DTU02901_professor

September 19, 2019

1 Exploring and undertanding documental databases with topic models and graph analysis

1.1 Exercise notebook

Version 1.0
Date: Aug 31, 2017
Authors:

- Jerónimo Arenas-García (jeronimo.arenas@uc3m.es)
- Jesús Cid-Sueiro (jcid@tsc.uc3m.es)

[1]:
```
# Common imports

import numpy as np
# import pandas as pd
# import os
from os.path import isfile, join
# import scipy.io as sio
# import scipy
import zipfile as zp
# import shutil
# import difflib
```

```
Vendor:  Continuum Analytics, Inc.
Package: mkl
Message: trial mode expires in 30 days
```

1.2 1. Corpus acquisition

In this block we will work with collections of text documents. The objectives will be:

- Find the most important topics in the collection and assign documents to topics
- Analyze the structure of the collection by means of graph analysis

We will work with a collection of research projects funded by the US National Science Foundation, that you can find under the `./data` directory. These files are publicly available from the NSF website.

(*As a side note, there are many other available text collections to work with. In particular, the NLTK library has many examples, that you can explore using the* `nltk.download()` *tool.*

```
import nltk
nltk.download()
```

for instance, you can take the gutemberg dataset

```
Mycorpus = nltk.corpus.gutenberg
text_name = Mycorpus.fileids()[0]
raw = Mycorpus.raw(text_name)
Words = Mycorpus.words(text_name)
```

Also, tools like Gensim or Sci-kit learn include text databases to work with).

1.3 1.1. Exploring file structure

NSF project information is provided in XML files. Projects are yearly grouped in `.zip` files, and each project is saved in a different XML file. To explore the structure of such files, we will use the file `160057.xml`. Parsing XML files in python is rather easy using the `ElementTree` module.

To introduce some common functions to work with XML files we will follow this tutorial.

1.3.1 1.1.1. File format

To start with, you can have a look at the contents of the example file. We are interested on the following information of each project:

- Project identifier
- Project Title
- Project Abstract
- Budget
- Starting Year (we will ignore project duration)
- Institution (name, zipcode, and state)

[2]:
```
xmlfile = '../data/1600057.xml'

with open(xmlfile,'r') as fin:

    print(fin.read())
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rootTag>
<Award>
<AwardTitle>Novel States in Spin-Orbit-Coupled and Correlated
Materials</AwardTitle>
<AwardEffectiveDate>08/01/2016</AwardEffectiveDate>
```

<AwardExpirationDate>12/31/2016</AwardExpirationDate>
<AwardAmount>306810</AwardAmount>
<AwardInstrument>
<Value>Continuing grant</Value>
</AwardInstrument>
<Organization>
<Code>03070000</Code>
<Directorate>
<LongName>Direct For Mathematical &amp; Physical Scien</LongName>
</Directorate>
<Division>
<LongName>Division Of Materials Research</LongName>
</Division>
</Organization>
<ProgramOfficer>
<SignBlockName>Tomasz Durakiewicz</SignBlockName>
</ProgramOfficer>
<AbstractNarration>Non-technical Abstract:&lt;br/&gt;Modern condensed matter physics research has produced novel materials with fundamental properties that underpin a remarkable number of cutting-edge technologies. It is now generally accepted that novel materials are necessary for critical advances in technologies and whoever discovers novel materials generally controls the science and technology of the future. Transition metal oxides have attracted enormous interest within both the basic and applied science communities. However, for many decades, the overwhelming balance of effort was focused on the 3d-elements (such as iron, copper, etc.) and their compounds; the heavier 4d- and 5d-elements (such as ruthenium, iridium, etc., which constitute two thirds of the d-elements listed in the Periodic Table) and their compounds have been largely ignored until recently. The principal investigator seeks to discover novel materials containing 4d- and/or 5d-elements and understand how they offer wide-ranging opportunities for the discovery of new physics and, ultimately, new device paradigms. This project also provides rigorous training to all students involved, focusing on synthesis and characterization techniques covering a broad spectrum of materials and experimental probes available in the principal investigator's laboratory. &lt;br/&gt;&lt;br/&gt;Technical Abstract:&lt;br/&gt;Physics driven by spin-orbit interactions is among the most important topics in contemporary condensed matter physics. Since the spin-orbit interaction is comparable to the on-site Coulomb and other relevant interactions, it creates a unique balance between competing interactions that drive complex behaviors and exotic states not observed in other materials. The project encompasses a systematic effort to elucidate physics of novel phenomena in spin-orbit-coupled and correlated materials and a rigorous search for new materials having exotic ground states. This project focuses on the following areas: (1) Novel phenomena at high pressures and high magnetic fields, (2) Unusual correlations between the insulating gap and magnetic transition in iridates and ruthenates, (3) Exotic metallic and superconducting states in iridates, (4) Mott insulators with "intermediate-strength" spin-orbit interaction and other competing energies, and (5) Single-crystal synthesis and

```
search for novel materials. The principal investigator is one of a few key
pioneers who have initiated seminal studies on iridates and, before that,
ruthenates, and has comprehensive facilities and proven expertise for single-
crystal synthesis and wide-ranging studies of structural, transport, magnetic,
thermal and dielectric properties as functions of temperature, magnetic field,
pressure and doping.</AbstractNarration>
<MinAmdLetterDate>08/05/2016</MinAmdLetterDate>
<MaxAmdLetterDate>08/05/2016</MaxAmdLetterDate>
<ARRAAmount/>
<AwardID>1600057</AwardID>
<Investigator>
<FirstName>Gang</FirstName>
<LastName>Cao</LastName>
<EmailAddress>gang.cao@colorado.edu</EmailAddress>
<StartDate>08/05/2016</StartDate>
<EndDate/>
<RoleCode>Principal Investigator</RoleCode>
</Investigator>
<Institution>
<Name>University of Kentucky Research Foundation</Name>
<CityName>Lexington</CityName>
<ZipCode>405260001</ZipCode>
<PhoneNumber>8592579420</PhoneNumber>
<StreetAddress>109 Kinkead Hall</StreetAddress>
<CountryName>United States</CountryName>
<StateName>Kentucky</StateName>
<StateCode>KY</StateCode>
</Institution>
<ProgramElement>
<Code>1710</Code>
<Text>CONDENSED MATTER PHYSICS</Text>
</ProgramElement>
<ProgramElement>
<Code>1712</Code>
<Text>DMR SHORT TERM SUPPORT</Text>
</ProgramElement>
<ProgramReference>
<Code>9150</Code>
<Text>EXP PROG TO STIM COMP RES</Text>
</ProgramReference>
</Award>
</rootTag>
```

1.3.2 1.1.2. Parsing XML

XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree. The `ElementTree` module has two classes for this purpose:

- `ElementTree` represents the whole XML document as a tree
- `Element` represents a single node in this tree

We can import XLM data by reading an XML file:

[3]:
```python
import xml.etree.ElementTree as ET
tree = ET.parse(xmlfile)
root = tree.getroot
```

or directly reading a string:

[4]:
```python
root = ET.fromstring(open(xmlfile,'r').read())
```

`fromstring()` parses XML from a string directly into an `Element`, which is the root element of the parsed tree. Other parsing functions may create an `ElementTree`, but we will not cover them here.

As an `Element`, root has a tag and a dictionary of attributes:

[5]:
```python
print(root.tag)
print(root.attrib)
```

```
rootTag
{}
```

It also has children nodes over which we can iterate:

[6]:
```python
for child in root:
    print(child.tag, child.attrib)
```

```
Award {}
```

Children are nested, and we can access specific child nodes by index. We can also access the text of specified elements. For instance:

