Topic analysis

In [1]:
```
import json
from sklearn.feature_extraction.text import TfidfVectorizer
import os
import re

def get_fnames():
    """Read all text files in a folder.
    """
    fnames = []
    for root,_,files in os.walk("./abstracts/awards_2002"):
        for fname in files:
            if fname[-4:] == ".txt":
                fnames.append(os.path.join(root, fname))
    return fnames

print("Number of abstracts in folder awards_2002: {}".format(len(get_fnames()
```

```
Number of abstracts in folder awards_2002: 9923
```

In [2]:
```
name_list = get_fnames()

def read_file(fname):
    with open(fname, 'r',encoding="ISO-8859-1") as f:
        # skip all lines until abstract
        for line in f:
            if "Abstract    :" in line:
                break

        # get abstract as a single string
        abstract = ' '.join([line[:-1].strip() for line in f])
        abstract = re.sub(' +', ' ', abstract)  # remove double spaces
        return abstract
```

In [3]:
```
documents = []

for i in name_list:
    documents.append(read_file(i))
```

In [4]:
```
print(documents[6])
```

```
This proposal focuses on problems in q-series and partitions. There are five s
eparate parts of this work. The first part considers research tied to applicat
ions of the construction of representations of Lie algebras. Next the investig
ator looks at new q-series methods related to special problems in number theor
y. The third part discusses applications of the Omega software package (http:/
/www.uni-linz.ac.at/research/ combinat/risc/software/Omega/) which is being de
veloped by the investigator in collaboration with colleagues at Linz. The focu
s in this latter section is on mutli-dimensional partitions. The fourth sectio
n is devoted to the study of Bailey chains and a consideration ofhow recent di
scoveries of the investigator may lead to new applications of this concept. Th
e proposal concludes with consideration of three major unsolved problems in th
e theory of partitions: (1) the Friedman-Joichi- Stanton conjecture, (2) the B
orwein conjecture and (3) the Okada conjecture. Each of these three conjecture
s has been around for some time. The theme of this proposal put succinctly mig
ht be: Building bridges from partitions and q-series (two intrinsically deep a
nd charming but sometimes rather introverted topics) to several branches of ma
thematics and science. The first two sections are devoted to relating this wor
k to representation theory and number theory, two branches of mathematics; in
each instance, it is clear that this interaction will not only enrich the obje
ct fields, but also will provide new insights for partitions and q-series. The
work on the Omega package has great potential. Here the investigator and his c
ollaborators have found numerous instances where research discoveries have gon
e from being unthinkable to easily reached. The possible applications to multi
-dimensional partitions should lead to insights in combinatorics and, hopefull
y, the combinatorial aspects of physics. The work on Bailey chains in the past
has had profound impact on statistical mechanics in physics. The more this met
hod is advanced, the more we may expect these mutually beneficial applications
to continue. The final section on three unsolved problems appears, at first, t
o be a purely internal study. However, as has often happend in the past, whene
ver new methods are discovered to solve really hard problems, there is almost
always a spillover into vital applications.
```

Features extraction by tf-idf with different parameters

In [5]:
```
len(documents)
```

Out[5]:
```
9923
```

In [6]:
```
# Set parameters and initialize
tfidf_vectorizer = TfidfVectorizer()
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)

# Check matrix shape
tfidf_matrix.toarray().shape # N_docs x N_terms
```

Out[6]:
```
(9923, 53816)
```

In [7]:
```
# Set parameters and initialize
tfidf_vectorizer1 = TfidfVectorizer(stop_words = 'english', lowercase= True,
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix1 = tfidf_vectorizer1.fit_transform(documents)

# Check matrix shape
tfidf_matrix1.toarray().shape # N_docs x N_terms
```

Out[7]: (9923, 27001)

In [8]:
```
# Set parameters and initialize
tfidf_vectorizer2 = TfidfVectorizer(stop_words = 'english', lowercase= True,
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix2 = tfidf_vectorizer2.fit_transform(documents)

# Check matrix shape
tfidf_matrix2.toarray().shape # N_docs x N_terms
```

Out[8]: (9923, 190731)

In [9]:
```
# Set parameters and initialize
tfidf_vectorizer3 = TfidfVectorizer(stop_words = 'english', lowercase= True,
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix3 = tfidf_vectorizer3.fit_transform(documents)

# Check matrix shape
tfidf_matrix3.toarray().shape # N_docs x N_terms
```

