handshake_takehome

January 21, 2022

1 Basic data Analysis

[1]:
```
!pip install fuzzywuzzy
!pip install plotly
!pip install gensim
!pip install webcolors
```

```
Collecting fuzzywuzzy
  Using cached fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
Collecting plotly
  Downloading plotly-5.5.0-py2.py3-none-any.whl (26.5 MB)
     |                  | 26.5 MB 39.5 MB/s eta 0:00:01
Collecting tenacity>=6.2.0
  Downloading tenacity-8.0.1-py3-none-any.whl (24 kB)
Requirement already satisfied: six in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from plotly) (1.15.0)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.5.0 tenacity-8.0.1
Collecting gensim
  Downloading gensim-4.1.2-cp38-cp38-macosx_10_9_x86_64.whl (24.0 MB)
     |                  | 24.0 MB 23 kB/s  eta 0:00:01
Requirement already satisfied: smart-open>=1.8.1 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from gensim) (3.0.0)
Requirement already satisfied: scipy>=0.18.1 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from gensim) (1.5.2)
Requirement already satisfied: numpy>=1.17.0 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from gensim) (1.21.4)
Requirement already satisfied: requests in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from smart-
open>=1.8.1->gensim) (2.24.0)
Requirement already satisfied: idna<3,>=2.5 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from requests->smart-
open>=1.8.1->gensim) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from requests->smart-
open>=1.8.1->gensim) (3.0.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from requests->smart-
open>=1.8.1->gensim) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from requests->smart-
open>=1.8.1->gensim) (1.25.11)
Installing collected packages: gensim
Successfully installed gensim-4.1.2
Collecting webcolors
  Downloading webcolors-1.11.1-py3-none-any.whl (9.9 kB)
Installing collected packages: webcolors
Successfully installed webcolors-1.11.1
```

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
import plotly.express as px
```

[3]:
```
majors = pd.read_csv(r"majors.csv")
users = pd.read_csv(r"users.csv")
```

[4]:
```
majors.head()
```

[4]:

	MAJOR_ID	MAJOR_NAME
0	fafc74b2-2815-4bef-b011-c8a37e5073c2	Chemical Engineering
1	a731d3b9-ec07-4d09-83fd-ed443bd2b731	Undergraduate Pathway
2	61aa9457-4f86-4954-86d4-b855167fd34b	Pre-General Engineering
3	1b3bff54-b2bb-4fa3-bc8e-a8d525097c17	Social Work
4	47402f29-939a-4d31-bea3-765581941d55	Accounting

[5]:
```
majors.MAJOR_NAME.value_counts()
```

[5]:
```
Clinical Sci:Medical Tech Cert                                         1
Coaching                                                               1
IntSt-ApHumBeh (BA)                                                    1
Aeronautical Management Technology (Unmanned Aerial Systems)           1
Biology Biomedical Option                                              1
                                                                      ..
Latin Am Carib&US Latino (MA)                                          1
Business Leadership                                                    1
Marriage and Family Therapy MS                                         1
Comb St in Early Child & Sp Ed                                         1
Commercial Entrepreneurship                                            1
Name: MAJOR_NAME, Length: 14726, dtype: int64
```

1.0.1 Note: There are a lot distinct majors which is also mentioned on the problem statement

Let's analyze these names

[6]:
```
print(f"There are total {majors.MAJOR_NAME.nunique()} distinct names")
```

