Drexel University

College of Computing and Informatics

INFO 371 – Data Mining Applications

Assignment 2

Name: Yuming Chen

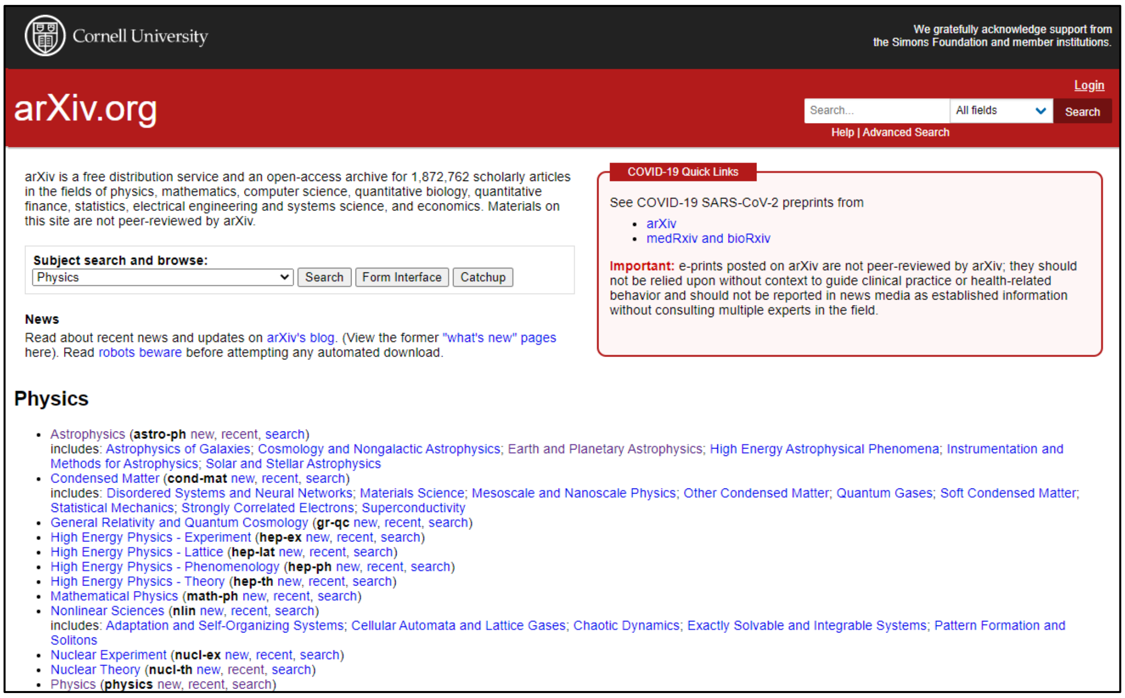
Student Number: 320180939611

Set date: Monday, March 29, 2021 Due Date: 11:59pm, Sunday April 25, 2021

Catalog

1. Data collection

The source of data is a famous curated research-sharing platform named **arXiv** (*https://arxiv.org*) supported by Cornell University Library. **arXiv** is the world's premier e-print repository in physics, math, computer science and related disciplines enabling scientists worldwide to share and access research before it is formally published. In this assignment, I chose three different domains which were "Physics", "Mathematic (math)" and "Computer Science (cs)" to complete the next steps.


Cornell University

We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org

Login

Search... All fields Search

Help | Advanced Search

arXiv is a free distribution service and an open-access archive for 1,872,762 scholarly articles in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering and systems science, and economics. Materials on this site are not peer-reviewed by arXiv.

Subject search and browse:
Physics | Search | Form Interface | Catchup

News
Read about recent news and updates on arXiv's blog. (View the former "what's new" pages here). Read robots beware before attempting any automated download.