Out[9]: (9923, 117124)

In [10]:
```
# Set parameters and initialize
tfidf_vectorizer4 = TfidfVectorizer(stop_words = 'english', lowercase= True,
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix4 = tfidf_vectorizer4.fit_transform(documents)

# Check matrix shape
tfidf_matrix4.toarray().shape # N_docs x N_terms
```

Out[10]: (9923, 334856)

In [11]:
```
# Set parameters and initialize
tfidf_vectorizer5 = TfidfVectorizer(stop_words = 'english', lowercase= True, 
# Tip: the vectorizer also supports extracting n-gram features (common short

# Calcualate term-document matrix with tf-idf scores
tfidf_matrix5 = tfidf_vectorizer5.fit_transform(documents)

# Check matrix shape
tfidf_matrix5.toarray().shape # N_docs x N_terms
```

Out[11]: (9923, 54049)

In [12]:
```
## Inspect document frequencies (counts) of terms

from collections import Counter
terms_in_docs = tfidf_vectorizer1.inverse_transform(tfidf_matrix1)
token_counter = Counter()
for terms in terms_in_docs:
    token_counter.update(terms)

for term, count in token_counter.most_common(20):
    print("%d\t%s" % (count, term))
```

```
6642    research
5401    project
3958    new
3314    study
3184    provide
3037    used
2967    development
2944    students
2909    data
2898    use
2889    university
2868    understanding
2787    using
2774    work
2727    based
2627    high
2606    program
2596    develop
2534    important
2506    systems
```

In [13]:

```
## Inspect key words per document

features1 = tfidf_vectorizer1.get_feature_names()

for doc_i in range(5):
    print("\nDocument %d, key words by TF-IDF" % doc_i)
    for term, score in sorted(list(zip(features1,tfidf_matrix1.toarray()[doc_
        print("%.2f\t%s" % (score, term))
```

```
Document 0, key words by TF-IDF
0.30    chow
0.20    hodge
0.20    algebraic
0.15    geometry
0.15    subgroup

Document 1, key words by TF-IDF
0.24    friendships
0.23    cultivating
0.22    ethnically
0.21    control
0.20    exchanging

Document 2, key words by TF-IDF
0.31    updating
0.20    reference
0.19    secondly
0.18    method
0.16    inman

Document 3, key words by TF-IDF
0.24    conference
0.21    computations
0.20    learn
0.19    theory
0.19    group

Document 4, key words by TF-IDF
0.22    uncontrollable
0.21    commonality
0.19    uncertainties
0.19    preferences
0.18    alternative
```

In [14]:
```
## Inspect top terms per document

features2 = tfidf_vectorizer2.get_feature_names()
for doc_i in range(5):
    print("\nDocument %d, key words by TF-IDF" % doc_i)
    for term, score in sorted(list(zip(features2,tfidf_matrix2.toarray()[doc_
        print("%.2f\t%s" % (score, term))
```

```
Document 0, key words by TF-IDF
0.21    algebraic geometry
0.17    geometry algebraic
0.11    abelian varieties
0.11    algebraic finite
0.11    chow groups

Document 1, key words by TF-IDF
0.29    control engineering
0.19    2002 american
0.19    academic positions
0.19    special sessions
0.19    support american

Document 2, key words by TF-IDF
0.20    basis method
0.20    model updating
0.17    based simulation
0.16    proposed effort
0.14    model based

Document 3, key words by TF-IDF
0.19    group theory
0.16    open problems
0.13    28 2003
0.13    area learn
0.13    conference goal

Document 4, key words by TF-IDF
0.23    design alternative
0.20    design selection
0.19    engineering design
0.18    product line
0.12    alternative set
```

In [15]:
```
features3 = tfidf_vectorizer3.get_feature_names()
for doc_i in range(5):
    print("\nDocument %d, key words by TF-IDF" % doc_i)
    for term, score in sorted(list(zip(features3,tfidf_matrix3.toarray()[doc_
        print("%.2f\t%s" % (score, term))
```

```
Document 0, key words by TF-IDF
0.27    example algebraic geometry
0.27    focuses different aspects
0.27    phenomena mathematical problems
0.27    problems various fields
0.27    quantum mechanics theory

Document 1, key words by TF-IDF
0.47    nsf career program
0.47    preparing students professional
0.45    american control conference
0.43    diverse group students
0.41    support travel expenses

Document 2, key words by TF-IDF
0.44    model based simulation
0.28    aim research develop
0.28    daniel inman virginia
0.28    proposed effort improve
0.28    research develop improved

Document 3, key words by TF-IDF
0.25    24 28 2003
0.25    goal provide forum
0.25    including solid state
0.25    march 24 28
0.25    recent research developments

Document 4, key words by TF-IDF
0.41    develop integrated framework
0.38    research advance state
0.38    research develop integrated
0.36    research provide opportunities
0.35    provide opportunities students
```