```
There are total 14726 distinct names
```

[7]:
```
majors.MAJOR_NAME.sort_values().unique().tolist()[:20]
```

[7]:
```
[' Applied Legal Studies (BA, BS)',
 ' International Relations',
 '(OLD) Comm Sci & Disorders',
 '(OLD) Int Studies:Intl Affair',
 '*IT Applications Dev Opt',
 '*IT Bus/Systems Analysis Opt',
 '-',
 '3-2 Engineering',
 '3D Animation and Game Design',
 '3D Digital Design',
 '4+1 Undergraduate Engineering',
 'A&S - Open Option (XXAS)',
 'A&S 3-3 Law Program',
 'A&S Business-Dual',
 'A&S Non-Degree',
 'A&S/Business Dual',
 'AA Degree',
 'ACCT-ADL',
 'ACCT-BSBA',
 'ACCT-MS']
```

1.0.2 Note:

There are a lot of names such as 'AA Degree','ACCT-ADL', 'ACCT-BSBA', 'ACCT-MS' etc. which does not have a full form. Given a chance I would love to know those in order to formulate better groups. Here, for simplicity I would group those others or have sub-groups of others such as "Others-MS", "Others-BS" etc.

[8]:
```
# majors["major_group_id"]=

# df['PETALS'] = df['BLOOM'].str.extract('(\\(.*?)\\)', expand=False).str.
 ↪strip()
no_acc = majors.MAJOR_NAME.str.extract(r'([A-Z]+\-+[A-Z]+)+', expand=False).
 ↪nunique()
print(f"There are {no_acc} accronyms of majors")
```

```
There are 469 accronyms of majors
```

[9]:
```
majors["major_accronyms"]= majors.MAJOR_NAME.str.extract(r'([A-Z]+\-+[A-Z]+)+',␣
↪expand=True)
majors["major_accronyms"].dropna().head(20)
```

[9]:
```
7         MICR-PHD
32         HIST-BA
54         FINE-BA
59        PHTR-DPT
109      NURS-BSNU
110        INBU-MS
116        BIOL-BS
121        COUN-MA
122    PHRD-PHRMD
142        CHEM-BS
144        PSYC-BA
166       FINE-BFA
167       EDHD-PHD
172      MGMT-BSBA
175        GEOS-MA
176       PADM-MPA
182        EVSC-MS
193        COMM-BA
220        GEOG-BA
221       SOCI-MIN
Name: major_accronyms, dtype: object
```

1.1 Now let's group those names

1.1.1 Approach

There are endless possibilities to group these names. Some of the ways I could think of are- - 1. Looking into similar names using keywords. But it will not scale well for larger applications and will be tidious for 14,726 distinct names - 2. Grouping the names using unsupervised models such as k-means clustering. But it has to go through NLP pipeline to take similar names into account - 3. Find cosine similarity using bag of words and then group based on some threshold - 4. Topic modelling using LDA and then cluster using k-means. This would scale up really well and can take simlar names based on their stems or root into account. FYI - This is something I worked in one my previous projects(with my Ex-employer)

I will try approach 4 since it is the most scalable solution here

[10]:
```
!pip install nltk
#Libraries for preprocessing
from gensim.parsing.preprocessing import remove_stopwords
import string
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import webcolors
```

```
#Download once if using NLTK for preprocessing
import nltk
nltk.download('punkt')

#Libraries for vectorisation
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.model_selection import GridSearchCV
from fuzzywuzzy import fuzz

#Libraries for clustering
from sklearn.cluster import KMeans
```

```
Requirement already satisfied: nltk in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (3.5)
Requirement already satisfied: joblib in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from nltk) (0.17.0)
Requirement already satisfied: click in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from nltk) (7.1.2)
Requirement already satisfied: tqdm in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from nltk) (4.50.2)
Requirement already satisfied: regex in
/opt/homebrew/anaconda3/lib/python3.8/site-packages (from nltk) (2020.10.15)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/sukanyasaha/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
/opt/homebrew/anaconda3/lib/python3.8/site-packages/fuzzywuzzy/fuzz.py:11:
UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein
to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-
Levenshtein to remove this warning')
```

[11]:
```
text1 = majors["MAJOR_NAME"].astype("str")
```

1.1.2 Data Cleaning:

Here I am removing stop words, punctuations, digits

[12]:
```
text2 = [remove_stopwords(x)\
        .translate(str.maketrans('','',string.punctuation))\
        .translate(str.maketrans('','',string.digits))\
        for x in text1]
print(text2[:5])
```

