk', 'www.google.com',
'www.wamu.com', 'www.excite.com', 'http://www.peterre.com', 'www.logitech.com',
'www.mozilla.org', 'http://texturizer.net/firebird', 'www.xy.com"',
'www.blogger.com', 'www.pcpitstop.com', 'www.mozilla.org"', 'www.zoneedit.com',
'www.libpr0n.com', 'www.us.army.mil', 'www.linuxmail.org', 'www.debian.org',
'www.lexis.com', 'http://www.lexis.com', 'www.m-w.com',
'http://www.woolworth.de', 'www.file.com', 'www.alternate.de',
'http://ftp.mozilla.org/pub/mozilla.org/firebird/nightly', 'www.microsoft.com',
'http://extensionroom.mozdev.org/more-info', 'http://www.odeon.co.uk/odeon',
'http://www.cctvusa.com', 'www.mp3.de', 'www.microsoft.com',
'http://www.trenitalia.com/home/it', 'www.rmvplus.de',
'https://www.fortify.net', 'www.microsoft.com', 'http://irc-galleria.net',
"www.uboot.com'", 'www.microsoft.com', 'http://www.mozilla.org/products',
'www.lycos.co.uk"', 'www.calciomercato.com', 'http://labs.google.com/cgi-bin',
'www.odeon.co.uk/odeon', 'www.atozwebtools.com', 'www.X.com',
'http://www.timbressuisses.ch', 'www.vipernetworks.com',
'https://www.eposasp.com/ebpp', 'www.yahoo.com', 'www.intellicast.com',
'www.w3c.org']
Commands in the file: 120
['Ctrl+Mousewheel', 'ctrl+tab', 'Ctrl+Mousewheel', 'ALT+F', 'ctrl+dragging',
'Ctrl+Click', 'ctrl+Click', 'CTRL+click', 'Ctrl+Enter', 'Ctrl+B', 'Ctrl+Enter',
'CTRL+F', 'Ctrl+Tab', 'Ctrl+Shift', 'Ctrl+Space', 'Alt+Home', 'ctrl+d',
'ctrl+t', 'ctrl+enter', 'alt+enter', 'ctrl+enter', 'ctrl+enter', 'ctrl+enter',
'Ctrl+L', 'Ctrl+G', 'ctrl+enter', 'Ctrl+Q', 'Ctrl+W', 'alt+back', 'Ctrl+L',
'alt+f', 'Ctrl+L', 'alt+scroll', 'alt+scroll', 'Ctrl+Enter', 'ALT+D', 'Alt+D',
'Alt+d', 'Ctrl+Enter', 'CTRL+Enter', 'ctrl+pgup', 'Alt+Enter', 'ALT+d',
'Ctrl+S', 'Alt+Click', 'Alt+F', 'Alt+drag', 'CTRL+K', 'Ctrl+W', 'Ctrl+W',
'Alt+Enter', 'Ctrl+M', 'Ctrl+K', 'Ctrl+Tab', 'CTRL+E', 'alt+enter', 'alt+enter',
'Ctrl+W', 'ctrl+T', 'Ctrl+P', 'Alt+Enter', 'Alt+Enter', 'Ctrl+Tab', 'CTRL+Y',
'CTRL+L', 'Ctrl+K', 'Ctrl+K', 'Alt+D', 'Ctrl+A', 'Ctrl+F', 'Ctrl+Tab',
'Ctrl+Enter', 'Ctrl+W', 'Ctrl+tab', 'Ctrl+x', 'Alt+f', 'ctrl+K', 'ctrl+K',
'Alt+D', 'Ctrl+Enter', 'Ctrl+K', 'Ctrl+Shift', 'Ctrl+mnemonic', 'Ctrl+E',
'Ctrl+wheel', 'ctrl+pgup', 'Alt+f', 'Alt+Enter', 'Ctrl+P', 'Ctrl+K', 'Alt+D',
'ctrl+f', 'Ctrl+W', 'ctrl+t', 'Alt+Left', 'Ctrl+mouse', 'ALT+F',
'Ctrl+Backspace', 'Ctrl+Backspace', 'CTRL+Backspace', 'CTRL+B', 'CTRL+I',
'Ctrl+Backspace', 'Ctrl+Shift', 'CTRL+PGUP', 'Alt+f', 'Ctrl+Return', 'CTRL+Y',
'CTRL+E', 'Ctrl+Enter', 'Ctrl+E', 'Ctrl+Enter', 'CTRL+TAB', 'CTRL+ALT',
'Ctrl+Tab', 'Ctrl+Shift', 'Ctrl+E', 'ctrl+e', 'Ctrl+F', 'Ctrl+F']
```

The first exercise of the assignment required to build two regular expressions in order to find all the URLs and all the keyboard shortcuts in a file, precisely "firefox.txt". After having read the file content with the method "raw()" of the library "webtext" we proceeded with the construnction of the regular expressions. At the beggining of both the regular expressions we put the character "(?i)", which specifies that the search has to be case insensitive and it"whole words only". The first regular expression searches for all the strings that start with"www." or "https://" and this is done with the construct "(?:https?://|www[.])". Moreover, as it was noticed that there were some urls in the form "https://https", "www.www.http" or "www.*" we add the construct "(?!http)(?!www.http)(?[*])" in order to avoid them. Finally we add the construct "(?:[^\s()<>]+)." to specify that the root domain and the top level domain have to be not empty in the url. The second regular expression searches for all the strings that start with "ctrl+", "alt+" or "win+" and this is done by the construct "(?:ctrl|alt|win)[+]". Then we search for a serie of characters different from "+()<>#-;" with the construct "(?:[^\s()<>#-;]+)". In the end the number of URLs found in the file is 60, while the number of commands is 120.

1.2 Second exercise

[3]:
```
#Second Part of the assignment