In [16]:

```
features4 = tfidf_vectorizer4.get_feature_names()
for doc_i in range(5):
    print("\nDocument %d, key words by TF-IDF" % doc_i)
    for term, score in sorted(list(zip(features4,tfidf_matrix4.toarray()[doc_
        print("%.2f\t%s" % (score, term))
```

```
Document 0, key words by TF-IDF
0.21    chow
0.14    hodge
0.14    algebraic
0.14    algebraic geometry
0.11    geometry algebraic

Document 1, key words by TF-IDF
0.22    control engineering
0.15    2002 american
0.15    academic positions
0.15    nsf career program
0.15    preparing students professional

Document 2, key words by TF-IDF
0.18    updating
0.15    basis method
0.15    model updating
0.14    model based simulation
0.13    based simulation

Document 3, key words by TF-IDF
0.15    group theory
0.13    open problems
0.12    conference
0.10    computations
0.10    learn

Document 4, key words by TF-IDF
0.17    design alternative
0.15    design selection
0.14    engineering design
0.14    uncontrollable
0.14    product line
```

In [17]:

```
features5 = tfidf_vectorizer5.get_feature_names()
for doc_i in range(5):
    print("\nDocument %d, key words by TF-IDF" % doc_i)
    for term, score in sorted(list(zip(features5,tfidf_matrix5.toarray()[doc_
        print("%.2f\t%s" % (score, term))
```

```
Document 0, key words by TF-IDF
0.27    chow
0.18    hodge
0.18    algebraic
0.18    algebraic geometry
0.14    geometry algebraic

Document 1, key words by TF-IDF
0.27    control engineering
0.17    diverse group students
0.17    particular students
0.17    students seeking
0.16    cultivating

Document 2, key words by TF-IDF
0.22    updating
0.18    model based simulation
0.16    based simulation
0.14    reference
0.14    proposed effort

Document 3, key words by TF-IDF
0.20    group theory
0.17    open problems
0.16    conference
0.14    computations
0.13    learn

Document 4, key words by TF-IDF
0.21    design alternative
0.18    engineering design
0.17    product line
0.16    commonality
0.15    uncertainties
```

In [18]:

```
from sklearn.cluster import KMeans
km1 = KMeans()
km1.fit(tfidf_matrix1)
```

Out[18]:

```
KMeans()
```

In [19]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [20]:
```
print_clusters(tfidf_matrix1, km1.labels_)
```

```
Cluster: 0 (2424 docs)
wireless, software, networks, power, sensor, grid, language, network, distribu
ted, decision

Cluster: 1 (1699 docs)
polymer, magnetic, spin, quantum, nmr, reactions, laser, nano, molecules, film
s

Cluster: 2 (225 docs)
available, zygotic, zygomycota, zygomycetes, zygmund, zworski, zurich, zuni, z
ro2, zr

Cluster: 3 (1802 docs)
workshop, conference, teachers, reu, stem, mathematics, engineering, meeting,
learning, curriculum

Cluster: 4 (1865 docs)
ice, ocean, mantle, climate, solar, arctic, galaxies, seismic, stars, wind

Cluster: 5 (84 docs)
fellowship, mathematical, sciences, postdoctoral, informatics, training, micro
bial, minority, fellowships, biology

Cluster: 6 (578 docs)
equations, manifolds, spaces, algebraic, theory, geometry, algebras, quantum,
hyperbolic, topology

Cluster: 7 (1246 docs)
genes, species, protein, plant, proteins, gene, plants, genome, evolutionary,
genetic
```

In [21]:

```
from sklearn.cluster import KMeans

# Do clustering

km1 = KMeans()
km1.fit(tfidf_matrix2)
```