[7]:
```python
for child in root[0]:
    print(child.tag, child.attrib, child.text)
```

```
AwardTitle {} Novel States in Spin-Orbit-Coupled and Correlated Materials
AwardEffectiveDate {} 08/01/2016
AwardExpirationDate {} 12/31/2016
AwardAmount {} 306810
AwardInstrument {}

Organization {}

ProgramOfficer {}

AbstractNarration {} Non-technical Abstract:<br/>Modern condensed matter physics
research has produced novel materials with fundamental properties that underpin
a remarkable number of cutting-edge technologies. It is now generally accepted
that novel materials are necessary for critical advances in technologies and
whoever discovers novel materials generally controls the science and technology
```

```
of the future. Transition metal oxides have attracted enormous interest within
both the basic and applied science communities.  However, for many decades, the
overwhelming balance of effort was focused on the 3d-elements (such as iron,
copper, etc.) and their compounds; the heavier 4d- and 5d-elements (such as
ruthenium, iridium, etc., which constitute two thirds of the d-elements listed
in the Periodic Table) and their compounds have been largely ignored until
recently. The principal investigator seeks to discover novel materials
containing 4d- and/or 5d-elements and understand how they offer wide-ranging
opportunities for the discovery of new physics and, ultimately, new device
paradigms. This project also provides rigorous training to all students
involved, focusing on synthesis and characterization techniques covering a broad
spectrum of materials and experimental probes available in the principal
investigator's laboratory.  <br/><br/>Technical Abstract:<br/>Physics driven by
spin-orbit interactions is among the most important topics in contemporary
condensed matter physics. Since the spin-orbit interaction is comparable to the
on-site Coulomb and other relevant interactions, it creates a unique balance
between competing interactions that drive complex behaviors and exotic states
not observed in other materials. The project encompasses a systematic effort to
elucidate physics of novel phenomena in spin-orbit-coupled and correlated
materials and a rigorous search for new materials having exotic ground states.
This project focuses on the following areas: (1) Novel phenomena at high
pressures and high magnetic fields, (2) Unusual correlations between the
insulating gap and magnetic transition in iridates and ruthenates, (3) Exotic
metallic and superconducting states in iridates, (4) Mott insulators with
"intermediate-strength" spin-orbit interaction and other competing energies, and
(5) Single-crystal synthesis and search for novel materials. The principal
investigator is one of a few key pioneers who have initiated seminal studies on
iridates and, before that, ruthenates, and has comprehensive facilities and
proven expertise for single-crystal synthesis and wide-ranging studies of
structural, transport, magnetic, thermal and dielectric properties as functions
of temperature, magnetic field, pressure and doping.
MinAmdLetterDate {} 08/05/2016
MaxAmdLetterDate {} 08/05/2016
ARRAAmount {} None
AwardID {} 1600057
Investigator {}

Institution {}

ProgramElement {}

ProgramElement {}

ProgramReference {}
```

The presented classes and functions are all you need to solve the following exercise. However, there are many other interesting functions that can probably make it easier for you to work with

XML files. For more information, please refer to the ElementTree API.

1.3.3 1.1.3. Exercise: Parsing the XML project files

Implement a function that parses the XML files and provides as its output a dictionary with fields:

```
project_code      (string)
title             (string)
abstract          (string)
budget            (float)
year              (string)
institution       (tuple with elements: name, zipcode, and statecode)
```

[8]:
```python
def parse_xmlproject(xml_string):
    """This function processess the specified XML field,
    and outputs a dictionary with the desired project information

    :xml_string: String with XML content
    :Returns: Dictionary with indicated files
    """

    #<SOL>
    root = ET.fromstring(xml_string)
    dictio = {}

    for child in root[0]:
        if child.tag.lower() == 'awardtitle':
            dictio['title'] = child.text
        elif child.tag.lower() == 'awardeffectivedate':
            dictio['year'] = str(child.text[-4:])
        elif child.tag.lower() == 'awardamount':
            dictio['budget'] = float(child.text)
        elif child.tag.lower() == 'abstractnarration':
            dictio['abstract'] = child.text
        elif child.tag.lower() == 'awardid':
            dictio['project_code'] = child.text
        elif child.tag.lower() == 'institution':
            #For the institution we have to access the children elements
            #and search for the name, zipcode, and statecode only
            name = ''
            zipcode = ''
            statecode = ''
            for child2 in child:
                if child2.tag.lower() == 'name':
                    name = child2.text
                elif child2.tag.lower() == 'zipcode':
                    zipcode = child2.text
                elif child2.tag.lower() == 'statecode':
                    statecode = child2.text
            dictio['institution'] = (name, zipcode, statecode)

    return dictio
    #</SOL>

parse_xmlproject(open(xmlfile,'r').read())
```

[8]:
```
{'abstract': 'Non-technical Abstract:<br/>Modern condensed matter physics
research has produced novel materials with fundamental properties that underpin
a remarkable number of cutting-edge technologies. It is now generally accepted
that novel materials are necessary for critical advances in technologies and
whoever discovers novel materials generally controls the science and technology
of the future. Transition metal oxides have attracted enormous interest within
both the basic and applied science communities.  However, for many decades, the
overwhelming balance of effort was focused on the 3d-elements (such as iron,
copper, etc.) and their compounds; the heavier 4d- and 5d-elements (such as
ruthenium, iridium, etc., which constitute two thirds of the d-elements listed
in the Periodic Table) and their compounds have been largely ignored until
recently. The principal investigator seeks to discover novel materials
containing 4d- and/or 5d-elements and understand how they offer wide-ranging
opportunities for the discovery of new physics and, ultimately, new device
paradigms. This project also provides rigorous training to all students
involved, focusing on synthesis and characterization techniques covering a broad
spectrum of materials and experimental probes available in the principal
investigator\'s laboratory.  <br/><br/>Technical Abstract:<br/>Physics driven by
spin-orbit interactions is among the most important topics in contemporary
condensed matter physics. Since the spin-orbit interaction is comparable to the
on-site Coulomb and other relevant interactions, it creates a unique balance
between competing interactions that drive complex behaviors and exotic states
not observed in other materials. The project encompasses a systematic effort to
elucidate physics of novel phenomena in spin-orbit-coupled and correlated
materials and a rigorous search for new materials having exotic ground states.
This project focuses on the following areas: (1) Novel phenomena at high
pressures and high magnetic fields, (2) Unusual correlations between the
insulating gap and magnetic transition in iridates and ruthenates, (3) Exotic
metallic and superconducting states in iridates, (4) Mott insulators with
"intermediate-strength" spin-orbit interaction and other competing energies, and
(5) Single-crystal synthesis and search for novel materials. The principal
investigator is one of a few key pioneers who have initiated seminal studies on
iridates and, before that, ruthenates, and has comprehensive facilities and
proven expertise for single-crystal synthesis and wide-ranging studies of
structural, transport, magnetic, thermal and dielectric properties as functions
of temperature, magnetic field, pressure and doping.',
 'budget': 306810.0,
 'institution': ('University of Kentucky Research Foundation',
  '405260001',
  'KY'),
 'project_code': '1600057',
 'title': 'Novel States in Spin-Orbit-Coupled and Correlated Materials',
 'year': '2016'}
```

1.4 1.2. Building the dataset

Now, we will use the function you just implemented, to create a database that we will use throughout this module.

For simplicity, and given that the dataset is not too large, we will keep all projects in the RAM. The dataset will consist of a list containing the dictionaries associated to each of the considered projects in a time interval.

[9]:
```
# Construct an iterator (or a list) for the years you want to work with
years = range(2015,2017)
datafiles_path = '../data/'
NSF_data = []

for year in years:

    zpobj = zp.ZipFile(join(datafiles_path, str(year)+'.zip'))
    for fileinzip in zpobj.namelist():
        if fileinzip.endswith('xml'):

            #Some files seem to be incorrectly parsed
            try:
                project_dictio = parse_xmlproject(zpobj.read(fileinzip))
                if project_dictio['abstract']:
                    NSF_data.append(project_dictio)
            except:
                pass
```

We will extract some characteristics of the constructed dataset:

[10]:
```
print('Number of projects in dataset:', len(NSF_data))

####
budget_data = list(map(lambda x: x['budget'], NSF_data))
print('Average budget of projects in dataset:', np.mean(budget_data))

####
insti_data = list(map(lambda x: x['institution'], NSF_data))
print('Number of unique institutions in dataset:', len(set(insti_data)))

####
counts = dict()
for project in NSF_data:
    counts[project['year']] = counts.get(project['year'],0) + 1
```

```
print('Breakdown of projects by starting year:')
for el in counts:
    print(el, ':', counts[el])
```

```
Number of projects in dataset: 24342
Average budget of projects in dataset: 342411.624435
Number of unique institutions in dataset: 2786
Breakdown of projects by starting year:
2018 : 3
2016 : 12401
2014 : 344
2017 : 2554
2015 : 9039
2013 : 1
```

Exercise Compute the average length of the abstracts of all projects in the dataset

[11]:
```
#<SOL>
abstractlen_data = list(map(lambda x: len(x['abstract']), NSF_data))
print('Average length of projects abstracts (in characters):', np.
 ↪mean(abstractlen_data))
#</SOL>
```

```
Average length of projects abstracts (in characters): 2605.88780708
```

2 2. Corpus Processing

Topic modelling algorithms process vectorized data. In order to apply them, we need to transform the raw text input data into a vector representation. To do so, we will remove irrelevant information from the text data and preserve as much relevant information as possible to capture the semantic content in the document collection.

Thus, we will proceed with the following steps:

1. Tokenization
2. Homogeneization
3. Cleaning
4. Vectorization

2.1 2.1. Tokenization

For the first steps, we will use some of the powerful methods available from the Natural Language Toolkit. In order to use the `word_tokenize` method from nltk, you might need to get the appropriate libraries using `nltk.download()`. You must select option "d) Download", and identifier "punkt"

[12]:
```
import nltk

# You should comment this code fragment if the package is already available.
# Select option "d) Download", and identifier "punkt"
# nltk.download()
```

We will create a list that contains just the abstracts in the dataset. As the order of the elements in a list is fixed, it will be later straightforward to match the processed abstracts to metadata associated to their corresponding projects.