jar_location = 'C:/Users/utente/Desktop/Drawer/EIT_MasterSchool/DataScience/
 ↪IENLP/stanford-ner-2020-11-17/stanford-ner.jar'
model_location_3classes = 'C:/Users/utente/Desktop/Drawer/EIT_MasterSchool/
 ↪DataScience/IENLP/stanford-ner-2020-11-17/classifiers/english.all.3class.
 ↪distsim.crf.ser.gz'
model_location_4classes = 'C:/Users/utente/Desktop/Drawer/EIT_MasterSchool/
 ↪DataScience/IENLP/stanford-ner-2020-11-17/classifiers/english.conll.4class.
 ↪distsim.crf.ser.gz'
model_location_7classes = 'C:/Users/utente/Desktop/Drawer/EIT_MasterSchool/
 ↪DataScience/IENLP/stanford-ner-2020-11-17/classifiers/english.muc.7class.
 ↪distsim.crf.ser.gz'
st3 = StanfordNERTagger(model_location_3classes,jar_location,encoding='utf-8')
st4 = StanfordNERTagger(model_location_4classes,jar_location,encoding='utf-8')
st7 = StanfordNERTagger(model_location_7classes,jar_location,encoding='utf-8')

hamlet = nltk.corpus.gutenberg.words('shakespeare-hamlet.txt')
hamletText = listToString(hamlet)

tokenized_text = word_tokenize(hamletText)
text_ner3 = st3.tag(tokenized_text)
text_ner4 = st4.tag(tokenized_text)
text_ner7 = st7.tag(tokenized_text)

def text_classification(ner):
    people = []
    for tag, chunk in groupby(ner, lambda x:x[1]):
        if tag == "PERSON":
            word = ""
            for w, t in chunk:
                word += w
            people.append(word)
    return people

p = []
print("**** 3 classes ****")
p = text_classification(text_ner3)
peopleNoDuplicates = list(dict.fromkeys(p))
print("Number of people: ", len(peopleNoDuplicates))
print(peopleNoDuplicates)

p = []
print("**** 4 classes ****")
p = text_classification(text_ner4)
```

```
peopleNoDuplicates = list(dict.fromkeys(p))
print("Number of people: ", len(peopleNoDuplicates))
print(peopleNoDuplicates)

p = []
print("**** 7 classes ****")
p = text_classification(text_ner7)
peopleNoDuplicates = list(dict.fromkeys(p))
print("Number of people: ", len(peopleNoDuplicates))
print(peopleNoDuplicates)
```