```
['Chemical Engineering', 'Undergraduate Pathway', 'PreGeneral Engineering',
'Social Work', 'Accounting']
```

1.1.3 Note:

Also it is a good idea to retrieve the root words since their are variations of majors names such as Engineering vs Engineer etc.

[13]:
```
def stemSentence(sentence):
    porter = PorterStemmer()
    token_words = word_tokenize(sentence)
    stem_sentence = [porter.stem(word) for word in token_words]
    return ' '.join(stem_sentence)

text3 = pd.Series([stemSentence(x) for x in text2])
print(text3[:5])
```

```
0           chemic engin
1     undergradu pathway
2         pregener engin
3            social work
4                account
dtype: object
```

1.1.4 Note:

Now let's create bag of words using scikit learn's CountVectorizer. This is a step in feature extraction, it will help me to create features from words in major names

[14]:
```
#Bag of words
vectorizer_cv = CountVectorizer(analyzer='word')
X_cv = vectorizer_cv.fit_transform(text3)
```

This will result in a sparse matrix

[15]:
```
sparse_matrix = pd.concat([text1, pd.DataFrame(X_cv.toarray())], axis=1)
sparse_matrix.dropna()
```

[15]:
```
                          MAJOR_NAME  0  1  2  3  4  5  6  7  8  …  5446  \
0               Chemical Engineering  0  0  0  0  0  0  0  0  0  …     0
1              Undergraduate Pathway  0  0  0  0  0  0  0  0  0  …     0
2            Pre-General Engineering  0  0  0  0  0  0  0  0  0  …     0
3                        Social Work  0  0  0  0  0  0  0  0  0  …     0
4                         Accounting  0  0  0  0  0  0  0  0  0  …     0
…                                  … .. .. .. .. .. .. .. .. ..  …     …
14723   History/Theory Of Architecture  0  0  0  0  0  0  0  0  0  …     0
14724             Pre Professional  0  0  0  0  0  0  0  0  0  …     0
14725  Evening MBA - Full or Part Time  0  0  0  0  0  0  0  0  0  …     0
14726             Finance- Graduate  0  0  0  0  0  0  0  0  0  …     0
14727          Global Management (MGM)  0  0  0  0  0  0  0  0  0  …     0

       5447  5448  5449  5450  5451  5452  5453  5454  5455
```

```
0          0      0      0      0      0      0      0      0      0
1          0      0      0      0      0      0      0      0      0
2          0      0      0      0      0      0      0      0      0
3          0      0      0      0      0      0      0      0      0
4          0      0      0      0      0      0      0      0      0
…     …    …    …    …    …    …    …    …    …
14723      0      0      0      0      0      0      0      0      0
14724      0      0      0      0      0      0      0      0      0
14725      0      0      0      0      0      0      0      0      0
14726      0      0      0      0      0      0      0      0      0
14727      0      0      0      0      0      0      0      0      0

[14728 rows x 5457 columns]
```

1.1.5 Note:

Now I will use TF-IDF to calculate the frequency of the words and compare it to the frequencies of all words in the text to assign it a weighted score of importance. For this I will use scikit learn's TfidfVectorizer

[16]:
```python
#TF-IDF (word level)
vectorizer_wtf = TfidfVectorizer(analyzer='word')
X_wtf = vectorizer_wtf.fit_transform(text3)
```

[18]:
```python
tf_idf_matrix = pd.concat([text1, pd.DataFrame(X_wtf.toarray())], axis=1)
tf_idf_matrix
```

[18]:
```
                              MAJOR_NAME    0    1    2    3    4    5    6  \
0                   Chemical Engineering  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1                  Undergraduate Pathway  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2                Pre-General Engineering  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3                            Social Work  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4                             Accounting  0.0  0.0  0.0  0.0  0.0  0.0  0.0
…                                    …  …  …  …  …  …  …  …
14723   History/Theory Of Architecture  0.0  0.0  0.0  0.0  0.0  0.0  0.0
14724                 Pre Professional  0.0  0.0  0.0  0.0  0.0  0.0  0.0
14725  Evening MBA - Full or Part Time  0.0  0.0  0.0  0.0  0.0  0.0  0.0
14726                Finance- Graduate  0.0  0.0  0.0  0.0  0.0  0.0  0.0
14727          Global Management (MGM)  0.0  0.0  0.0  0.0  0.0  0.0  0.0