[13]:
```
from nltk.tokenize import word_tokenize

NSF_abstracts = list(map(lambda x: x['abstract'], NSF_data))

tokenized_abstracts = []
nprojects = len(NSF_abstracts)

for n, abstract in enumerate(NSF_abstracts):
    if not n%100:
        print('\rTokenizing abstract', n, 'out of', nprojects, end='',␣
 ↪flush=True)
    tokenized_abstracts.append(word_tokenize(abstract))

print('\n\n The corpus has been tokenized. Check the result for the first␣
 ↪abstract:')
print(NSF_abstracts[0])
print(tokenized_abstracts[0])
```

```
Tokenizing abstract 24300 out of 24342

 The corpus has been tokenized. Check the result for the first abstract:
The past few years have seen unprecedented growth in mobile data consumption.
Powered in large part by the rapid adoption of smart phones and tablets, the
growth in wireless data creates phenomenal challenges for the wireless industry,
which has been unable to meet the demand for rich mobile content through
cellular networks. This has led to the investigation of solutions by network
operators that aim to utilize WiFi radios present in these mobile devices to
deliver content without using the cellular radio links, also known as content-
offloading.  Industry-led approaches aim to utilize WiFi infrastructure in the
form of access points to offload this content, but these have various deployment
issues. Research has lately focused on the potential of proximity-based peer
content sharing, since proximity enables low-power, high speed data exchanges
which in turn allows mobile devices to proactively share data with one another.
No large-scale study using real-world situations has established the potential
for such content-sharing to provide capacity gains, until now. This proposal
aims to conduct solid, preliminary pilot studies evaluating several of the
foundational claims in this area, specifically as to whether sufficient
potential exists in the right time, right place, and with reasonably viable
deployment scenarios. The proposed work will gather and evaluate pilot data in
both highly amenable environments (WiFi across multiple tailgate offerings of
hundreds of users) as well as more challenging environments (daily commuter
trains to / from Chicago).<br/><br/>The broader impact of the work will be to
either demonstrate the potential viability for proximity-based solutions or to
present compelling evidence that such proximity solutions are unlikely to yield
significant benefits. Further broader impacts for the work include data sharing
capabilities with respect to temporal characterizations of redundancy across
mobile devices in several real-world scenarios.
['The', 'past', 'few', 'years', 'have', 'seen', 'unprecedented', 'growth', 'in',
'mobile', 'data', 'consumption', '.', 'Powered', 'in', 'large', 'part', 'by',
'the', 'rapid', 'adoption', 'of', 'smart', 'phones', 'and', 'tablets', ',',
'the', 'growth', 'in', 'wireless', 'data', 'creates', 'phenomenal',
'challenges', 'for', 'the', 'wireless', 'industry', ',', 'which', 'has', 'been',
'unable', 'to', 'meet', 'the', 'demand', 'for', 'rich', 'mobile', 'content',
'through', 'cellular', 'networks', '.', 'This', 'has', 'led', 'to', 'the',
'investigation', 'of', 'solutions', 'by', 'network', 'operators', 'that', 'aim',
'to', 'utilize', 'WiFi', 'radios', 'present', 'in', 'these', 'mobile',
'devices', 'to', 'deliver', 'content', 'without', 'using', 'the', 'cellular',
'radio', 'links', ',', 'also', 'known', 'as', 'content-offloading', '.',
'Industry-led', 'approaches', 'aim', 'to', 'utilize', 'WiFi', 'infrastructure',
'in', 'the', 'form', 'of', 'access', 'points', 'to', 'offload', 'this',
'content', ',', 'but', 'these', 'have', 'various', 'deployment', 'issues', '.',
'Research', 'has', 'lately', 'focused', 'on', 'the', 'potential', 'of',
'proximity-based', 'peer', 'content', 'sharing', ',', 'since', 'proximity',
'enables', 'low-power', ',', 'high', 'speed', 'data', 'exchanges', 'which',
'in', 'turn', 'allows', 'mobile', 'devices', 'to', 'proactively', 'share',
'data', 'with', 'one', 'another', '.', 'No', 'large-scale', 'study', 'using',
'real-world', 'situations', 'has', 'established', 'the', 'potential', 'for',
'such', 'content-sharing', 'to', 'provide', 'capacity', 'gains', ',', 'until',
'now', '.', 'This', 'proposal', 'aims', 'to', 'conduct', 'solid', ',',
'preliminary', 'pilot', 'studies', 'evaluating', 'several', 'of', 'the',
'foundational', 'claims', 'in', 'this', 'area', ',', 'specifically', 'as', 'to',
'whether', 'sufficient', 'potential', 'exists', 'in', 'the', 'right', 'time',
',', 'right', 'place', ',', 'and', 'with', 'reasonably', 'viable', 'deployment',
'scenarios', '.', 'The', 'proposed', 'work', 'will', 'gather', 'and',
'evaluate', 'pilot', 'data', 'in', 'both', 'highly', 'amenable', 'environments',
'(', 'WiFi', 'across', 'multiple', 'tailgate', 'offerings', 'of', 'hundreds',
'of', 'users', ')', 'as', 'well', 'as', 'more', 'challenging', 'environments',
'(', 'daily', 'commuter', 'trains', 'to', '/', 'from', 'Chicago', ')', '.', '<',
'br/', '>', '<', 'br/', '>', 'The', 'broader', 'impact', 'of', 'the', 'work',
'will', 'be', 'to', 'either', 'demonstrate', 'the', 'potential', 'viability',
'for', 'proximity-based', 'solutions', 'or', 'to', 'present', 'compelling',
'evidence', 'that', 'such', 'proximity', 'solutions', 'are', 'unlikely', 'to',
'yield', 'significant', 'benefits', '.', 'Further', 'broader', 'impacts', 'for',
'the', 'work', 'include', 'data', 'sharing', 'capabilities', 'with', 'respect',
'to', 'temporal', 'characterizations', 'of', 'redundancy', 'across', 'mobile',
'devices', 'in', 'several', 'real-world', 'scenarios', '.']
```

2.1.1 2.2. Homogeneization

By looking at the tokenized corpus you may verify that there are many tokens that correspond to punktuation signs and other symbols that are not relevant to analyze the semantic content. They can be removed using the stemming or lemmatization tools from `nltk`.

The homogeneization process will consist of:

1. Removing capitalization: capital alphabetic characters will be transformed to their corresponding lowercase characters.
2. Removing non alphanumeric tokens (e.g. punktuation signs)
3. Stemming/Lemmatization: removing word terminations to preserve the root of the words and ignore grammatical information.

Exercise Convert all tokens in `tokenized_abstracts` to lowercase (using the `.lower()` method) and remove non alphanumeric tokens (that you can detect with `.isalnum()` method). You can complete the following code fragment with a single line of code ...

[14]:
```
filtered_abstracts = []

for n, abstract in enumerate(tokenized_abstracts):
    if not n%100:
        print('\rFiltering abstract', n, 'out of', nprojects, end='', flush=True)

    #<SOL>
    filtered_abstracts.append([el.lower() for el in abstract if el.isalnum()])
    #</SOL>

print('\n',filtered_abstracts[0])
```

```
Filtering abstract 24300 out of 24342
 ['the', 'past', 'few', 'years', 'have', 'seen', 'unprecedented', 'growth',
'in', 'mobile', 'data', 'consumption', 'powered', 'in', 'large', 'part', 'by',
'the', 'rapid', 'adoption', 'of', 'smart', 'phones', 'and', 'tablets', 'the',
'growth', 'in', 'wireless', 'data', 'creates', 'phenomenal', 'challenges',
'for', 'the', 'wireless', 'industry', 'which', 'has', 'been', 'unable', 'to',
'meet', 'the', 'demand', 'for', 'rich', 'mobile', 'content', 'through',
'cellular', 'networks', 'this', 'has', 'led', 'to', 'the', 'investigation',
'of', 'solutions', 'by', 'network', 'operators', 'that', 'aim', 'to', 'utilize',
'wifi', 'radios', 'present', 'in', 'these', 'mobile', 'devices', 'to',
'deliver', 'content', 'without', 'using', 'the', 'cellular', 'radio', 'links',
'also', 'known', 'as', 'approaches', 'aim', 'to', 'utilize', 'wifi',
'infrastructure', 'in', 'the', 'form', 'of', 'access', 'points', 'to',
'offload', 'this', 'content', 'but', 'these', 'have', 'various', 'deployment',
'issues', 'research', 'has', 'lately', 'focused', 'on', 'the', 'potential',
'of', 'peer', 'content', 'sharing', 'since', 'proximity', 'enables', 'high',
'speed', 'data', 'exchanges', 'which', 'in', 'turn', 'allows', 'mobile',
'devices', 'to', 'proactively', 'share', 'data', 'with', 'one', 'another', 'no',
'study', 'using', 'situations', 'has', 'established', 'the', 'potential', 'for',
'such', 'to', 'provide', 'capacity', 'gains', 'until', 'now', 'this',
'proposal', 'aims', 'to', 'conduct', 'solid', 'preliminary', 'pilot', 'studies',
'evaluating', 'several', 'of', 'the', 'foundational', 'claims', 'in', 'this',
'area', 'specifically', 'as', 'to', 'whether', 'sufficient', 'potential',
'exists', 'in', 'the', 'right', 'time', 'right', 'place', 'and', 'with',
'reasonably', 'viable', 'deployment', 'scenarios', 'the', 'proposed', 'work',
'will', 'gather', 'and', 'evaluate', 'pilot', 'data', 'in', 'both', 'highly',
'amenable', 'environments', 'wifi', 'across', 'multiple', 'tailgate',
'offerings', 'of', 'hundreds', 'of', 'users', 'as', 'well', 'as', 'more',
'challenging', 'environments', 'daily', 'commuter', 'trains', 'to', 'from',
'chicago', 'the', 'broader', 'impact', 'of', 'the', 'work', 'will', 'be', 'to',
'either', 'demonstrate', 'the', 'potential', 'viability', 'for', 'solutions',
'or', 'to', 'present', 'compelling', 'evidence', 'that', 'such', 'proximity',
'solutions', 'are', 'unlikely', 'to', 'yield', 'significant', 'benefits',
'further', 'broader', 'impacts', 'for', 'the', 'work', 'include', 'data',
'sharing', 'capabilities', 'with', 'respect', 'to', 'temporal',
'characterizations', 'of', 'redundancy', 'across', 'mobile', 'devices', 'in',
'several', 'scenarios']
```