```
**** 3 classes ****
Number of people:  317
['WilliamShakespeare', 'ScoenaPrima', 'Barnardo', 'Francisco', 'FranciscoFran',
'Horatio', 'Marcellus', 'Hor', 'Giue', 'Starre', 'Scholler', 'HoratioBarn',
'Marke', 'HoratioHora', 'Barn', 'HoratioHor', 'Heauen', 'Norwey', 'Law',
'Heraldrie', 'Hamlet', 'Motiue', 'Loe', 'MarcellusMar', 'Cocke', 'Obiect',
'SauioursBirch', 'Morne', 'Walkes', 'ScenaSecunda', 'ClaudiusKingofDenmarke',
'GertrudetheQueene', 'Ophelia', 'Colleagued', 'Cornelius', 'Vncle', 'Voltemand',
'Giuing', 'Laer', 'Sonne', 'Blacke', 'CourtierCosin', 'SonneQu', 'MadamKing',
'ManetHamlet', 'Selfe', 'Niobe', 'Ile', 'WittenbergHoratio', 'Beene',
'SableSiluer', 'Doe', 'Ophel', 'Bulke', 'Hee', 'Carue', 'Moone', 'Primrose',
'Steele', 'HamletPolon', 'MyLord', 'RoyallDane', 'Soules', 'Cliffe', 'Heau',
'DenmarkeHor', 'Fiers', 'Rankly', 'Angell', 'Milke', 'Lazar', 'Queene', 'Booke',
'Braine', 'Vnmixt', 'Villaine', 'HamletHor', 'Hillo', 'HeauenMar', 'Graue',
'SaintPatricke', 'Marcell', 'Reynoldo', 'ReynoldoReynol', 'Danskers', 'Polon',
'Rouse', 'Reynol', 'Obserue', 'MusickeReynol', 'Vndertakings', 'Rosincrane',
'Rosincrance', 'Guildensterne', 'Moreouer', 'Neighbour', 'Rosin',
'EntreatieGuil', 'GuildensterneQu', 'Guil', 'Policie', 'Qu', 'Voltumand',
'SayVoltumand', 'Volt', 'Giues', 'Madam', 'Maiestie', 'Breuitie', 'Soule',
'SoulesIdoll', 'Sunne', 'Hammes', 'Honestie', 'Madnesse', 'Sanitie',
'Rosincran', 'Guild', 'Denmarke', 'Lenton', 'Yases', 'SuccessionRosin', 'Boyes',
'Naturall', 'Philosophie', 'Elsonower', 'Garbe', 'Hawke', 'Hearke', 'Rossius',
'Buzze', 'Iephta', 'AeneasTale', 'Pyrrhus', 'SableArmesBlacke', 'Stoopes',
'Milkie', 'Orbe', 'Priam', 'Naue', 'Baudry', 'BissonRheume', 'Dost', 'Gonzago',
'Hecuba', 'Iohn', 'Coward', 'Lye', 'Liuer', 'Scullion', 'Diuel', 'Melancholly',
'WhereinIle', 'LunacyRosin', 'Gertrude', 'Lawes', 'Bodkin', 'Nimph', 'Rich',
'BeautieOphe', 'Beautie', 'Ophe', 'Ham', 'Fellowes', 'HeauensHam',
'FarwellOphe', 'Rose', 'Honie', 'QueeneMother', 'Cryer', 'Herod', 'Scorne',
'Bodie', 'Norman', 'Hora', 'Hindges', 'Euen', 'IuliusCaesar', 'Ladie',
'Memorie', 'MichingMalicho', 'PatientlieHam', 'Hymen', 'Wormwood',
'WormwoodBapt', 'Willes', 'WifeHam', 'Dukes', 'Baptista', 'Rauen', 'Lucian',
'Damon', 'PaiockeHora', 'Musick', 'MusickeGuild', 'Libertie',
'AduancementRosin', 'Clowd', 'Masse', 'SouleofNero', 'Seales', 'MaiestieRosin',
'Armour', 'Fetters', 'Raine', 'Iustice', 'blacke', 'Sallery', 'Rood', 'Ducate',
'KillesPoloniusQu', 'Seale', 'Moore', 'Frost', 'WillQu', 'StyeQu', 'Chappell',
'HamletHam', 'KinneRosin', 'Fox', 'Hoa', 'Begger', 'MotherKing', 'HamletHamlet',