         7    8  …  5446  5447  5448  5449  5450  5451  5452  5453  5454  \
0      0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1      0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2      0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3      0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4      0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
…     …  …  …  …   …   …   …   …   …  …   …
14723  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
```

```
14724  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
14725  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
14726  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
14727  0.0  0.0  …   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

       5455
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0
…      …
14723   0.0
14724   0.0
14725   0.0
14726   0.0
14727   0.0

[14728 rows x 5457 columns]
```

[19]:
```
#TF-IDF (n-gram level)
vectorizer_ntf = TfidfVectorizer(analyzer='word',ngram_range=(1,2))
X_ntf = vectorizer_ntf.fit_transform(text3)
```

1.1.6 LDA(Latent Dirichlet Allocation)

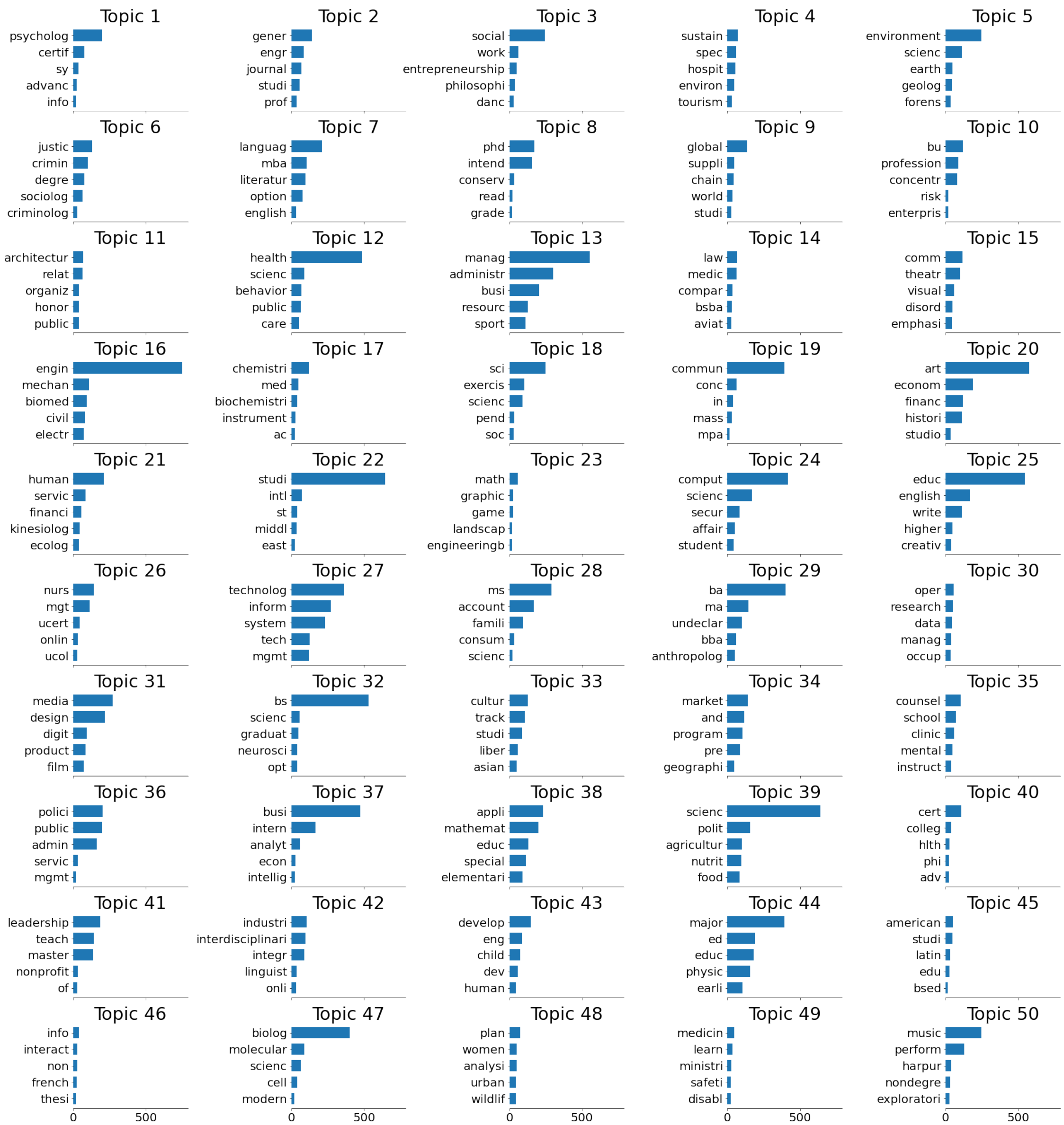
Latent Dirichlet Allocation (LDA) helps to look for patterns to formulate topics of documents based on words. The idea here is to use similar words to build a topic. It is useful for large number of documents and it is highly scalable. The onlly challenge here is to figure out optimun number of topics and give those topics generalized names. Here I have 50 topics and top 5 words per topics

[20]:
```
#LDA
lda = LatentDirichletAllocation(n_components=50, learning_decay=0.9)
X_lda = lda.fit(X_cv)