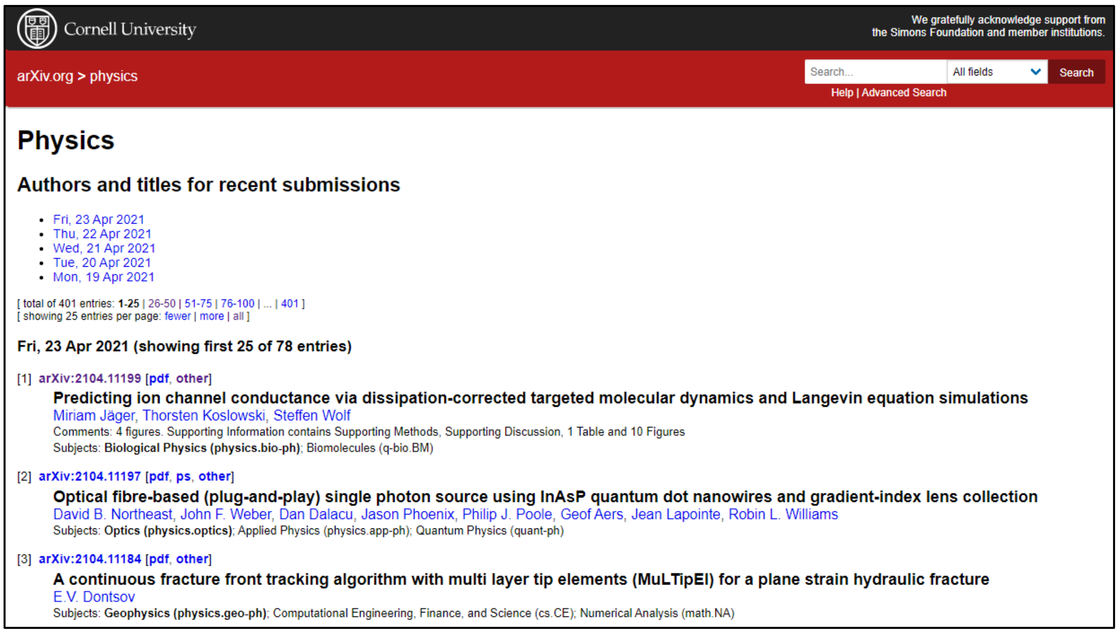
COVID-19 Quick Links

See COVID-19 SARS-CoV-2 preprints from
- arXiv
- medRxiv and bioRxiv

Important: e-prints posted on arXiv are not peer-reviewed by arXiv; they should not be relied upon without context to guide clinical practice or health-related behavior and should not be reported in news media as established information without consulting multiple experts in the field.

Physics

- Astrophysics (**astro-ph** new, recent, search)
  includes: Astrophysics of Galaxies; Cosmology and Nongalactic Astrophysics; Earth and Planetary Astrophysics; High Energy Astrophysical Phenomena; Instrumentation and Methods for Astrophysics; Solar and Stellar Astrophysics
- Condensed Matter (**cond-mat** new, recent, search)
  includes: Disordered Systems and Neural Networks; Materials Science; Mesoscale and Nanoscale Physics; Other Condensed Matter; Quantum Gases; Soft Condensed Matter; Statistical Mechanics; Strongly Correlated Electrons; Superconductivity
- General Relativity and Quantum Cosmology (**gr-qc** new, recent, search)
- High Energy Physics - Experiment (**hep-ex** new, recent, search)
- High Energy Physics - Lattice (**hep-lat** new, recent, search)
- High Energy Physics - Phenomenology (**hep-ph** new, recent, search)
- High Energy Physics - Theory (**hep-th** new, recent, search)
- Mathematical Physics (**math-ph** new, recent, search)
- Nonlinear Sciences (**nlin** new, recent, search)
  includes: Adaptation and Self-Organizing Systems; Cellular Automata and Lattice Gases; Chaotic Dynamics; Exactly Solvable and Integrable Systems; Pattern Formation and Solitons
- Nuclear Experiment (**nucl-ex** new, recent, search)
- Nuclear Theory (**nucl-th** new, recent, search)
- Physics (**physics** new, recent, search)


Cornell University

We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org > physics

Search... All fields Search

Help | Advanced Search

Physics

Authors and titles for recent submissions

- Fri, 23 Apr 2021
- Thu, 22 Apr 2021
- Wed, 21 Apr 2021
- Tue, 20 Apr 2021
- Mon, 19 Apr 2021

[total of 401 entries: **1-25** | 26-50 | 51-75 | 76-100 | ... | 401]
[showing 25 entries per page: fewer | more | all]

Fri, 23 Apr 2021 (showing first 25 of 78 entries)

[1] **arXiv:2104.11199** [**pdf**, **other**]
Predicting ion channel conductance via dissipation-corrected targeted molecular dynamics and Langevin equation simulations
Miriam Jäger, Thorsten Koslowski, Steffen Wolf
Comments: 4 figures. Supporting Information contains Supporting Methods, Supporting Discussion, 1 Table and 10 Figures
Subjects: **Biological Physics (physics.bio-ph)**; Biomolecules (q-bio.BM)

[2] **arXiv:2104.11197** [**pdf**, **ps**, **other**]
Optical fibre-based (plug-and-play) single photon source using InAsP quantum dot nanowires and gradient-index lens collection
David B. Northeast, John F. Weber, Dan Dalacu, Jason Phoenix, Philip J. Poole, Geof Aers, Jean Lapointe, Robin L. Williams
Subjects: **Optics (physics.optics)**; Applied Physics (physics.app-ph); Quantum Physics (quant-ph)

[3] **arXiv:2104.11184** [**pdf**, **other**]
A continuous fracture front tracking algorithm with multi layer tip elements (MuLTipEl) for a plane strain hydraulic fracture
E.V. Dontsov
Subjects: **Geophysics (physics.geo-ph)**; Computational Engineering, Finance, and Science (cs.CE); Numerical Analysis (math.NA)

1.1 Python Code

1.1.1 Crawler

Firstly, I captured and analyzed the packets of **arXiv** website. Then, I wrote a python script with “request” package which is a simple HTTP library to get the contents of abstracts from **arXiv** website for each article in the 3 given domains. According to the requirement, I filtered abstracts with *words less than 150*. Moreover, in order to reduce the negative impact on the model’s effect caused by the abstracts which belong to several domains at same time, I filtered the *abstracts which belong to serval domains*.

```python
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Simple python script to get abstract for article in given domains from arXiv.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"
from bs4 import BeautifulSoup
from tqdm import tqdm
import pandas as pd
import requests
import json
import time
import re

class ArxivCrawler:
    def __init__(self, headers_path, domains):
        with open(headers_path, 'r') as f:
            self.headers = json.load(f)
        self.root = "https://arxiv.org"
        self.domains = domains
        self.process = tqdm(self.domains)

    def get(self, url):
        while True:
            try:
                return BeautifulSoup(requests.get(url, headers=self.headers).text, "lxml")
            except requests.exceptions.ConnectionError:
                self.process.set_description("Fixing")
                time.sleep(10)
                continue

    def run(self):
        results = pd.DataFrame(columns=["abstract", "domain"])
        for domain in self.process:
            self.process.set_description("Article in Domain {} collecting".format(domain))
            domain_url = self.get(self.root + "/list/{}/recent".format(domain)).find(text="all")
                         .parent.get("href")
            soup = self.get(self.root + domain_url)
            articles = zip(soup.find_all("dt"), soup.find_all("dd"))
            count = 0
            for article_url, article_info in articles:
                article_url = article_url.find(attrs={"title": "Abstract"}).get("href")
                if self.__filter(domain, article_info.find(attrs={"class": "list-subjects"}).text):
                    self.process.set_description("Article No.{} in Domain {} downloading"
                                                 .format(count, domain))
                    soup = self.get(self.root + article_url)
                    abstract = self.__text_process(soup.find(attrs={"class": "abstract mathjax"})
                                                   .text)
                    if abstract:
                        results = results.append({"abstract": abstract, "domain": domain},
                                                 ignore_index=True)
                        count += 1
                if count >= 20:
                    break
        return results

    @staticmethod
    def __filter(domain, info):
        return len(re.findall("\({}[^;]*\)".format(domain), info)) >= len(info.split(";"))-1

    @staticmethod
    def __text_process(text):
        text = text.replace("\n", " ").strip()
        return text[len("Abstract:"):] if len(text[len("Abstract:"):].split("")) >= 150 else False
```