'Mother', 'Armie', 'Cap', 'greeneTurfe', 'OpheliaOphe', 'Valentine', 'Alacke',
'Yong', 'Thicke', 'LaertesLaer', 'Iuggel', 'Childe', 'Burne', 'DoueLaer',
'Steward', 'Rosemary', 'Fennell', 'Rew', 'Robin', 'AllFlaxen', 'Gramercy',
'ChristianSoules', 'OpheliaLaer', 'Hatchment', 'Saylor', 'Chace', 'Saile',
'Shippe', 'Wood', 'Claudio', 'Natur', 'NormanLaer', 'LamoundKin', 'Sancturize',
'Requit', 'Brooke', 'Pul', 'GoodmanDeluerClown', 'Argall', 'Adam', 'Carpenter',
'Asse', 'Clowne', 'Scull', 'Chaplesse', 'Pate', 'Sheepe', 'Calues', 'Carde',
'Kibe', 'Ifaith', 'Tanner', 'YoricksScull', 'Yorick', 'Alexander', 'Caesar',
'Cerimony', 'Peebles', 'SoulesLaer', 'GraueLaer', 'Depriu', 'Pelion',
'DaneLaer', 'Spleenatiue', 'HamletGen', 'Akers', 'Doue', 'Dogge', 'HoratioHam',
'Villaines', 'Deuis', 'Yeomans', 'Assis', 'Modell', 'Popt', 'Osricke',
'DenmarkeHam', 'Bonet', 'Osr', 'Girdle', 'Germaine', 'Cannon', 'Complie',
'Foyles', 'CousenHamlet', 'Stopes', 'AntikeRoman', 'HeauenIle', 'Storie',
'Prince', 'Fortin', 'FINIS', 'PrinceofDenmarke']
**** 4 classes ****
Number of people:  494
['WilliamShakespeare', 'ActusPrimus', 'Barnardo', 'Fran', 'FranciscoFran',
'Horatio', 'Marcellus', 'DaneFran', 'MarcellusMar', 'Fantasie', 'Starre',
'Scholler', 'HoratioBarn', 'HoratioHora', 'HoratioHor', 'Warlike', 'Barn',
'God', 'Armour', 'Pollax', 'Martiall', 'Taske', 'Dar', 'Heraldrie', 'Cou',
'Hamlet', 'Shark', 'Motiue', 'Sourse', 'Romage', 'Loe', 'Maiesticall', 'Ayre',
'MockeryBarn', 'Cocke', 'ThroateAwake', 'Obiect', 'SauioursBirch', 'Walkes',
'EasterneHill', 'ScenaSecunda', 'ClaudiusKing', 'Gertrude', 'Ophelia',
'Ioyntresse', 'DoleTaken', 'Colleagued', 'Voltemand', 'Cornelius', 'Vncle',
'Bedrid', 'Giuing', 'Dane', 'Head', 'Natiue', 'Laer', 'Denmarke', 'Pollonius',
'CosinHamlet', 'Sonne', 'Seeke', 'Madam', 'InkyCloake', 'Mother', 'Blacke',
'Riuer', 'Moods', 'Seeme', 'Minde', 'AnVnderstanding', 'CourtierCosin',
'SonneQu', 'ManetHamlet', 'Hiperion', 'Vnkle', 'Ile', 'Horatio)Hor', 'Heauens',
'Beene', 'Ielly', 'SableSiluer', 'Hell', 'Conuoy', 'Violet', 'Froward', 'Ophel',
'Bulke', 'Maid', 'Prodigall', 'CankerGalls', 'Ophe', 'Doe', 'Shew',
'LibertineHimselfe', 'Primrose', 'Grapple', 'Soule', 'Steele', 'HamletPolon',
'Polon', 'Girle', 'Vnsifted', 'MarryIle', 'Springes', 'Woodcocks', 'Broakers',
'Keepes', 'RoyallDane', 'Canoniz', 'Hearsed', 'SepulcherWherein', 'Reuisits',
'Moone', 'Soules', 'Cliffe', 'Heau', 'Marke', 'Doom', 'Fiers', 'Angell',
'Gates', 'Aygre', 'Milke', 'Tetter', 'Queene', 'Vnhouzzled', 'Matine', 'Adue',
'Booke', 'Braine', 'Vnmixt', 'Villaine', 'HamletHor', 'Hillo', 'Graue',