#Plot topics function. Code from: https://scikit-learn.org/stable/auto_examples/
↪applications/plot_topics_extraction_with_nmf_lda.html
def plot_top_words(model, feature_names, n_top_words, title, plot_axis_x=10,␣
↪plot_axis_y=5):
    fig, axes = plt.subplots(plot_axis_x, plot_axis_y, figsize=(30, 30),␣
↪sharex=True)
    axes = axes.flatten()
    for topic_idx, topic in enumerate(model.components_):
        top_features_ind = topic.argsort()[:-n_top_words - 1:-1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]
```

```
        ax = axes[topic_idx]
        ax.barh(top_features, weights, height=0.7)
        ax.set_title(f'Topic {topic_idx +1}',
                     fontdict={'fontsize': 30})
        ax.invert_yaxis()
        ax.tick_params(axis='both', which='major', labelsize=20)
        for i in 'top right left'.split():
            ax.spines[i].set_visible(False)
        fig.suptitle(title, fontsize=40)
    plt.subplots_adjust(top=0.90, bottom=0.05, wspace=0.90, hspace=0.3)
    plt.show()

#Show topics
n_top_words = 5
feature_names = vectorizer_cv.get_feature_names()
plot_top_words(X_lda, feature_names, n_top_words, '', plot_axis_x=10,␣
 ↪plot_axis_y=5)
```



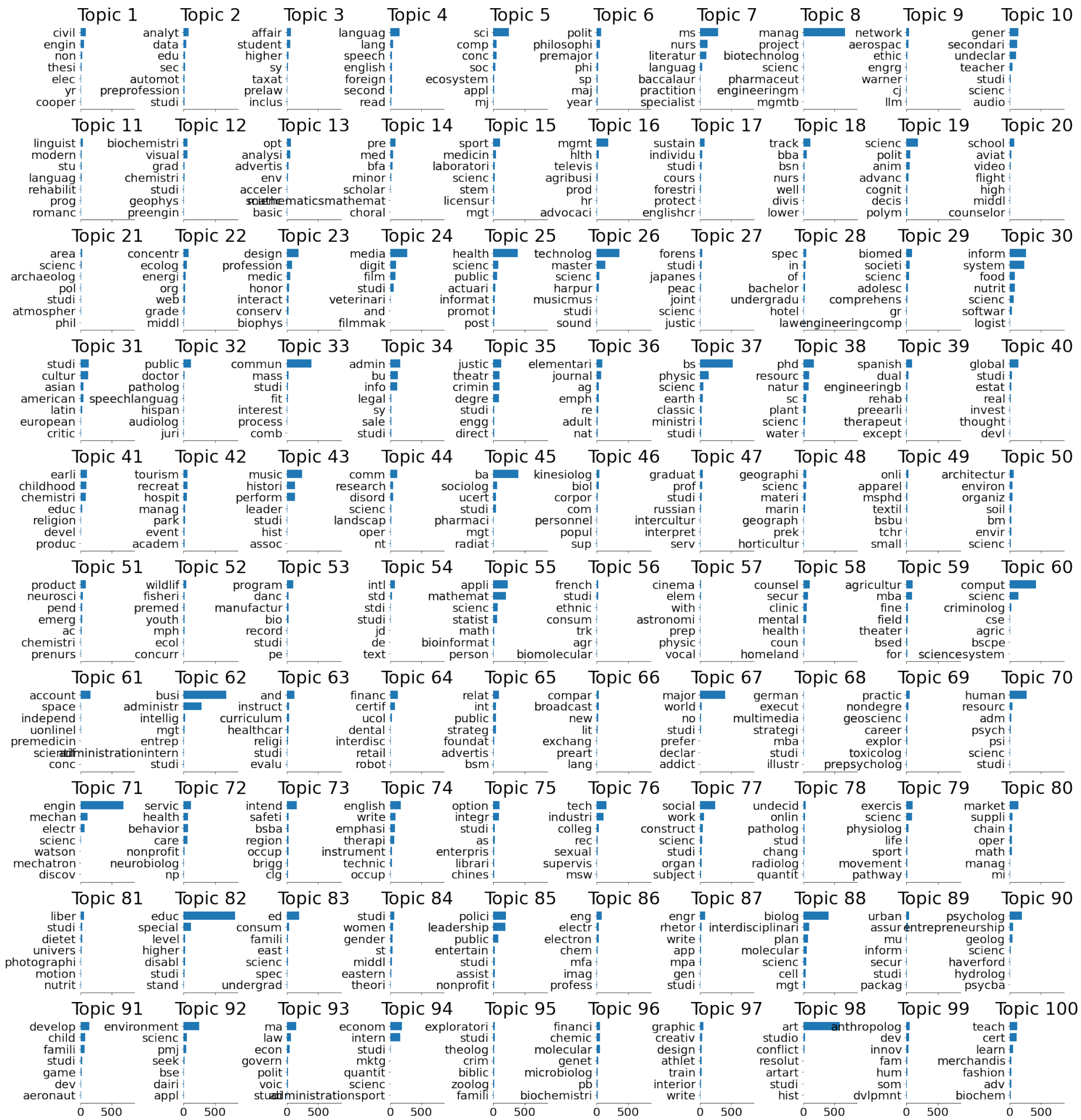
1.1.7 Note:

We see that Language, English, Literature are all in one topic but it also puts Write, Finance, Medicine in one topic which clearly is wrong. Hence, let's tune those hyper parameters a little

[21]:
```python
# modifying learning_decay=0.7 and n_components=100
lda = LatentDirichletAllocation(n_components=100, learning_decay=0.7)
X_lda = lda.fit(X_cv)