2.1.2 2.2.1. Stemming vs Lemmatization

At this point, we can choose between applying a simple stemming or ussing lemmatization. We will try both to test their differences.

The lemmatizer from NLTK is based on WordNet. If you have not used wordnet before, you will likely need to download it from nltk (use the nltk.download() command)

[15]:
```python
stemmer = nltk.stem.SnowballStemmer('english')
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()

print('Result for the first abstract in dataset applying stemming')
print([stemmer.stem(el) for el in filtered_abstracts[0]])

print('Result for the first abstract in the dataset applying lemmatization')
print([wnl.lemmatize(el) for el in filtered_abstracts[0]])
```

```
Result for the first abstract in dataset applying stemming
['the', 'past', 'few', 'year', 'have', 'seen', 'unpreced', 'growth', 'in',
'mobil', 'data', 'consumpt', 'power', 'in', 'larg', 'part', 'by', 'the',
'rapid', 'adopt', 'of', 'smart', 'phone', 'and', 'tablet', 'the', 'growth',
'in', 'wireless', 'data', 'creat', 'phenomen', 'challeng', 'for', 'the',
'wireless', 'industri', 'which', 'has', 'been', 'unabl', 'to', 'meet', 'the',
'demand', 'for', 'rich', 'mobil', 'content', 'through', 'cellular', 'network',
'this', 'has', 'led', 'to', 'the', 'investig', 'of', 'solut', 'by', 'network',
'oper', 'that', 'aim', 'to', 'util', 'wifi', 'radio', 'present', 'in', 'these',
'mobil', 'devic', 'to', 'deliv', 'content', 'without', 'use', 'the', 'cellular',
'radio', 'link', 'also', 'known', 'as', 'approach', 'aim', 'to', 'util', 'wifi',
'infrastructur', 'in', 'the', 'form', 'of', 'access', 'point', 'to', 'offload',
'this', 'content', 'but', 'these', 'have', 'various', 'deploy', 'issu',
'research', 'has', 'late', 'focus', 'on', 'the', 'potenti', 'of', 'peer',
'content', 'share', 'sinc', 'proxim', 'enabl', 'high', 'speed', 'data',
'exchang', 'which', 'in', 'turn', 'allow', 'mobil', 'devic', 'to', 'proactiv',
'share', 'data', 'with', 'one', 'anoth', 'no', 'studi', 'use', 'situat', 'has',
'establish', 'the', 'potenti', 'for', 'such', 'to', 'provid', 'capac', 'gain',
'until', 'now', 'this', 'propos', 'aim', 'to', 'conduct', 'solid',
'preliminari', 'pilot', 'studi', 'evalu', 'sever', 'of', 'the', 'foundat',
'claim', 'in', 'this', 'area', 'specif', 'as', 'to', 'whether', 'suffici',
'potenti', 'exist', 'in', 'the', 'right', 'time', 'right', 'place', 'and',
'with', 'reason', 'viabl', 'deploy', 'scenario', 'the', 'propos', 'work',
'will', 'gather', 'and', 'evalu', 'pilot', 'data', 'in', 'both', 'high', 'amen',
'environ', 'wifi', 'across', 'multipl', 'tailgat', 'offer', 'of', 'hundr', 'of',
'user', 'as', 'well', 'as', 'more', 'challeng', 'environ', 'daili', 'commut',
'train', 'to', 'from', 'chicago', 'the', 'broader', 'impact', 'of', 'the',
'work', 'will', 'be', 'to', 'either', 'demonstr', 'the', 'potenti', 'viabil',
'for', 'solut', 'or', 'to', 'present', 'compel', 'evid', 'that', 'such',
'proxim', 'solut', 'are', 'unlik', 'to', 'yield', 'signific', 'benefit',
'further', 'broader', 'impact', 'for', 'the', 'work', 'includ', 'data', 'share',
'capabl', 'with', 'respect', 'to', 'tempor', 'character', 'of', 'redund',
'across', 'mobil', 'devic', 'in', 'sever', 'scenario']
Result for the first abstract in the dataset applying lemmatization
['the', 'past', 'few', 'year', 'have', 'seen', 'unprecedented', 'growth', 'in',
'mobile', 'data', 'consumption', 'powered', 'in', 'large', 'part', 'by', 'the',
'rapid', 'adoption', 'of', 'smart', 'phone', 'and', 'tablet', 'the', 'growth',
'in', 'wireless', 'data', 'creates', 'phenomenal', 'challenge', 'for', 'the',
'wireless', 'industry', 'which', 'ha', 'been', 'unable', 'to', 'meet', 'the',
'demand', 'for', 'rich', 'mobile', 'content', 'through', 'cellular', 'network',
'this', 'ha', 'led', 'to', 'the', 'investigation', 'of', 'solution', 'by',
'network', 'operator', 'that', 'aim', 'to', 'utilize', 'wifi', 'radio',
'present', 'in', 'these', 'mobile', 'device', 'to', 'deliver', 'content',
'without', 'using', 'the', 'cellular', 'radio', 'link', 'also', 'known', 'a',
'approach', 'aim', 'to', 'utilize', 'wifi', 'infrastructure', 'in', 'the',
'form', 'of', 'access', 'point', 'to', 'offload', 'this', 'content', 'but',
'these', 'have', 'various', 'deployment', 'issue', 'research', 'ha', 'lately',
'focused', 'on', 'the', 'potential', 'of', 'peer', 'content', 'sharing',
'since', 'proximity', 'enables', 'high', 'speed', 'data', 'exchange', 'which',
'in', 'turn', 'allows', 'mobile', 'device', 'to', 'proactively', 'share',
'data', 'with', 'one', 'another', 'no', 'study', 'using', 'situation', 'ha',
'established', 'the', 'potential', 'for', 'such', 'to', 'provide', 'capacity',
'gain', 'until', 'now', 'this', 'proposal', 'aim', 'to', 'conduct', 'solid',
'preliminary', 'pilot', 'study', 'evaluating', 'several', 'of', 'the',
'foundational', 'claim', 'in', 'this', 'area', 'specifically', 'a', 'to',
'whether', 'sufficient', 'potential', 'exists', 'in', 'the', 'right', 'time',
'right', 'place', 'and', 'with', 'reasonably', 'viable', 'deployment',
'scenario', 'the', 'proposed', 'work', 'will', 'gather', 'and', 'evaluate',
'pilot', 'data', 'in', 'both', 'highly', 'amenable', 'environment', 'wifi',
'across', 'multiple', 'tailgate', 'offering', 'of', 'hundred', 'of', 'user',
'a', 'well', 'a', 'more', 'challenging', 'environment', 'daily', 'commuter',
'train', 'to', 'from', 'chicago', 'the', 'broader', 'impact', 'of', 'the',
'work', 'will', 'be', 'to', 'either', 'demonstrate', 'the', 'potential',
'viability', 'for', 'solution', 'or', 'to', 'present', 'compelling', 'evidence',
'that', 'such', 'proximity', 'solution', 'are', 'unlikely', 'to', 'yield',
'significant', 'benefit', 'further', 'broader', 'impact', 'for', 'the', 'work',
'include', 'data', 'sharing', 'capability', 'with', 'respect', 'to', 'temporal',
'characterization', 'of', 'redundancy', 'across', 'mobile', 'device', 'in',
'several', 'scenario']
```

One of the advantages of the lemmatizer method is that the result of lemmmatization is still a true word, which is more advisable for the presentation of text processing results and lemmatization.