'SaintPatricke', 'Marcell', 'Armes', 'Reynoldo', 'ReynoldoReynol', 'Reynol',
'Danskers', 'Quarelling', 'LordPolon', 'CountryReynol', 'GentlemanPolon',
'Rouse', 'MusickeReynol', 'LordHamlet', 'Vngartred', 'Vndertakings', 'Heauen',
'Rosincrane', 'Moreouer', 'Neighbour', 'Gentrie', 'Hope', 'Rosin',
'EntreatieGuil', 'Rosincrance', 'GuildensterneQu', 'Guildensterne', 'Voltumand',
'Highnesse', 'Maiestie', 'Giues', 'Dutie', 'Breuitie', 'Madnesse', 'Perpend',
'SoulesIdoll', 'Lier', 'Mistris', 'Magots', 'Carrion', 'Amber', 'Hammes',
'Honestie', 'Sanitie', 'Rosincran', 'Wards', 'Guil', 'MaiesticallRoofe',
'Angel', 'Lenton', 'Knight', 'Yases', 'SuccessionRosin', 'Controuersie',
'Boyes', 'Hercules', 'Vnckle', 'Naturall', 'Philosophie', 'Garbe', 'AuntMother',
'Winde', 'Hawke', 'Prophesie', 'Rossius', 'Buzze', 'Historie', 'Historicall',
'Plautus', 'IephtaIudge', 'PonsChanson', 'Ladiship', 'Cauiarie', 'Sallets',
'Dido', 'Pyrrhus', 'SableArmesBlacke', 'Trick', 'Sonnes', 'Bak', 'Murthers',
'Carbuncles', 'PyrrhusOldeGrandsirePriam', 'ForeGod', 'Rebellious', 'Stoopes',
'Milkie', 'ReuerendPriam', 'TyrantPyrrhus', 'Newtrall', 'Orbe', 'Priam',
'Baudry', 'BissonRheume', 'Loines', 'Pesant', 'Hecuba', 'Rascall', 'Coward',
'Liuer', 'Gall', 'SlauesOffall', 'Remorselesse', 'Letcherous', 'Asse', 'Deere',
'Drab', 'Scullion', 'Malefactions', 'Diuell', 'Weaknesse', 'Melancholly',
'WhereinIle', 'LunacyRosin', 'Deuotions', 'HarlotsCheeke', 'Arrowes',
'Deuoutly', 'Scornes', 'Lawes', 'Bodkin', 'BeautieOphe', 'Beautie', 'Fellowes',
'Knaues', 'Dowrie', 'FarwellOphe', 'Wantonnesse', 'Honie', 'Bels', 'AndIle',
'Cryer', 'Smoothnesse', 'Pery', 'Herod', 'Pagan', 'Norman', 'Villanous',
'Hindges', 'VulcansStythe', 'Torches', 'Ifaith', 'IuliusCaesar', 'Brutus',
'Ladie', 'Memorie', 'Anon', 'Poysoner', 'MichingMalicho', 'MischeefeOphe',
'Poesie', 'Moones', 'Hymen', 'Vnite', 'Sunne', 'MooneMake', 'Honour',
'Wormwood', 'WormwoodBapt', 'Fruite', 'Greefe', 'Frend', 'OurWilles', 'Gonzago',
'Baptista', 'Lucianus', 'Rauen', 'Drugges', 'HecatsBan', 'Magicke', 'Murtherer',
'Hart', 'Turke', 'ProuinciallRoses', 'Damon', 'ThisRealme', 'PaiockeHora',
'Rim', 'Musick', 'MusickeGuild', 'Closset', 'Libertie', 'Musicke', 'Organe',
'Clowd', 'CamellPolon', 'Masse', 'WeazellPolon', 'Daggers', 'Seales',
'MaiestieRosin', 'Somnet', 'Spoakes', 'Ruine', 'Fetters', 'Raine', 'Iustice',
'Buyes', 'Iledoo', 'Sallery', 'ThisPhysicke', 'Rood', 'MotherQu', 'Ducate',
'Hyperions', 'Seale', 'Mildew', 'Moore', 'Vertue', 'Ardure', 'WillQu', 'Stew',
'StyeQu', 'Soldiours', 'Grace', 'Vnkles', 'Gibbe', 'Secrecie', 'Vnpegge',
'Maiesty', 'MotherClossets', 'Chappell', 'KinneRosin', 'Spundge', 'King',