1.1.2 Prepare the data in the proper ARFF format

1.1.2.1 Remove the non-English characters

I wrote python script with “re” package which provides regular expression matching operations to remove the non-English characters including digits in the data set gotten by crawler.

```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Python script to remove the non-English characters.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"

import pandas as pd
import re

fil = re.compile(u'[^a-zA-Z0-9]+
', re.UNICODE)
data = pd.read_csv("abstracts.csv")
data["abstract"] = data.abstract.apply(lambda x: fil.sub(' ', x))
data.domain = data.domain.apply(lambda x:"\"{}\"".format(x))
data.to_csv("abstracts_new.csv", index=False)
```

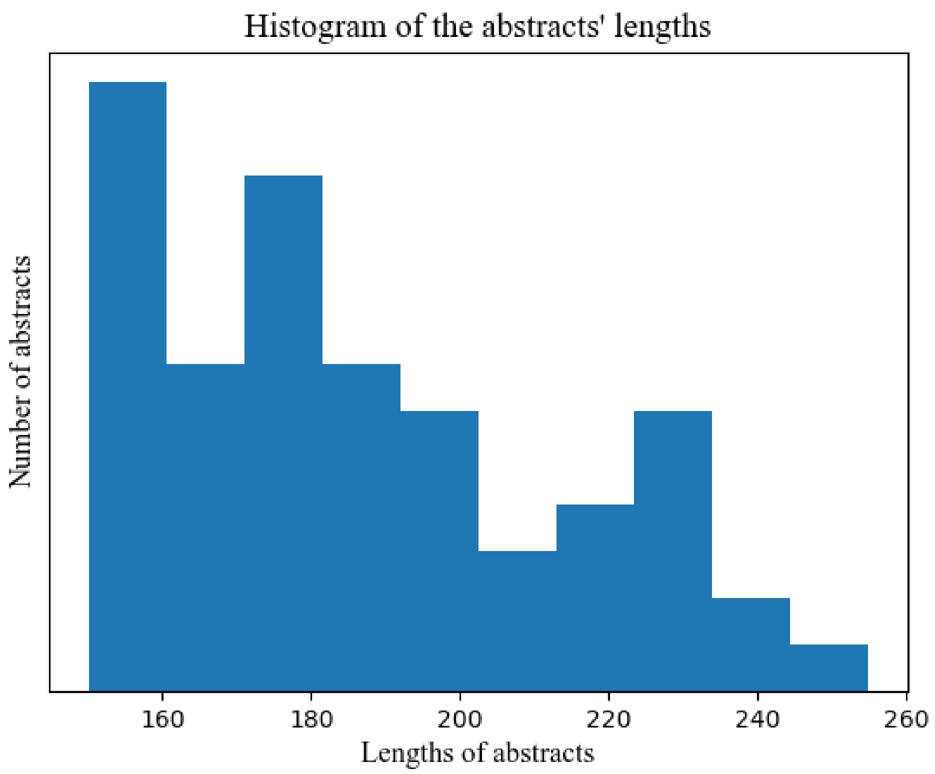
1.1.2.2 Restructure

I wrote python script to restructure the DataFrame into standard format of Weka’s ARFF file.