#Show topics
n_top_words = 7
```

```
feature_names = vectorizer_cv.get_feature_names()
plot_top_words(X_lda, feature_names, n_top_words, '', plot_axis_x=10,␣
 ↪plot_axis_y=10)
```

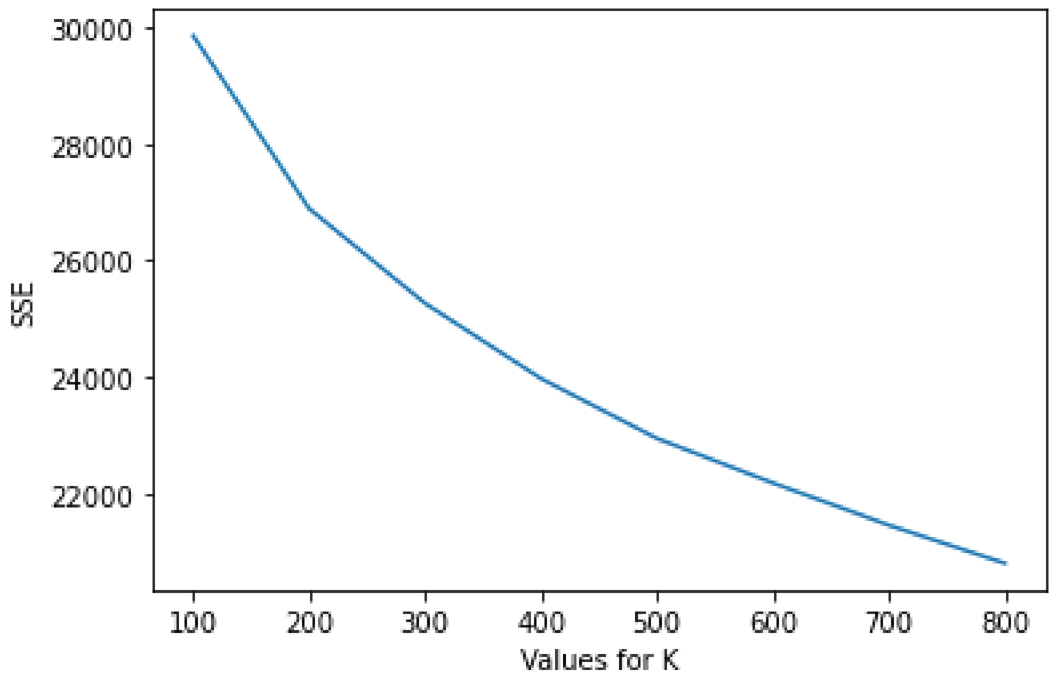


1.2 Clustering

Now I will cluster the topics for naming those topics. I am showing sum of squared errors or SSE output based on number of clusters.

[22]:
```
#Test increments of 100 clusters using elbow method
sse={}
for k in np.arange(100,900,100):
```

```
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(X_cv)
    sse[k] = kmeans.inertia_
plt.plot(list(sse.keys()),list(sse.values()))
plt.xlabel('Values for K')
plt.ylabel('SSE')
plt.show();
```



1.2.1 Note:

Increasing the number of clusters decreases SSE, but having 800 major names would probably be too much. Hence, I am going with the first ELBO value which is 200

[26]:
```
#Create 200 clusters
kmeans = KMeans(n_clusters=200)
kmeans.fit(X_cv)
result = pd.concat([text1,pd.DataFrame(X_cv.toarray(),columns=vectorizer_cv.
 ↪get_feature_names())],axis=1)
result['cluster'] = kmeans.predict(X_cv)
```

1.2.2 Cluster validation and labeling

Now that i have generated those clusters it is important to vaidate wether these are appropriate and meaningful. Here, I am trying to label those clusters based on.

For naming it is easier to give these clusters simple labels based on matrix column names retived from the LDA model that has non zero entries.

[29]:
```python
clusters = result['cluster'].unique()
labels = []
for i in range(len(clusters)):
    subset = result[result['cluster'] == clusters[i]]
    words = ' '.join([x for x in np.where(subset.all()!=0,subset.columns,None)␣
 ↪if x and x!='Name' and x!='cluster' and len(x.split()) == 1])
    labels.append(words)
labels_table = pd.DataFrame(zip(clusters,labels),columns=['cluster','label'])
result_labelled = pd.merge(result,labels_table,on='cluster',how='left')
```

[51]:
```python
major_groups = result_labelled.rename({'cluster': 'major_group_id','label':␣
 ↪'major_group_name'},␣
 ↪axis=1)[['MAJOR_NAME','major_group_id','major_group_name']]