'Guild', 'Fox', 'Hee', 'Begger', 'Progresse', 'Quicknesse', 'MotherKing',
'HamletHamlet', 'Armie', 'FortinbrasClaimes', 'Spurnes', 'Turfe', 'OpheliaOphe',
'Valentine', 'Alacke', 'Thicke', 'OpheliaDiuided', 'Antiquity', 'Diuinity',
'Iuggel', 'LooserLaer', 'Childe', 'Burne', 'DeereMaid', 'DoueLaer', 'Steward',
'Rosemary', 'Fennell', 'Rew', 'Daysie', 'Robin', 'AllFlaxen', 'Gramercy',
'ChristianSoules', 'OpheliaLaer', 'Saylor', 'Pyrate', 'Warlicke', 'Chace',
'Saile', 'Shippe', 'Sith', 'Selfe', 'Claudio', 'Natur', 'NormanLaer',
'AndIemme', 'Sancturize', 'Brooke', 'Nettles', 'Daysies', 'Purples',
'Clambring', 'Pul', 'GoodmanDeluerClown', 'Argall', 'Ditchers', 'Adam', 'Mason',
'Yaughan', 'Clowne', 'Scull', 'Chaplesse', 'Mazard', 'Pickhaxe', 'Spade',
'Shouell', 'Conueyances', 'Boxe', 'Carde', 'Kibe', 'Tanner', 'Decayer', 'Scul',
'YoricksScull', 'Yorick', 'Alexander', 'ImperiallCaesar', 'Wall', 'Cerimony',
'MarkeLaer', 'Shardes', 'Flints', 'Peebles', 'BuriallLaer', 'Requiem',
'SoulesLaer', 'GraueLaer', 'Depriu', 'DaneLaer', 'Spleenatiue', 'Vntill', 'Woo',
'MakeOssa', 'Doue', 'Mew', 'Dogge', 'HoratioHam', 'Grop', 'Larded', 'Denmarks',
'Goblins', 'Villaines', 'Deuis', 'Yeomans', 'Coniuration', 'Assis', 'Modell',
'Subscrib', 'Popt', 'Portraiture', 'Osricke', 'Chowgh', 'Bonet', 'Poniards',
'Germaine', 'Lapwing', 'Complie', 'Foyles', 'Gauntlets', 'MotherLaer',
'CousenHamlet', 'QueeneCarowses', 'MadamKing', 'Woodcocke', 'Vnbated',
'DamnedDane', 'Dyes', 'Heere', 'Storie', 'Drumme', 'Fortin', 'Inuite',
'SouldioursMusicke']
**** 7 classes ****
Number of people:  200
['WilliamShakespeare', 'Barnardo', 'Horatio', 'Marcellus', 'Leige', 'DaneFran',
'MarcellusMar', 'Heauen', 'Armour', 'Martiallstalke', 'Loe', 'Cocke',
'SauioursBirch', 'ClaudiusKing', 'Gertrude', 'Cornelius', 'King', 'Sonne',
'Blacke', 'Moods', 'CourtierCosin', 'KingsRouce', 'Selfe', 'Hiperion', 'Niobe',
'Heauens', 'Beene', 'Moone', 'Doe', 'Shew', 'LibertineHimselfe', 'Primrose',
'Soule', 'selfeHaue', 'greeneGirle', 'Bloudburnes', 'LordHamlet', 'RoyallDane',
'Canoniz', 'Soules', 'Cliffe', 'Heau', 'Vertue', 'Angell', 'Milke', 'MostLazar',
'Booke', 'Graue', 'SaintPatricke', 'Marcell', 'Marke', 'GentlemanPolon',
'Rosincrane', 'Rosincrance', 'Neighbour', 'NephewesLeuies', 'Giues',
'Businesse', 'Breuitie', 'Celestiall', 'SoulesIdoll', 'Sunne', 'deereOphelia',
'Carrion', 'Hammes', 'Rosin', 'MaiesticallRoofe', 'Lenton', 'Boyes', 'Garbe',
'Hawke', 'Polon', 'Pastoricall', 'Comicall', 'Historicall', 'SableArmesBlacke',
'Bak', 'winde', 'loe', 'Milkie', 'Pyrrhus', 'BissonRheume', 'Liuer',
'SlauesOffall', 'Scullion', 'Braine', 'LunacyRosin', 'Giue', 'Gertrudeleaue',