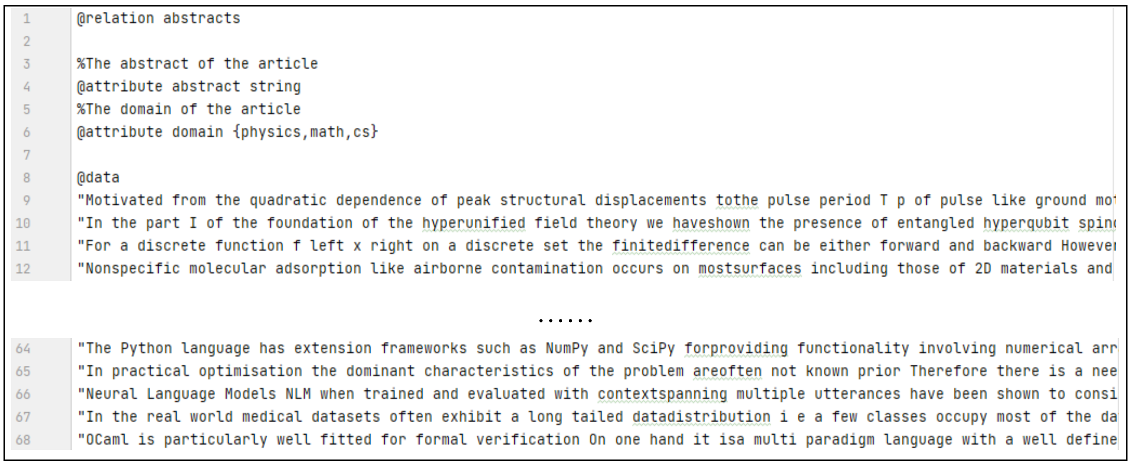
```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Python script to process and reformat "csv" file into "arff" file.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"

import pandas as pd

class ArffWriter:
    def __init__(self, data: pd.DataFrame, attributes: dict, filename: str):
        self.data = data
        self.attributes = attributes
        self.filename = filename

    def write(self):
        with open(self.filename+".arff", "w",encoding="utf-8",errors="ignore") as f:
            f.write("@relation {}\n\n".format(self.filename))
            for name, attribute in self.attributes.items():
                f.write("%{}\n".format(attribute["des"]))
                f.write("@attribute {name} {type}\n"
                        .format(name=name, type=attribute["type"]))
            f.write("\n@data\n")
            for values in self.data.values:
                for value_id in range(len(values)):
                    mark = "\"" if attributes[list(attributes.keys())[value_id]]["type"] ==
                    "string" else ""
                    f.write("{}{}{}".format(mark,values[value_id],mark))
                    if value_id != len(values)-1:
                        f.write(",")
                f.write("\n")

if __name__ == "__main__":
    filename = "abstracts"
    data = pd.read_csv("abstracts_new.csv")
    attributes = {"abstract":
                      {"type": "string",
                       "des": "The abstract of the article"},
                  "domain":
                      {"type": "{%s}" % ",".join(data["domain"].unique()),
                       "des": "The domain of the article"}}
    arffwriter = ArffWriter(data[["abstract","domain"]],attributes,filename)
    arffwriter.write()
```

1.2 Overview of Dataset

Domain	**Number of Articles**	**Length of articles' abstracts**	
		Max(words)	**Min**(words)
Computer Science	20	257	155
Mathematics	20	324	157
Physics	20	280	158



1.3 Overview of ARFF file

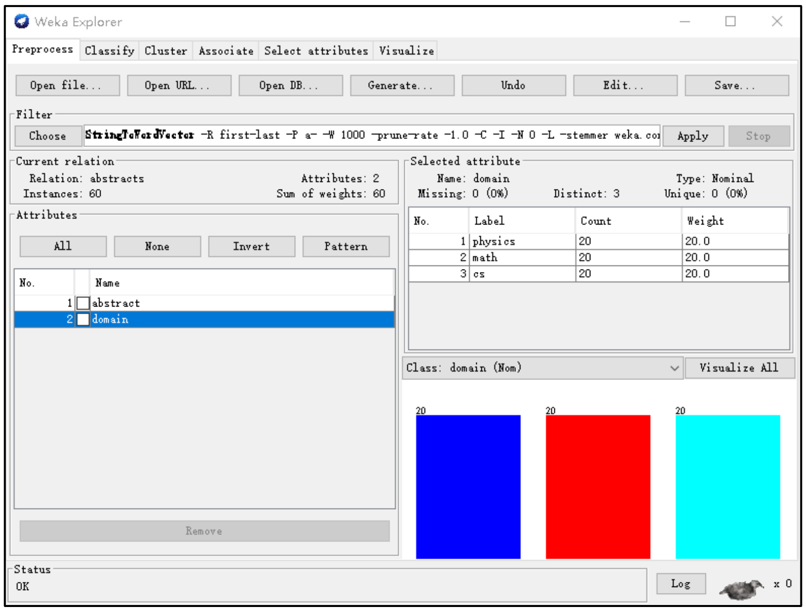


```
@relation abstracts

%The abstract of the article
@attribute abstract string
%The domain of the article
@attribute domain {physics,math,cs}

@data
"Motivated from the quadratic dependence of peak structural displacements tothe pulse period T p of pulse like ground mo
"In the part I of the foundation of the hyperunified field theory we haveshown the presence of entangled hyperqubit spin
"For a discrete function f left x right on a discrete set the finitedifference can be either forward and backward Howeve
"Nonspecific molecular adsorption like airborne contamination occurs on mostsurfaces including those of 2D materials and

                                    ......

"The Python language has extension frameworks such as NumPy and SciPy forproviding functionality involving numerical arr
"In practical optimisation the dominant characteristics of the problem areoften not known prior Therefore there is a nee
"Neural Language Models NLM when trained and evaluated with contextspanning multiple utterances have been shown to consi
"In the real world medical datasets often exhibit a long tailed datadistribution i e a few classes occupy most of the da
"OCaml is particularly well fitted for formal verification On one hand it isa multi paradigm language with a well define
```

2. Task 1: Data Preprocessing, Transformation, and Analysis

2.1 String Vectorized to Binary Representation

2.1.1 Open **abstracts.arff** in Weka Explorer and choose **unsupervised-> attribute-> StringToWordVector** for Filter.



Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter

Choose **StringToWordVector** -R first-last -P a- -W 1000 -prune-rate -1.0 -C -I -N 0 -L -stemmer weka.co Apply Stop

Current relation
Relation: abstracts   Attributes: 2
Instances: 60   Sum of weights: 60