However, without using contextual information, lemmatize() does not remove grammatical differences. This is the reason why "is" or "are" are preserved and not replaced by infinitive "be".

As an alternative, we can apply .lemmatize(word, pos), where 'pos' is a string code specifying the part-of-speech (pos), i.e. the grammatical role of the words in its sentence. For instance, you can check the difference between `wnl.lemmatize('is')` and `wnl.lemmatize('is, pos='v')`.

Exercise Complete the following code fragment to lemmatize all abstracts in the NSF dataset

[16]:
```
lemmatized_abstracts = []

for n, abstract in enumerate(filtered_abstracts):
    if not n%100:
        print('\rLemmatizing abstract', n, 'out of', nprojects, end='',␣
↪flush=True)

    #<SOL>
    lemmatized_abstracts.append([wnl.lemmatize(el) for el in abstract])
    #</SOL>

print('Result for the first abstract in the dataset applying lemmatization')
print('\n',lemmatized_abstracts[0])
```

```
Lemmatizing abstract 24300 out of 24342Result for the first abstract in the
dataset applying lemmatization

 ['the', 'past', 'few', 'year', 'have', 'seen', 'unprecedented', 'growth', 'in',
'mobile', 'data', 'consumption', 'powered', 'in', 'large', 'part', 'by', 'the',
'rapid', 'adoption', 'of', 'smart', 'phone', 'and', 'tablet', 'the', 'growth',
'in', 'wireless', 'data', 'creates', 'phenomenal', 'challenge', 'for', 'the',
'wireless', 'industry', 'which', 'ha', 'been', 'unable', 'to', 'meet', 'the',
'demand', 'for', 'rich', 'mobile', 'content', 'through', 'cellular', 'network',
'this', 'ha', 'led', 'to', 'the', 'investigation', 'of', 'solution', 'by',
'network', 'operator', 'that', 'aim', 'to', 'utilize', 'wifi', 'radio',
'present', 'in', 'these', 'mobile', 'device', 'to', 'deliver', 'content',
'without', 'using', 'the', 'cellular', 'radio', 'link', 'also', 'known', 'a',
'approach', 'aim', 'to', 'utilize', 'wifi', 'infrastructure', 'in', 'the',
'form', 'of', 'access', 'point', 'to', 'offload', 'this', 'content', 'but',
'these', 'have', 'various', 'deployment', 'issue', 'research', 'ha', 'lately',
'focused', 'on', 'the', 'potential', 'of', 'peer', 'content', 'sharing',
'since', 'proximity', 'enables', 'high', 'speed', 'data', 'exchange', 'which',
'in', 'turn', 'allows', 'mobile', 'device', 'to', 'proactively', 'share',
'data', 'with', 'one', 'another', 'no', 'study', 'using', 'situation', 'ha',
'established', 'the', 'potential', 'for', 'such', 'to', 'provide', 'capacity',
'gain', 'until', 'now', 'this', 'proposal', 'aim', 'to', 'conduct', 'solid',
'preliminary', 'pilot', 'study', 'evaluating', 'several', 'of', 'the',
'foundational', 'claim', 'in', 'this', 'area', 'specifically', 'a', 'to',
'whether', 'sufficient', 'potential', 'exists', 'in', 'the', 'right', 'time',
'right', 'place', 'and', 'with', 'reasonably', 'viable', 'deployment',
'scenario', 'the', 'proposed', 'work', 'will', 'gather', 'and', 'evaluate',
'pilot', 'data', 'in', 'both', 'highly', 'amenable', 'environment', 'wifi',
'across', 'multiple', 'tailgate', 'offering', 'of', 'hundred', 'of', 'user',
'a', 'well', 'a', 'more', 'challenging', 'environment', 'daily', 'commuter',
'train', 'to', 'from', 'chicago', 'the', 'broader', 'impact', 'of', 'the',
'work', 'will', 'be', 'to', 'either', 'demonstrate', 'the', 'potential',
'viability', 'for', 'solution', 'or', 'to', 'present', 'compelling', 'evidence',
'that', 'such', 'proximity', 'solution', 'are', 'unlikely', 'to', 'yield',
'significant', 'benefit', 'further', 'broader', 'impact', 'for', 'the', 'work',
'include', 'data', 'sharing', 'capability', 'with', 'respect', 'to', 'temporal',
'characterization', 'of', 'redundancy', 'across', 'mobile', 'device', 'in',
'several', 'scenario']
```

2.2 2.3. Cleaning

The third step consists of removing those words that are very common in language and do not carry out usefull semantic content (articles, pronouns, etc).

Once again, we might need to load the stopword files using the download tools from `nltk`

Exercise In the second line below we read a list of common english stopwords. Clean `lemmatized_abstracts` by removing all tokens in the stopword list.

[17]:
```python
from nltk.corpus import stopwords
stopwords_en = stopwords.words('english')

clean_abstracts = []

for n, abstract in enumerate(lemmatized_abstracts):
    if not n%100:
        print('\rCleaning abstract', n, 'out of', nprojects, end='', flush=True)

    # Remove all tokens in the stopwords list and append the result to␣
 ↪clean_abstracts
    # <SOL>
    clean_tokens = [token for token in abstract if token not in stopwords_en]
    # </SOL>
    clean_abstracts.append(clean_tokens)

print('\n Let us check tokens after cleaning:')
print(clean_abstracts[0])
```

```
Cleaning abstract 24300 out of 24342
 Let us check tokens after cleaning:
['past', 'year', 'seen', 'unprecedented', 'growth', 'mobile', 'data',
'consumption', 'powered', 'large', 'part', 'rapid', 'adoption', 'smart',
'phone', 'tablet', 'growth', 'wireless', 'data', 'creates', 'phenomenal',
'challenge', 'wireless', 'industry', 'ha', 'unable', 'meet', 'demand', 'rich',
'mobile', 'content', 'cellular', 'network', 'ha', 'led', 'investigation',
'solution', 'network', 'operator', 'aim', 'utilize', 'wifi', 'radio', 'present',
'mobile', 'device', 'deliver', 'content', 'without', 'using', 'cellular',
'radio', 'link', 'also', 'known', 'approach', 'aim', 'utilize', 'wifi',
'infrastructure', 'form', 'access', 'point', 'offload', 'content', 'various',
'deployment', 'issue', 'research', 'ha', 'lately', 'focused', 'potential',
'peer', 'content', 'sharing', 'since', 'proximity', 'enables', 'high', 'speed',
'data', 'exchange', 'turn', 'allows', 'mobile', 'device', 'proactively',
'share', 'data', 'one', 'another', 'study', 'using', 'situation', 'ha',
'established', 'potential', 'provide', 'capacity', 'gain', 'proposal', 'aim',
'conduct', 'solid', 'preliminary', 'pilot', 'study', 'evaluating', 'several',
'foundational', 'claim', 'area', 'specifically', 'whether', 'sufficient',
'potential', 'exists', 'right', 'time', 'right', 'place', 'reasonably',
'viable', 'deployment', 'scenario', 'proposed', 'work', 'gather', 'evaluate',
'pilot', 'data', 'highly', 'amenable', 'environment', 'wifi', 'across',
'multiple', 'tailgate', 'offering', 'hundred', 'user', 'well', 'challenging',
'environment', 'daily', 'commuter', 'train', 'chicago', 'broader', 'impact',
'work', 'either', 'demonstrate', 'potential', 'viability', 'solution',
'present', 'compelling', 'evidence', 'proximity', 'solution', 'unlikely',
'yield', 'significant', 'benefit', 'broader', 'impact', 'work', 'include',
'data', 'sharing', 'capability', 'respect', 'temporal', 'characterization',
'redundancy', 'across', 'mobile', 'device', 'several', 'scenario']
```

2.3 2.4. Vectorization

Up to this point, we have transformed the raw text collection of articles in a list of articles, where each article is a collection of the word roots that are most relevant for semantic analysis. Now, we need to convert these data (a list of token lists) into a numerical representation (a list of vectors, or a matrix). To do so, we will start using the tools provided by the `gensim` library.

As a first step, we create a dictionary containing all tokens in our text corpus, and assigning an integer identifier to each one of them.