Out[21]:

```
KMeans()
```

In [22]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [23]:
```
print_clusters(tfidf_matrix2, km1.labels_)
```

```
Cluster: 0 (377 docs)
physical chemistry, transition metal, surface chemistry, number areas, areas i
ncluding, ray diffractometer, time resolved, focus research, analytical surfac
e, reactive intermediates

Cluster: 1 (57 docs)
sciences fellowship, mathematical sciences, biological informatics, goal fello
wship, sciences fellowships, research training, entitled modeling, entitled in
fluence, entitled role, fellowship fy2002

Cluster: 2 (758 docs)
mathematics science, middle school, computer science, community college, engin
eering technology, science mathematics, program intended, graduate education,
school districts, math science

Cluster: 3 (386 docs)
earth science, researchers represent, disciplinary databases, data sets, proof
concept, knowledge discovery, digital library, analysis networked, amounts div
erse, agencies geological

Cluster: 4 (6352 docs)
postdoctoral fellowship, number theory, real time, algebraic geometry, dynamic
al systems, representation theory, microbial biology, solar wind, dark matter,
star formation

Cluster: 5 (765 docs)
reu site, research experience, spin polarized, self assembly, shipboard scient
ific, scientific support, support equipment, equipment including, magnetic pro
perties, condensed matter

Cluster: 6 (746 docs)
sea ice, ice sheet, ice core, organic matter, carbon cycle, 000 years, carbon
dioxide, climate change, southern ocean, ice cores

Cluster: 7 (482 docs)
research fellowship, ii project, phase ii, fuel cell, drug delivery, fuel cell
s, silicon carbide, high temperature, rich media, cutting tool
```

In [24]:

```
km1 = KMeans()
km1.fit(tfidf_matrix3)
```

Out[24]:

```
KMeans()
```

In [25]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [26]:
```
print_clusters(tfidf_matrix3, km1.labels_)
```

```
Cluster: 0 (100 docs)
support month research, award support month, fellowship program enables, inter
national research fellowship, month research fellowship, archaeal bacterial vi
ral, additional training microbial, bacterial viral species, avail unique rese
arch, assist new scientists

Cluster: 1 (142 docs)
engineering initiative nsf, funded experimental physical, initiative nsf 01, 1
57 category ner, 01 157 category, project funded experimental, physical chemis
try program, students postdoctoral research, project addresses fundamental, re
ceived response nanoscale

Cluster: 2 (16 docs)
astrophysics postdoctoral fellowship, astronomy astrophysics postdoctoral, awa
rded nsf astronomy, carry program research, fellowship carry program, research
education experimental, particle physics based, experimental elementary partic
le, education experimental elementary, requests support group

Cluster: 3 (92 docs)
carnegie mellon university, collaborative research project, basis equality rec
iprocity, international collaborative research, supports collaborative researc
h, award supports collaborative, bringing leading experts, project multidiscip
linary international, multidisciplinary international collaborative, principal
investigator dr

Cluster: 4 (9124 docs)
mathematical sciences fellowship, national science foundation, partial differe
ntial equations, science foundation support, small grant exploratory, grant ex
ploratory research, proposal number cts, foundation support dr, dissertation r
esearch project, world wide web

Cluster: 5 (32 docs)
doctoral dissertation research, award award provide, enable promising student,
establish strong independent, economic cultural environmental, dissertation re
search improvement, global innovation networks, serve baseline comparison, tre
e ring chronologies, depth interviews key

Cluster: 6 (386 docs)
business innovation research, innovation research sbir, small business innovat
ion, sbir phase project, research sbir phase, innovation research phase, resea
rch phase project, phase ii project, phase project develop, phase project prop
oses

Cluster: 7 (31 docs)
biological informatics occupational, informatics fy2002 fellowship, biological
informatics fy2002, biological informatics research, biology informational com
putational, career biological informatics, computational mathematical statisti
cal, contributing future vitality, attain goal fellowship, academia industry a
ttain
```