Selected attribute
Name: domain   Type: Nominal
Missing: 0 (0%)   Distinct: 3   Unique: 0 (0%)

No.	Label	Count	Weight
1	physics	20	20.0
2	math	20	20.0
3	cs	20	20.0

Attributes

All | None | Invert | Pattern

No.		Name
1	☐	abstract
2	☐	domain

Remove

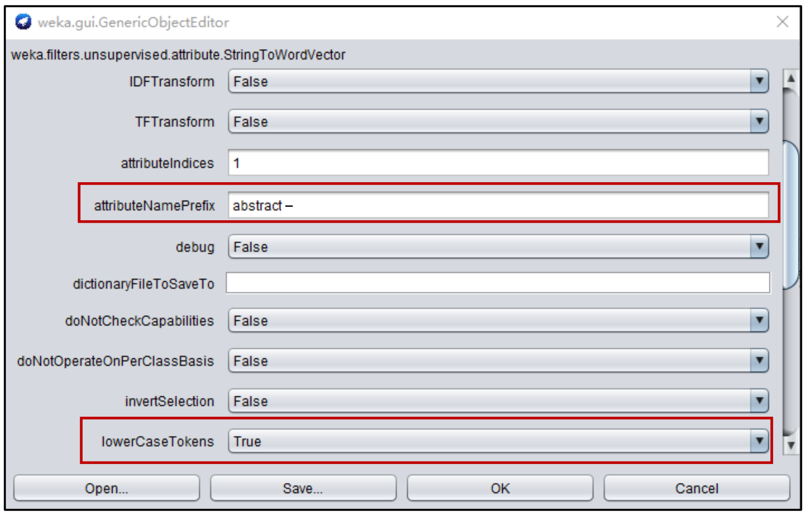
Class: domain (Nom)   Visualize All

Status
OK   Log   x 0

2.1.2 Click the text box containing “StringToWordVector –R …” to bring up the property editor to change one option below:

*2.1.2.1 Select **true** for **lowerCaseTokens** and click **OK***

To avoid the Exception “***Problem filtering instances: attribute names are not unique. Cause: domain***” from Weka caused by the duplicated attribute name, I added a prefix “abstract_” to each result with **attributeNamePrefix**.



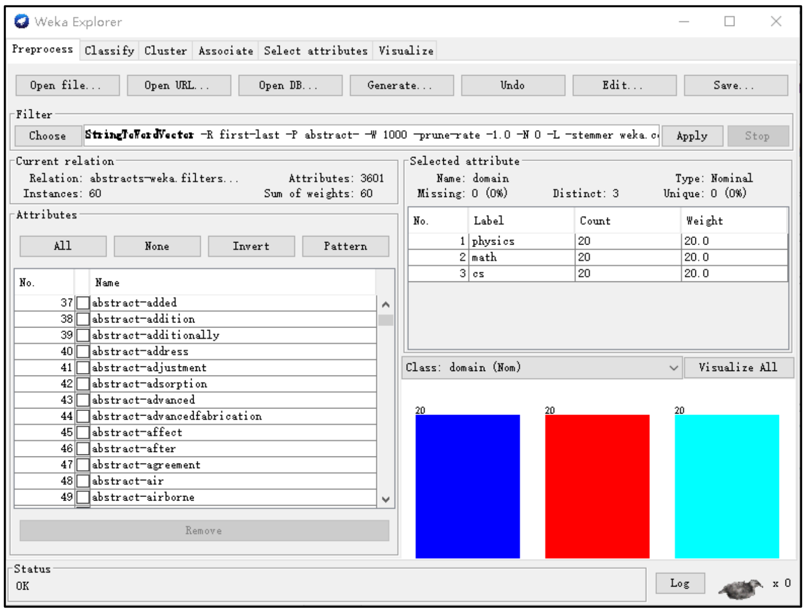
weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.StringToWordVector

IDFTransform	False
TFTransform	False
attributeIndices	1
attributeNamePrefix	abstract_
debug	False
dictionaryFileToSaveTo	
doNotCheckCapabilities	False
doNotOperateOnPerClassBasis	False
invertSelection	False
lowerCaseTokens	True

Open... | Save... | OK | Cancel

2.1.3 Click **Apply** to run the vectorization filter and you will see a number of new word attributes after it is done.



Filter: StringToWordVector -R first-last -P abstract- -W 1000 -prune-rate -1.0 -N 0 -L -stemmer weka.c

Current relation
Relation: abstracts-weka.filters...   Attributes: 3601
Instances: 60   Sum of weights: 60

Selected attribute
Name: domain   Type: Nominal
Missing: 0 (0%)   Distinct: 3   Unique: 0 (0%)

No.	Label	Count	Weight
1	physics	20	20.0
2	math	20	20.0
3	cs	20	20.0

Class: domain (Nom)

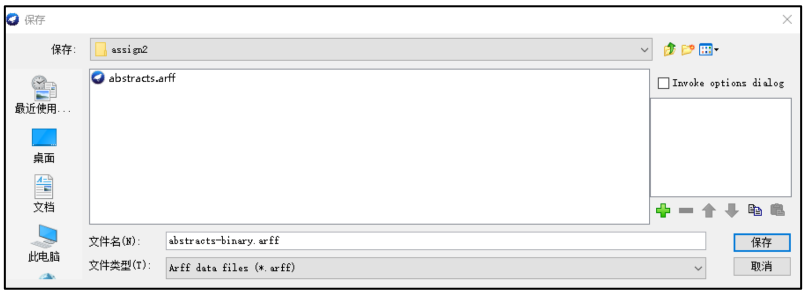
Attributes

No.		Name
37	☐	abstract-added
38	☐	abstract-addition
39	☐	abstract-additionally
40	☐	abstract-address
41	☐	abstract-adjustment
42	☐	abstract-adsorption
43	☐	abstract-advanced
44	☐	abstract-advancedfabrication
45	☐	abstract-affect
46	☐	abstract-after
47	☐	abstract-agreement
48	☐	abstract-air
49	☐	abstract-airborne