[18]:
```
import gensim

# Create dictionary of tokens
D = gensim.corpora.Dictionary(clean_abstracts)
n_tokens = len(D)
```

```
print('The dictionary contains', n_tokens, 'terms')
print('First terms in the dictionary:')
for n in range(10):
    print(str(n), ':', D[n])
```

```
The dictionary contains 60688 terms
First terms in the dictionary:
0 : rapid
1 : scenario
2 : several
3 : approach
4 : using
5 : point
6 : solid
7 : creates
8 : potential
9 : offload
```

We can also filter out terms that appear in too few or too many of the documents in the dataset:

[19]:
```
no_below = 5 #Minimum number of documents to keep a term in the dictionary
no_above = .75 #Maximum proportion of documents in which a term can appear to be␣
 ↪kept in the dictionary

D.filter_extremes(no_below=no_below,no_above=no_above, keep_n=25000)
n_tokens = len(D)

print('The dictionary contains', n_tokens, 'terms')

print('First terms in the dictionary:')
for n in range(10):
    print(str(n), ':', D[n])
```

```
The dictionary contains 20044 terms
First terms in the dictionary:
0 : factory
1 : folded
2 : fan
3 : occasionally
4 : subducted
5 : longitudinal
6 : horse
7 : northwest
8 : potential
9 : superlattices
```

In the second step, let us create a numerical version of our corpus using the `doc2bow` method. In general, `D.doc2bow(token_list)` transforms any list of tokens into a list of tuples `(token_id, n)`, one per each token in `token_list`, where `token_id` is the token identifier (according to dictionary `D`) and `n` is the number of occurrences of such token in `token_list`.

[20]:
```
corpus_bow = [D.doc2bow(doc) for doc in clean_abstracts]
```

At this point, it is good to make sure to understand what has happened. In `clean_abstracts` we had a list of token lists. With it, we have constructed a Dictionary, `D`, which assigns an integer identifier to each token in the corpus. After that, we have transformed each article (in `clean_abstracts`) in a list tuples `(id, n)`.

[21]:
```
print('Original article (after cleaning):')
print(clean_abstracts[0])
print('Sparse vector representation (first 10 components):')
print(corpus_bow[0][:10])
print('Word counts for the first project (first 10 components):')
print(list(map(lambda x: (D[x[0]], x[1]), corpus_bow[0][:10])))
```

```
Original article (after cleaning):
['past', 'year', 'seen', 'unprecedented', 'growth', 'mobile', 'data',
'consumption', 'powered', 'large', 'part', 'rapid', 'adoption', 'smart',
'phone', 'tablet', 'growth', 'wireless', 'data', 'creates', 'phenomenal',
'challenge', 'wireless', 'industry', 'ha', 'unable', 'meet', 'demand', 'rich',
'mobile', 'content', 'cellular', 'network', 'ha', 'led', 'investigation',
'solution', 'network', 'operator', 'aim', 'utilize', 'wifi', 'radio', 'present',
'mobile', 'device', 'deliver', 'content', 'without', 'using', 'cellular',
'radio', 'link', 'also', 'known', 'approach', 'aim', 'utilize', 'wifi',
'infrastructure', 'form', 'access', 'point', 'offload', 'content', 'various',
'deployment', 'issue', 'research', 'ha', 'lately', 'focused', 'potential',
'peer', 'content', 'sharing', 'since', 'proximity', 'enables', 'high', 'speed',
'data', 'exchange', 'turn', 'allows', 'mobile', 'device', 'proactively',
'share', 'data', 'one', 'another', 'study', 'using', 'situation', 'ha',
'established', 'potential', 'provide', 'capacity', 'gain', 'proposal', 'aim',
'conduct', 'solid', 'preliminary', 'pilot', 'study', 'evaluating', 'several',
'foundational', 'claim', 'area', 'specifically', 'whether', 'sufficient',
'potential', 'exists', 'right', 'time', 'right', 'place', 'reasonably',
'viable', 'deployment', 'scenario', 'proposed', 'work', 'gather', 'evaluate',
'pilot', 'data', 'highly', 'amenable', 'environment', 'wifi', 'across',
'multiple', 'tailgate', 'offering', 'hundred', 'user', 'well', 'challenging',
'environment', 'daily', 'commuter', 'train', 'chicago', 'broader', 'impact',
'work', 'either', 'demonstrate', 'potential', 'viability', 'solution',
'present', 'compelling', 'evidence', 'proximity', 'solution', 'unlikely',
'yield', 'significant', 'benefit', 'broader', 'impact', 'work', 'include',
'data', 'sharing', 'capability', 'respect', 'temporal', 'characterization',
'redundancy', 'across', 'mobile', 'device', 'several', 'scenario']
Sparse vector representation (first 10 components):
[(8, 4), (398, 2), (764, 1), (862, 1), (911, 1), (1017, 1), (1138, 1), (1308,
4), (1310, 1), (1809, 1)]
Word counts for the first project (first 10 components):
[('potential', 4), ('radio', 2), ('gain', 1), ('consumption', 1), ('seen', 1),
('offload', 1), ('link', 1), ('ha', 4), ('gather', 1), ('preliminary', 1)]
```

Note that we can interpret each element of corpus_bow as a `sparse_vector`. For example, a list of tuples

```
[(0, 1), (3, 3), (5,2)]
```

for a dictionary of 10 elements can be represented as a vector, where any tuple `(id, n)` states that position `id` must take value `n`. The rest of positions must be zero.

```
[1, 0, 0, 3, 0, 2, 0, 0, 0, 0]
```

These sparse vectors will be the inputs to the topic modeling algorithms.
As a summary, the following variables will be relevant for the next chapters:

- `D`: A gensim dictionary. Term strings can be accessed using the numeric identifiers. For instance, `D[0]` contains the string corresponding to the first position in the BoW representation.
- `corpus_bow`: BoW corpus. A list containing an entry per project in the dataset, and consisting of the (sparse) BoW representation for the abstract of that project.
- `NSF_data`: A list containing an entry per project in the dataset, and consisting of metadata for the projects in the dataset

The way we have constructed the `corpus_bow` variable guarantees that the order is preserved, so that the projects are listed in the same order in the lists `corpus_bow` and `NSF_data`.

2.4 2.5. Dictionary properties

In the following code fragment, we build a list `all_counts` that contains tuples (terms, document_counts). You can use this list to calculate some statistics about the vocabulary of the dataset

[22]:
```
all_counts = [(D[el], D.dfs[el]) for el in D.dfs]
all_counts = sorted(all_counts, key=lambda x: x[1])
```

3 3. Topic Modeling

3.1 3.1. Training a topic model using Gensim LDA

Since we already have computed the dictionary and documents BoW representation using Gensim, computing the topic model is straightforward using the `LdaModel()` function. Please, refer to Gensim API documentation for more information on the different parameters accepted by the function:

[32]:
```
import gensim
num_topics = 50

ldag = gensim.models.ldamodel.LdaModel(corpus=corpus_bow, id2word=D,␣
 ↪num_topics=num_topics)
```

3.2 3.2. LDA model visualization

Gensim provides a basic visualization of the obtained topics:

[33]:
```
ldag.print_topics(num_topics=-1, num_words=10)
```

[33]:
```
[(0,
  '0.016*material + 0.014*stress + 0.011*structure + 0.011*strength +
0.010*mechanical + 0.009*3d + 0.009*composite + 0.009*model + 0.009*bone +
0.008*property'),
 (1,
  '0.016*game + 0.012*model + 0.011*design + 0.011*system + 0.011*decision +
0.011*financial + 0.010*transportation + 0.008*mechanism + 0.007*uncertainty +
0.006*theory'),
 (2,
  '0.016*star + 0.012*galaxy + 0.010*data + 0.009*model + 0.009*science +
0.008*astronomy + 0.007*simulation + 0.007*observation + 0.006*new +
0.006*team'),
 (3,
  '0.033*fellowship + 0.026*fellow + 0.020*postdoctoral + 0.020*host +
0.020*science + 0.018*scientist + 0.017*university + 0.015*axiom +
0.014*training + 0.013*institution'),
 (4,
  '0.024*change + 0.020*climate + 0.011*model + 0.010*region +
0.007*understanding + 0.006*impact + 0.006*community + 0.006*temperature +
0.005*effect + 0.005*study'),
 (5,
  '0.090*water + 0.018*urban + 0.014*city + 0.014*environmental + 0.014*arctic +
0.014*sustainability + 0.013*system + 0.012*management + 0.011*natural +
0.009*resource'),
 (6,
  '0.030*mesa + 0.029*patient + 0.015*hospital + 0.013*eruption + 0.012*volcano
+ 0.011*care + 0.010*volcanic + 0.009*injury + 0.008*aging + 0.007*concise'),
 (7,
  '0.022*animal + 0.013*evolution + 0.012*student + 0.010*study +
0.007*researcher + 0.007*group + 0.006*color + 0.006*food + 0.006*also +
0.006*origin'),
 (8,
  '0.023*student + 0.023*stem + 0.013*program + 0.013*faculty + 0.012*education
+ 0.011*engineering + 0.011*university + 0.010*community + 0.010*study +
0.009*college'),
 (9,
  '0.024*energy + 0.017*cost + 0.015*technology + 0.014*control + 0.013*power +
0.012*process + 0.010*market + 0.010*potential + 0.008*product + 0.008*phase'),
 (10,
  '0.010*sample + 0.009*island + 0.009*support + 0.009*instrumentation +
0.008*seafloor + 0.008*vessel + 0.008*pacific + 0.008*madagascar + 0.008*day +
0.007*nsf'),
 (11,
  '0.022*social + 0.017*behavior + 0.012*human + 0.010*individual + 0.010*people
+ 0.007*understanding + 0.007*cognitive + 0.007*behavioral + 0.007*decision +
0.007*system'),
 (12,
  '0.036*worker + 0.026*labor + 0.025*firm + 0.020*uas + 0.014*georgetown +
0.012*market + 0.012*cyberspace + 0.012*farmer + 0.012*app +
0.009*productivity'),
 (13,
  '0.018*science + 0.016*university + 0.016*program + 0.011*state + 0.011*site +
0.011*national + 0.010*center + 0.008*community + 0.008*activity +
0.008*graduate'),
 (14,
  '0.019*particle + 0.017*system + 0.011*physic + 0.010*measurement +
0.009*experiment + 0.008*light + 0.008*field + 0.007*new + 0.007*matter +
0.006*state'),
 (15,
  '0.028*ci + 0.021*grass + 0.017*uc + 0.016*migration + 0.016*zealand + 0.016*l
+ 0.013*hail + 0.009*mr + 0.008*santa + 0.008*road'),
 (16,
  '0.051*quantum + 0.020*physic + 0.015*topological + 0.013*theory +
0.012*structure + 0.012*new + 0.011*state + 0.010*computational + 0.010*study +
0.010*pi'),
 (17,
  '0.026*cell + 0.020*virus + 0.015*disease + 0.013*infection + 0.010*pathogen +
0.008*bacteria + 0.007*bacterial + 0.007*interaction + 0.006*study +
0.006*viral'),
 (18,
  '0.061*wave + 0.023*stream + 0.014*acoustic + 0.010*phosphorus +
0.008*antarctica + 0.007*aquatic + 0.007*black + 0.007*nitrogen + 0.007*station
+ 0.006*source'),
 (19,
  '0.027*conference + 0.027*student + 0.021*workshop + 0.015*researcher +
0.011*field + 0.010*support + 0.010*meeting + 0.010*opportunity +
0.010*participant + 0.009*science'),
 (20,
  '0.033*household + 0.016*limb + 0.015*sm + 0.011*atlas + 0.011*turnover +
0.011*bubble + 0.011*sliding + 0.010*aggregation + 0.009*surfactant +
0.007*gait'),
 (21,
  '0.031*specie + 0.012*population + 0.011*genetic + 0.010*evolutionary +
0.008*evolution + 0.008*trait + 0.007*ecological + 0.007*diversity + 0.007*study
+ 0.006*variation'),
 (22,
  '0.020*drug + 0.018*cancer + 0.014*treatment + 0.011*disease + 0.010*screening
+ 0.010*technology + 0.009*delivery + 0.008*potential + 0.008*cell +
0.008*method'),
 (23,
  '0.023*theory + 0.022*problem + 0.016*algorithm + 0.014*network + 0.010*group
+ 0.010*application + 0.009*new + 0.008*pi + 0.008*study + 0.008*graph'),
 (24,
  '0.024*software + 0.017*computing + 0.011*application + 0.010*technology +
0.010*system + 0.009*design + 0.009*platform + 0.009*tool + 0.009*performance +
0.008*user'),
 (25,
  '0.031*cell + 0.019*protein + 0.012*biological + 0.011*gene + 0.011*biology +
0.009*function + 0.009*mechanism + 0.008*molecular + 0.008*cellular +
0.007*dna'),
 (26,
  '0.016*timing + 0.014*detector + 0.011*groundwater + 0.008*underground +
0.008*richard + 0.008*gr + 0.007*particle + 0.007*np + 0.007*remediation +
0.006*axial'),
 (27,
  '0.022*security + 0.017*risk + 0.015*system + 0.012*safety + 0.011*vehicle +
0.011*hazard + 0.011*privacy + 0.010*attack + 0.008*building + 0.007*damage'),
 (28,
  '0.020*ocean + 0.014*seismic + 0.012*earth + 0.009*mantle + 0.009*process +
0.008*surface + 0.008*study + 0.008*ice + 0.007*water + 0.006*sediment'),
 (29,
  '0.021*molecule + 0.011*instrumentation + 0.010*new + 0.010*chemistry +
0.009*instrument + 0.008*organic + 0.008*radical + 0.008*molecular +
0.007*student + 0.007*chemical'),
 (30,
  '0.042*student + 0.021*engineering + 0.020*learning + 0.016*science +
0.015*education + 0.015*teacher + 0.015*program + 0.012*school + 0.010*stem +
0.009*development'),
 (31,
  '0.024*fluid + 0.013*flow + 0.011*system + 0.010*dynamic + 0.009*model +
0.008*effect + 0.008*thermal + 0.007*phenomenon + 0.007*behavior + 0.006*wave'),
 (32,
  '0.055*system + 0.015*design + 0.014*memory + 0.013*power + 0.011*energy +
0.008*storage + 0.008*wastewater + 0.008*performance + 0.008*security +
0.007*approach'),
 (33,
  '0.075*earthquake + 0.043*fault + 0.017*plate + 0.015*seismicity +
0.014*deformation + 0.012*oklahoma + 0.010*rock + 0.010*event + 0.010*induced +
0.009*zone'),
 (34,
  '0.054*data + 0.013*information + 0.012*network + 0.009*system + 0.008*user +
0.008*community + 0.008*new + 0.006*service + 0.006*infrastructure +
0.005*analysis'),
 (35,
  '0.017*imaging + 0.013*sensor + 0.010*tissue + 0.008*image + 0.008*medical +
0.007*new + 0.006*patient + 0.006*technique + 0.006*technology + 0.006*system'),
 (36,
  '0.020*equation + 0.018*problem + 0.018*model + 0.016*mathematical +
0.014*theory + 0.011*solution + 0.011*study + 0.011*geometry + 0.010*system +
0.010*space'),
 (37,
  '0.036*magma + 0.026*prison + 0.015*inmate + 0.015*hydrothermal +
0.014*magmatic + 0.012*minnesota + 0.011*thereof + 0.010*wisconsin +
0.009*immersion + 0.009*cl'),
 (38,
  '0.016*robot + 0.015*security + 0.014*system + 0.014*infrastructure +
0.013*cloud + 0.010*design + 0.010*building + 0.009*robotics + 0.009*application
+ 0.009*service'),
 (39,
  '0.050*material + 0.015*device + 0.012*property + 0.010*manufacturing +
0.010*new + 0.009*high + 0.008*application + 0.006*student + 0.006*energy +
0.006*surface'),
 (40,
  '0.032*plant + 0.021*soil + 0.015*carbon + 0.013*ecosystem + 0.010*microbial +
0.010*nutrient + 0.009*community + 0.009*nitrogen + 0.008*crop +
0.008*production'),
 (41,
  '0.019*polymer + 0.018*material + 0.013*molecular + 0.013*property +
0.010*structure + 0.010*membrane + 0.010*cell + 0.009*mechanical + 0.008*student
+ 0.007*simulation'),
 (42,
  '0.014*study + 0.011*human + 0.009*question + 0.009*history + 0.009*collection
+ 0.008*data + 0.008*group + 0.007*analysis + 0.006*volunteer + 0.006*fossil'),
 (43,
  '0.019*system + 0.017*device + 0.014*wireless + 0.014*communication +
0.013*network + 0.011*design + 0.010*technology + 0.010*mobile + 0.009*spectrum
+ 0.009*application'),
 (44,
  '0.036*solar + 0.017*wind + 0.016*energy + 0.010*field + 0.009*radar +
0.009*tandem + 0.009*weather + 0.007*storm + 0.007*magnetic + 0.007*radiation'),
 (45,
  '0.015*science + 0.015*geoscience + 0.013*thunderstorm + 0.011*virtual +
0.008*geosciences + 0.008*community + 0.007*student + 0.005*development +
0.005*tribal + 0.005*field'),
 (46,
  '0.058*battery + 0.053*ion + 0.022*electrochemical + 0.018*energy +
0.017*electrode + 0.016*patent + 0.011*membrane + 0.011*electrolyte +
0.010*channel + 0.010*union'),
 (47,
  '0.035*chemical + 0.026*chemistry + 0.025*reaction + 0.014*catalyst +
0.014*synthesis + 0.014*metal + 0.013*student + 0.012*new + 0.012*professor +
0.009*program'),
 (48,
  '0.052*data + 0.028*method + 0.018*model + 0.018*analysis + 0.016*statistical
+ 0.009*computational + 0.009*new + 0.009*approach + 0.008*develop +
0.008*problem'),
 (49,
  '0.039*language + 0.024*brain + 0.013*visual + 0.011*neural + 0.008*learning +
0.008*neuroscience + 0.007*speech + 0.006*understanding + 0.006*system +
0.006*processing')]
```