'Naturall', 'mortallcoile', 'Lawes', 'Bodkin', 'Fellowes', 'Madnesse', 'Cryer',
'Whirle', 'Pery', 'Scorne', 'Bodie', 'Norman', 'deereSoule', 'cramm',
'IuliusCaesar', 'Diuelweareblacke', 'MichingMalicho', 'Hymen', 'Feare',
'Greefe', 'Willes', 'Dukes', 'Rauen', 'ThynaturallMagicke', 'Deere', 'Hart',
'Damondeere', 'Musick', 'wonderfullSonne', 'Masse', 'Seales', 'MaiestieRosin',
'Fetters', 'Raine', 'Steele', 'Rood', 'Hyperionscurles', 'Seale', 'Moore',
'Chappell', 'KinneRosin', 'KingsCountenance', 'Hee', 'leaneBegger',
'greeneTurfe', 'Shrow', 'Alacke', 'Thicke', 'Keepes', 'Burne', 'downe',
'Steward', 'Rosemary', 'Fennell', 'Rew', 'Robin', 'ChristianSoules',
'Colaterall', 'Saylor', 'Chace', 'Guildensterne', 'Wisedome', 'Claudio',
'Natur', 'NormanLaer', 'fantastickeGarlands', 'Crow', 'Nettles', 'Daysies',
'Brooke', 'Pul', 'Crowner', 'Argall', 'Adam', 'Mason', 'Carpenter', 'Gallowes',
'Scull', 'Morrow', 'Chaplesse', 'Shouell', 'LordHoratio', 'Tanner', 'Scul',
'YoricksScull', 'Alexander', 'Beere', 'ImperiallCaesar', 'BuriallLaer',
'SoulesLaer', 'MinistringAngell', 'GraueLaer', 'Depriu', 'DaneLaer', 'Dogge',
'Deuis', 'Yeomans', 'Garlandweare', 'Assis', 'Subscrib', 'Bonet', 'Germaine',
'Hall', 'Ilegaine', 'Foyles', 'CousenHamlet', 'Dane', 'strick', 'Drumme',
'CaptainesBeareHamlet', 'SouldioursMusicke']
```

The second exercise required to perform a named entity recognition on the file, "shakespeare-hamlet.txt, in order to find all the people in the corpus. The recognition allows to categorize the various words written in a text. In this tecnique the categories are indicated by classes and a larger number of classes indicates a less general classification. The operations have been performed by using the Stanford NER tagger, which a Java implementation of Named Entity recognizer. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. Included with the download are good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION). However, we performed the classification also with 4 (LOCATION, PERSON, ORGANIZATION, MISC) and 7 classes (LOCATION, PERSON, ORGANIZATION, MONEY, PERCENT, DATE, TIME), in order to compare the results. Before starting with the recognition, we performed a text tokenization in order to have each word represented as a token. Then we proceeded with the text classification and we obtained the results which are shown above: - with a 3 classes classification the number of people is 317 - with a 4 classes classification the number of people is 494 - with a 7 classes classification the number of people is 200

1.3 Third exercise

[4]:
```
#Third part of the assignment