Status
OK

x 0

2.1.4 Click **Save…** to save the result (binary vector representation) as **abstracts-binary.arff**.

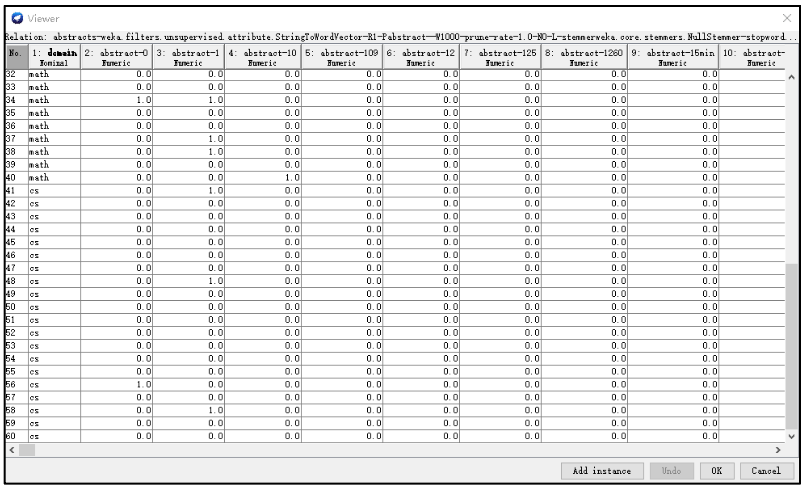


abstracts.arff

文件名(N): abstracts-binary.arff
文件类型(T): Arff data files (*.arff)

2.1.5 Click **Edit…** to view the processed data and manually sample to **create a data table**:

2.1.5.1 Edit Page



Relation: abstracts-weka.filters.unsupervised.attribute.StringToWordVector-R1-Pabstract—W1000-prune-rate-1.0-N0-L-stemmerweka.core.stemmers.NullStemmer-stopword...

No.	1: domain Nominal	2: abstract-0 Numeric	3: abstract-1 Numeric	4: abstract-10 Numeric	5: abstract-109 Numeric	6: abstract-12 Numeric	7: abstract-125 Numeric	8: abstract-1260 Numeric	9: abstract-15min Numeric	10: abstract- Numeric
32	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
33	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
34	math	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
35	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
36	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
37	math	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
38	math	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
39	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
40	math	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
41	cs	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
42	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
43	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
44	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
45	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
46	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
47	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
48	cs	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
49	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
50	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
51	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
52	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
53	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
54	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
55	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
56	cs	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
57	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
58	cs	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
59	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
60	cs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

2.1.5.2 Select word

① Stop Words

In order to select the proper Stop words, I computed the sum of binary vector of words for each domain. Then, I chose the words whose binary vector sum are same in all domains.

```
df = pd.read_csv("abstracts-binary.csv").groupby("domain").sum().T
df["similar_number"] = df.apply(lambda x:len(x)-len(set(x))+1,axis=1)
print(df[df["similar_number"] == 3] .sort_values(by=["cs"],ascending=False))
```

word	**Sum of Binary Vector**			**Similar Items**
	Computer Science	Mathematics	Physics	
the	20	20	20	3
of	20	20	20	3
to	20	20	19	2

② Rare Words

In order to select the proper rare words which are unique in each domain, I computed the sum of binary vector of words and selected one from the unique words for each domain.

```
df = pd.read_csv("abstracts-binary.csv").groupby("domain").sum().T
for col in df.columns:
    print(df[(df[filter(lambda x: x != col,df.columns)].sum(axis=1)==0) & (df[col] != 
           0)].sort_values(by=[col],ascending=False))
```

word	**Sum of Binary Vector**		
	Computer Science	Mathematics	Physics
convolutional	1	0	0
orthogonal	0	1	1
gauge	0	0	1

③ Other Words

I randomly chose three words which does not appear in the above two situations.

```
df = pd.read_csv("abstracts-binary.csv").groupby("domain").sum().T
print(df[df != 0].dropna().drop_duplicates(subset=df.columns))
```

④ Result

Type	**word**	**Sum of Binary Vector**		
		Computer Science	Mathematics	Physics
Stop words	the	20	20	20
	of	20	20	20
	to	20	20	19
Rare Words	convolutional	1	0	0
	orthogonal	0	1	1
	gauge	0	0	1
Other Words	algorithm	3	4	2
	system	3	3	4
	demonstrate	2	2	3

2.1.5.3 Table

	Feature/attribute weight								
Paper# (domain)	the	of	to	convolutional	orthogonal	gauge	algorithm	system	demonstrate
1 (Physics)	1	1	1	0	0	1	0	0	1
28 (Mathematics)	1	1	1	0	1	0	1	0	0
49 (Computer Science)	1	1	1	1	0	0	0	0	0