major_groups['major_group_name'].replace('MAJOR_NAME','Others',inplace= True)
major_groups['major_group_name'].replace('MAJOR_NAME ','',inplace= True)

major_groups['major_group_name'].replace(r'MAJOR_NAME\ ','',inplace= True,␣
 ↪regex= True)
```

1.2.3 Here are the final labels

[64]:
```python
major_groups.head(10)
```

[64]:

	MAJOR_NAME	major_group_id	major_group_name
0	Chemical Engineering	150	chemic engin
1	Undergraduate Pathway	174	undergradu
2	Pre-General Engineering	13	engin
3	Social Work	143	social work
4	Accounting	35	account
5	Business Administration	55	administr busi
6	Guest Student-Undergraduate	1	Others
7	MICR-PHD	1	Others
8	Robotics	1	Others
9	Nursing	27	nurs

[55]:
```python
major_groups['major_group_name'].unique().tolist()
```

[55]:
```
['chemic engin',
 'undergradu',
 'engin',
 'social work',
 'account',
 'administr busi',
 'Others',
 'nurs',
 'intern',
```

```
'undeclar',
'manag',
'studi',
'architectur',
'scienc',
'comput engin',
'comput',
'human',
'mathemat',
'psycholog',
'write',
'financ',
'biolog',
'histori',
'music',
'scienc social',
'busi',
'chemistri',
'inform technolog',
'inform system',
'econom',
'commun',
'criminolog',
'justic',
'educ elementari',
'human servic',
'environment scienc',
'educ',
'journal',
'theatr',
'technolog',
'seek',
'educ music',
'art',
'electr engin',
'secondari',
'ba',
'market',
'art scienc',
'civil engin',
'leadership',
'engin mechan',
'cultur',
'school',
'sociolog',
'bu',
'polit',
```

```
'media studi',
'art media',
'languag',
'hear speech',
'earth scienc',
'design',
'counsel',
'anim scienc',
'therapi',
'american',
'bs',
'literatur',
'ms',
'appli',
'social',
'art histori',
'nutrit',
'biomed engin',
'engr',
'polici',
'health scienc',
'exercis scienc',
'admin',
'subject',
'agricultur',
'environment',
'natur resourc',
'suppli',
'sport',
'aerospac engin',
'integr',
'sustain',
'educ secondari',
'health',
'administr public',
'servic',
'polici public',
'health public',
'engin scienc',
'administr health',
'student',
'human resourc',
'educ physic',
'system',
'educ english',
'product',
'perform',
```

```
'bsn',
'inform',
'laboratori',
'sci',
'scienc technolog',
'busi manag',
'estat real',
'ed',
'tech',
'digit',
'intend',
'social studi',
'analyt',
'exercis',
'childhood educ',
'health servic',
'manag technolog',
'secur',
'program',
'disord',
'special',
'media',
'languag literatur',
'art studio',
'major',
'engin manag',
'fine',
'public relat',
'bs scienc',
'mechan',
'biolog bs',
'educ technolog',
'bm music',
'public',
'photographi',
'chemic',
'art visual',
'soil',
'ba studi',
'bsw social work',
'film',
'mental',
'comm',
'relat',
'ac',
'mathemat scienc',
'food scienc',
```

```
'tourism',
'leadership org',
'crimin',
'global studi',
'natur sc',
'engin ms',
'phd',
'nat re',
'art languag',
'global health',
'manag ms',
'adolesc',
'art liber',
'brigg clg',
'educ mathemat',
'agricultur plant scienc',
'dev dvlpmnt humansoc',
'administr cours individu park recreat studi tourism']
```

[59]:
```
major_full_data = pd.merge(majors, major_groups, on= "MAJOR_NAME" )

major_full_data[["major_group_id", "major_group_name"]].to_csv("major_groups.
 ↪csv")
major_full_data[["MAJOR_ID","major_group_id"]].to_csv("major_group_mapping.csv")
```

[ ]:
```

```