A more useful visualization is provided by the python LDA visualization library, pyLDAvis. Before executing the next code fragment you need to install pyLDAvis:

```
>> pip install (--user) pyLDAvis
```

[34]:
```
import pyLDAvis.gensim as gensimvis
import pyLDAvis

vis_data = gensimvis.prepare(ldag, corpus_bow, D)
pyLDAvis.display(vis_data)
```

[34]:
```
<IPython.core.display.HTML object>
```

3.3 3.3. Gensim utility functions

In addition to visualization purposes, topic models are useful to obtain a semantic representation of documents that can later be used with some other purpose:

- In document classification problems
- In content-based recommendations systems

Essentially, the idea is that the topic model provides a (semantic) vector representation of documents, and use probability divergences to measure document similarity. The following functions of the `LdaModel` class will be useful in this context:

- `get_topic_terms(topic_id)`: Gets vector of the probability distribution among words for the indicated topic
- `get_document_topics(bow_vector)`: Gets (sparse) vector with the probability distribution among topics for the provided document

[39]:
```
ldag.get_topic_terms(topicid=0)
```

[39]:
```
[(15981, 0.015552882204555873),
 (10344, 0.013689160653796051),
 (10230, 0.011331027351944686),
 (7414, 0.010792993112945853),
 (18721, 0.009924017367650786 8),
 (8780, 0.009466792985872529 4),
 (3972, 0.009455010358413288 4),
 (3414, 0.009247652165986004 8),
 (16783, 0.008704692461959583 4),
 (6706, 0.008329090677586895 4)]
```

[40]:
```
ldag.get_document_topics(corpus_bow[0])
```

[40]:
```
[(24, 0.07445651416828809З),
 (32, 0.027490762943560187),
 (34, 0.43543936926992211),
 (42, 0.04565237041214714),
 (43, 0.41198860751547461)]
```

An alternative to the use of the `get_document_topics()` function is to directly transform a dataset using the `ldag` object as follows. You can apply this transformation to several documents at once, but then the result is an iterator from which you can build the corresponding list if necessary

[43]:
```
print(ldag[corpus_bow[0]])

print('When applied to a dataset it will provide an iterator')
print(ldag[corpus_bow[:3]])

print('We can rebuild the list from the iterator with a one liner')
print([el for el in ldag[corpus_bow[:3]]])
```

```
[(24, 0.075525636160123699), (32, 0.027751868561855377), (34,
0.42373872312125038), (42, 0.046070086119450258), (43, 0.41381195825026962)]
When applied to a dataset it will provide an iterator
<gensim.interfaces.TransformedCorpus object at 0x182b1c0b8>
We can rebuild the list from the iterator with a one liner
[[(24, 0.075675426482761668), (32, 0.027745756818862509), (34,
0.42208917739907476), (42, 0.046038033129709763), (43, 0.41392481534299502)],
[(9, 0.014832505935994967), (17, 0.028913783533470343), (19,
0.013635057248558577), (22, 0.067683787981276), (27, 0.081962654285507031), (31,
0.040019197258354б5), (35, 0.20869134724581406), (39, 0.42112582764087453), (43,
0.063422989822178З8), (46, 0.011968210084544127), (47, 0.014080681895500964)],
[(16, 0.025536399440439325), (17, 0.017797843390639168), (24,
0.19674931222987854), (38, 0.75071644493904355)]]
```

Finally, Gensim provides some useful functions to convert between formats, and to simplify interaction with numpy and scipy. The following code fragment converts a corpus in sparse format to a full numpy matrix

[59]:
```
reduced_corpus = [el for el in ldag[corpus_bow[:3]]]
reduced_corpus = gensim.matutils.corpus2dense(reduced_corpus, num_topics).T
print(reduced_corpus)
```

```
[[ 0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.07605597  0.          0.          0.
   0.          0.          0.01109915  0.          0.02766871  0.
   0.4203077   0.          0.          0.          0.          0.          0.
   0.          0.04601504  0.41399154  0.          0.          0.          0.
   0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.          0.
```

```
  0.          0.          0.01482547  0.          0.          0.          0.
  0.          0.          0.          0.02891483  0.          0.01363522
  0.          0.          0.06768388  0.          0.          0.          0.
  0.0819626   0.          0.          0.          0.04001787  0.          0.
  0.          0.20869374  0.          0.          0.          0.42116106
  0.          0.          0.          0.06342395  0.          0.
  0.01196972  0.01404717  0.          0.        ]
 [ 0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.02553238  0.01779855  0.          0.          0.
  0.          0.          0.          0.19676815  0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.75070095  0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.        ]]
```

Exercise Build a function that returns the most relevant projects for a given topic

[80]:
```
def most_relevant_projects(ldag, topicid, corpus_bow, nprojects=10):
    """This function returns the most relevant projects in corpus_bow

    : ldag: The trained topic model object provided by gensim
    : topicid: The topic for which we want to find the most relevant documents
    : corpus_bow: The BoW representation of documents in Gensim format
    : nprojects: Number of most relevant projects to identify

    : Returns: A list with the identifiers of the most relevant projects
    """

    print('Computing most relevant projects for Topic', topicid)
    print('Topic composition is:')
    print(ldag.show_topic(topicid))

    #<SOL>
    document_topic = [el for el in ldag[corpus_bow]]
    document_topic = gensim.matutils.corpus2dense(document_topic, ldag.
 ↪num_topics).T
    return np.argsort(document_topic[:,topicid])[::-1][:nprojects].tolist()
    #</SOL>

#To test the function we will find the most relevant projects for a subset of␣
 ↪the NSF dataset
project_id = most_relevant_projects(ldag, 17, corpus_bow[:10000])

#Print titles of selected projects
for idproject in project_id:
    print(NSF_data[idproject]['title'])
```

```
Computing most relevant projects for Topic 17
Topic composition is:
[('cell', 0.025947031075878084), ('virus', 0.019792277823564915), ('disease',
0.015158345097956067), ('infection', 0.013321907140761176), ('pathogen',
0.0096157042120448202), ('bacteria', 0.0084459189411148199), ('bacterial',
0.0074083118360803767), ('interaction', 0.0070722382090914818), ('study',
0.006189777718159286), ('viral', 0.0055450859431854897)]
RAPID:  On-site Disinfection and Survival of Ebola and Other Viruses in Human
Fecal Wastes and Sewage
Dissertation Research: Disentangling a Potential Tri-Kingdom Mutualism in the
Guiana Shield
RAPID: Tackling Critical Issues in the Ebola Epidemic through Modeling: Viral
Evolution
RAPID Proposal: Ebola virus stability in the environment - Implications for
outbreak control
RAPID: Understanding and leveraging asymptomatic infections to control Ebola
virus transmission
RAPID: Collaborative Research: Survival of Ebolavirus in the Water Environment:
Surrogate Development and Disinfection Effectiveness
RAPID: Collaborative Research: Survival of Ebolavirus in the Water Environment:
Surrogate Development and Disinfection Effectiveness
RAPID: Ebola virus population structure, genetic diversity and evolution
Collaborative Research: Modeling Immune Dynamics of RNA Viruses In Reservoir and
Nonreservoir Species
Collaborative Research: Modeling Immune Dynamics of RNA Viruses In Reservoir and
Nonreservoir Species
```

Exercise Build a function that computes the semantic distance between two documents. For this, you can use the functions (or code fragments) provided in the library `dist_utils.py`.

[ ]:
```
def pairwase_dist(doc1, doc2):
    """This function returns the Jensen-Shannon
    distance between the corresponding vectors of the documents

    : doc1: Semantic representation for the doc1 (a vector of length ntopics)
    : doc2: Semantic representation for the doc2 (a vector of length ntopics)

    : Returns: The JS distance between doc1 and doc2 (a number)
    """
    #<SOL>
    #</SOL>
```

3.3.1 Function that creates the Node CSV file for Gephi

[19]:
```
#print(NSF_data[0].keys())
#print(NSF_data[0]['institution'])
```

```python
def strNone(str_to_convert):
    if str_to_convert is None:
        return ''
    else:
        return str_to_convert

with open('NSF_nodes.csv','w') as fout:
    fout.write('Id;Title;Year;Budget;UnivName;UnivZIP;State\n')
    for project in NSF_data:
        fout.write(project['project_code']+';'+project['title']+';')
        fout.write(project['year']+';'+str(project['budget'])+';')
        fout.write(project['institution'][0]+';')
        fout.write(strNone(project['institution'][1])+';')
        fout.write(strNone(project['institution'][2])+'\n')
```