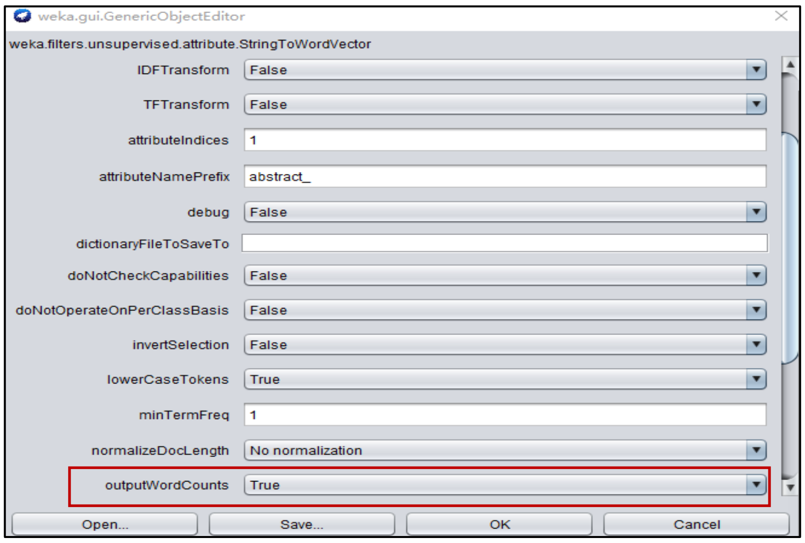
```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Python script to create a data table for each given word.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"

import pandas as pd

def to_table(from_path: str, to_path: str,words: iter[str]):
    df = pd.read_csv(from_path)
    df = df.groupby("domain").sample(1,  ,random_state=0)
    words = map(lambda word: "abstract_{}".format(word), words)
    papers = df.groupby("domain").sample(1).loc[:, map(lambda word:
                                                "abstract_{}".format(word), words)]
    papers.columns = words
    papers.index = ["{}({})".format(i, df.loc[i, "domain"].replace("\'", "")) for i in
                    papers.index]
    papers.index.name = "Paper# (domain)"
    papers.to_csv(to_path)
```

2.2 String Vectorized to TF Representation

2.2.1 Reopen **abstracts.arff** file in Weka and follow all steps in Task 1.A, except for step 2

*2.2.1.1 Select **true** for **lowerCaseTokens;***

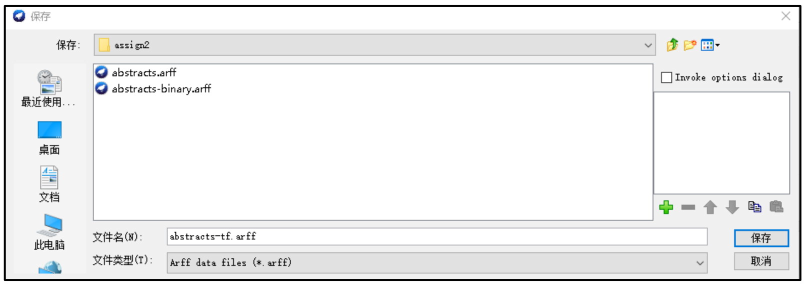
*2.2.1.2 Select **true** for **outputWordCounts** (this will produce Term Frequency/counts rather than binary 0/1 values).*


weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.StringToWordVector

IDFTransform	False
TFTransform	False
attributeIndices	1
attributeNamePrefix	abstract_
debug	False
dictionaryFileToSaveTo	
doNotCheckCapabilities	False
doNotOperateOnPerClassBasis	False
invertSelection	False
lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True

2.2.2 Save the processed file as **abstracts-tf.arff**



保存: assign2

abstracts.arff
abstracts-binary.arff

Invoke options dialog

文件名(N): abstracts-tf.arff

文件类型(T): Arff data files (*.arff)

2.2.3 Create a similar data table (of the same attributes/words and papers) with TF representation.

	Feature/attribute	weight							
Paper# (domain)	the	of	to	convolutional	orthogonal	gauge	algorithm	system	demonstrate
1 (Physics)	8	13	2	0	0	5	0	0	1
28 (Mathematics)	15	16	3	0	1	0	2	0	0
49 (Computer Science)	14	12	4	1	0	0	0	0	0

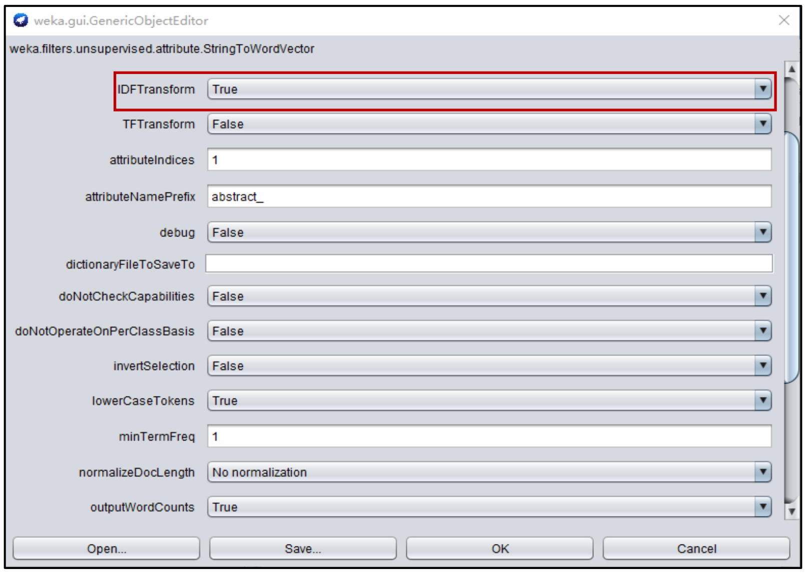
2.3 String Vectorized to TF*IDF Representation