In [27]:
```
km1 = KMeans()
km1.fit(tfidf_matrix4)
```

Out[27]:
```
KMeans()
```

In [28]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [29]:
```
print_clusters(tfidf_matrix4, km1.labels_)
```

```
Cluster: 0 (446 docs)
fuel, phase ii project, sensor, drug, phase ii, ii project, coatings, fuel cel
l, silicon, drug delivery

Cluster: 1 (1721 docs)
sensor, wireless, grid, routing, software, agents, mobile, power, code, memory

Cluster: 2 (578 docs)
manifolds, equations, algebraic, algebras, hyperbolic, spaces, number theory,
representation theory, conjecture, string

Cluster: 3 (225 docs)
available, zygotic, zygomycota aftol project, zygomycota aftol, zygomycota, zy
gomycetes ascomycetes basidiomycetes, zygomycetes ascomycetes, zygomycetes, zy
gmund operators spaces, zygmund operators

Cluster: 4 (126 docs)
fellowship, sciences fellowship, mathematical sciences fellowship, mathematica
l sciences, mathematical, sciences, postdoctoral fellowship, biological inform
atics, postdoctoral, informatics

Cluster: 5 (1966 docs)
conference, workshop, stem, symposium, teachers, reu, reu site, meeting, girls
, scholarship

Cluster: 6 (3540 docs)
ice, contract, galaxies, arctic, mantle, stars, forest, solar, soil, co2

Cluster: 7 (1321 docs)
nmr, spin, polymer, magnetic, french, films, reactions, cnrs, nanoparticles, n
anotubes
```

In [30]:

```
from sklearn.cluster import KMeans
km1 = KMeans()
km1.fit(tfidf_matrix5)
```

Out[30]:

```
KMeans()
```

In [31]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [32]:
```
print_clusters(tfidf_matrix5, km1.labels_)
```

```
Cluster: 0 (1232 docs)
stem, reu, teachers, reu site, scholarship, mathematics, girls, teacher, csems
, fellows

Cluster: 1 (3028 docs)
ice, mantle, solar, arctic, galaxies, ocean, forest, stars, wind, species

Cluster: 2 (444 docs)
phase ii, fuel, phase ii project, ii project, sensor, coatings, membrane, drug
, optical, fuel cell

Cluster: 3 (2299 docs)
manifolds, equations, quantum, grid, spaces, wireless, algebraic, code, softwa
re, power

Cluster: 4 (225 docs)
available, zurich, zr, zooplankton species, zooplankton, zoology, zoological,
zoning, zones long lived, zones long

Cluster: 5 (1802 docs)
protein, spin, nmr, proteins, complexes, polymer, molecules, magnetic, reactio
ns, arabidopsis

Cluster: 6 (126 docs)
fellowship, sciences fellowship, mathematical sciences fellowship, mathematica
l sciences, mathematical, sciences, postdoctoral fellowship, abroad, biologica
l informatics, postdoctoral

Cluster: 7 (767 docs)
workshop, conference, symposium, meeting, french, cnrs, gordon, congress, sess
ions, speakers
```

change k for ngram =(1,3)

In [33]:

```
from sklearn.cluster import KMeans

k1=3
model = KMeans(n_clusters= k1)
model.fit(tfidf_matrix5)
```

Out[33]:

```
KMeans(n_clusters=3)
```

In [34]:
```
import heapq, numpy as np

# Custom function to print top keywords for each cluster
def print_clusters(matrix, clusters, n_keywords=10):
    for cluster in range(min(clusters), max(clusters)+1):
        cluster_docs = [i for i, c in enumerate(clusters) if c == cluster]
        print("Cluster: %d (%d docs)" % (cluster, len(cluster_docs)))

        # Keep scores for top n terms
        new_matrix = np.zeros((len(cluster_docs), matrix.shape[1]))
        for cluster_i, doc_vec in enumerate(matrix[cluster_docs].toarray()):
            for idx, score in heapq.nlargest(n_keywords, enumerate(doc_vec),
                new_matrix[cluster_i][idx] = score

        # Aggregate scores for kept top terms
        keywords = heapq.nlargest(n_keywords, zip(new_matrix.sum(axis=0), fea
        print(', '.join([w for s,w in keywords]))
        print()
```

In [35]:
```
print_clusters(tfidf_matrix5, model.labels_)
```

```
Cluster: 0 (225 docs)
available, zurich, zr, zooplankton species, zooplankton, zoology, zoological,
zoning, zones long lived, zones long

Cluster: 1 (7098 docs)
ice, mantle, quantum, magnetic, protein, solar, manifolds, arctic, sensor, fau
lt

Cluster: 2 (2600 docs)
workshop, conference, fellowship, sciences fellowship, mathematical sciences f
ellowship, mathematical sciences, stem, reu, teachers, mathematical
```

In [ ]:
```

```