categories = reuters.categories()
nb = MultinomialNB()
stemmer = PorterStemmer()

def stop_words_removing(text):
    for w in stopwords.words('english'):   #removing stop words
            text = text.replace(w, '')
    return text

def stemming(text):
    tokens = nltk.word_tokenize(text)         #performing Stemming
    for w in tokens:
        text = text.replace(w, stemmer.stem(w))
    return text

#classification with stop-words removal pre-processing
vect = CountVectorizer(stop_words='english')
X_train = []
y_train = []
X_test = []
y_test = []

for f in reuters.fileids():
    if (f.find("training")!=-1):
        text = reuters.raw(f)
        catText = reuters.categories(f)
        #text = stop_words_removing(text)
        for c in catText:
            X_train.append(text)
            y_train.append(categories.index(c))
    else:
        text = reuters.raw(f)
        catText = reuters.categories(f)
        #text = stop_words_removing(text)
        for c in catText:
            X_test.append(text)
            y_test.append(categories.index(c))

vect.fit(X_train)
X_train_dtm = vect.transform(X_train)

X_test_dtm = vect.transform(X_test)

nb.fit(X_train_dtm, y_train)
```

```python
y_test_pred = nb.predict(X_test_dtm)

accScore = metrics.accuracy_score(y_test, y_test_pred)
print("Precision with stop words removing: ",accScore)


#classification with Stemming pre-processing
vect = CountVectorizer()
X_train = []
y_train = []
X_test = []
y_test = []

for f in reuters.fileids():
    if (f.find("training")!=-1):
        text = reuters.raw(f)
        catText = reuters.categories(f)
        text = stemming(text)
        for c in catText:
            X_train.append(text)
            y_train.append(categories.index(c))
    else:
        text = reuters.raw(f)
        catText = reuters.categories(f)
        text = stemming(text)
        for c in catText:
            X_test.append(text)
            y_test.append(categories.index(c))

vect.fit(X_train)
X_train_dtm = vect.transform(X_train)

X_test_dtm = vect.transform(X_test)

nb.fit(X_train_dtm, y_train)
y_test_pred = nb.predict(X_test_dtm)

accScore = metrics.accuracy_score(y_test, y_test_pred)
print("Precision with Stemming: ", accScore)
```

```
Precision with stop words removing:  0.6933760683760684
Precision with Stemming:  0.6733440170940171
```

The third exercise required to perform a text classification on a set of files, in order to understand their topics. In order to perform the classification we used a Naive Based model which has been trained by using the files named with the label “training”. After the training we made the prediction by using the set of files called “test” and we obtained the category of each of them. In the model the x dimension was represented by the text contained in the files, while the y dimension referred to the categories. Before proceeding with the classfier, it can be obeserved that we used the CountVectorizer object in order to turn a collection of text documents into numerical feature vectors (creating the Bag of Words). After the prediction we measured the accurancy by comparing the categories predicted with the actual ones. The text classification was performed both with the stop words removal and with the Stemming preprocessing and it emerged that the precision is greater whenever the text is subjected to a removing stop-words operation.

1.3.1 Discussion on how to improve the text classification

From the above section it can be observed that text classifition in combination with Stemming lead to a better results than the case it is performed with Stop Words Removing. However, it can be asserted that we performed the evaluation based only on the accurancy while Jurafsky and Martin affirm that the assessment of a text classfier should be realized by analyzing **Precision** and **Recall**. The former measures the percentage of the items that the system detected that are positive in reality, while the latter measures the percentace of items actually present in the input that were correctly identified by the system. Hence, for future works the first thing that can be implemented is the evaluation of the classifiers based on the **F-measure**, which is the combination of precision and recall in the following form:

$$\frac{(\beta^2+1)PR}{\beta^2 P+R}$$

where β differentially weights the importance of recall(R) and precision(P).

Another problem identified in the text classification performed in the section above is represented by the size of the test set, which might be not large enough to representative as it is equal only to the 28% of the whole collection of data. A solution can be found by applying a **10-fold-cross-validation**: this technique is described by Jurafsky and Martin and it consists in randomly choosing a training and a test division of the data for 10 times, and average these 10 runs to obtain an average error rate.

Furthermore, it can be seen that the activity of removing the stop words does not improve the classifier performance, as well as the action of Stemming. Hence, it can be suggested to make use of the entire vocubulary in order to have a faster execution, because the program will not have to analyze and transform each corpus of the training and the test set.

Finally, a way to improve the classfier performance could be represented by excluding those words which appear in the majority or in just a couple of documents. Precisely the solution is to drop the very frequent or infrequent words because these can influence the classification negatively.