2.3.1 Reopen **abstracts.arff** file in Weka and follow all steps in Task 1.A, except for step 2

*2.3.1.1 Select **true** for **lowerCaseTokens;***

*2.3.1.2 Select **true** for **outputWordCounts** (this will produce Term Frequency/counts rather than binary 0/1 values).*

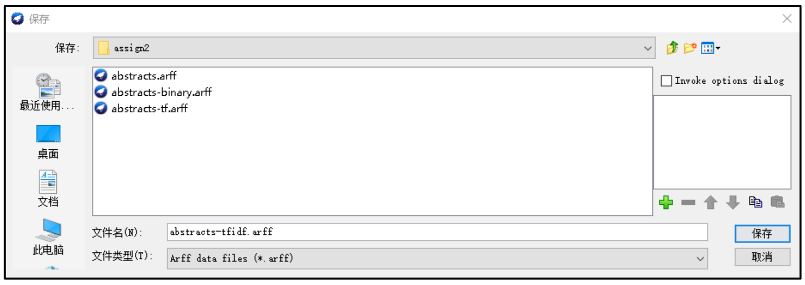
2.3.1.3 Select ***true*** *for* ***IDFTransform*** *(this with outputWordCounts generates TF*IDF weights)*



weka.filters.unsupervised.attribute.StringToWordVector

IDFTransform	True
TFTransform	False
attributeIndices	1
attributeNamePrefix	abstract_
debug	False
dictionaryFileToSaveTo	
doNotCheckCapabilities	False
doNotOperateOnPerClassBasis	False
invertSelection	False
lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True

2.3.2 Save the processed file as **abstracts-tfidf.arff**



abstracts.arff
abstracts-binary.arff
abstracts-tf.arff

文件名(N): abstracts-tfidf.arff
文件类型(T): Arff data files (*.arff)

2.3.3 Create the same data table with TF*IDF representation.

	Feature/attribute weight								
Paper# (domain)	the	of	to	convolutional	orthogonal	gauge	algorithm	system	demonstrate
1 (Physics)	0	0	0.03	0	0	20.5	0	0	2.1
28 (Mathematics)	0	0	0.05	0	4.1	0	3.8	0	0
49 (Computer Science)	0	0	0.06	3.4	0	0	0	0	0

2.4 Comparison and Discussion

By now, you should have three data tables with binary, TF, and TF*IDF representations. Compare them and discuss how weights change from one representation to the other. Use examples (e.g. specific papers and word tokens) to explain the difference and rationale.

2.4.1 Stop Words

Paper# (domain)	**Representation**	**the**	**of**	**to**
1 (Physics)	binary	1	1	1
	TF	8	13	2
	TF*IDF	0	0	0.03
28 (Mathematics)	binary	1	1	1
	TF	15	16	3
	TF*IDF	0	0	0.05
49 (Computer Science)	binary	1	1	1
	TF	14	12	4
	TF*IDF	0	0	0.06

*2.4.1.1 **Binary** representation*

When the Word Vector with **binary** representation, the weights of most Stop Words are 1. This is because the 1 in **binary** representation means that the word appears in the document and Stop Words which are common words that appear many times in most documents. For example, "the" appears in document #28, hence the weight of "the" is 1 for document #28 with **binary** representation. Furthermore, because the **binary** representation only can choose 1 and 0, the weights with **binary** representation are always between those with **TF** representation and **TF*IDF** representation.

*2.4.1.2 **TF** representation*

When the Word Vector with **TF** representation, the weights of most Stop Words are larger than **binary** representation and **TF*IDF** representation. This is because the **TF** (Term Frequency) refers to the number of times a term occurs in a document and the Stop Words are often found in each document. For example, the times that "of" appears in document #49 is 12, so the weight of "the" is 12 for document #49 with **TF** representation. Furthermore, because the Term Frequency is not limited and does not consider the Stop Words which occurs frequently in all documents, the weights of Stop Words with **TF** representation are often larger than those with **TF** representation and **TF*IDF** representation.

*2.4.1.2 **TF*IDF** representation*

When the Word Vector with **TF*IDF** representation, the weights of most Stop Words are smaller than **binary** representation and **TF*IDF** representation. The **TF*IDF** representation introduce the Inverse Document Frequency is a measure of how much information the word provides to reduce the negative impact caused by the common words which is meaningless such as Stop Words. The formula of **TF*IDF** is:

$$W_{t,d} = TF_{t,d} \times log(N / DF_t)$$

According to the formula, the weight of Stop Words is extremely low because the DF_t of them is relatively high. Takes "to" in document #1 as an example, its weight with **TF*IDF** representation is:

$$W_{to,\#1} = 2 \times log(60/59) \approx 0.036$$

2.4.2 Rare Words

Paper# (domain)	Representation	convolutional	orthogonal	gauge
1 (Physics)	binary	0	0	1
	TF	0	0	5
	TF*IDF	0	0	20.47
28 (Mathematics)	binary	0	1	0
	TF	0	1	0
	TF*IDF	0	4.09	0
49 (Computer Science)	binary	1	0	0
	TF	1	0	0
	TF*IDF	3.40	0	0

*2.4.2.1 **Binary** representation*

Due to the Rare Words are unique in each domain, the weight of each Rare Word in the document which does not belong to specific domain is 0, when the Word Vector with **binary** representation. For example, “convolutional” only appears in document #49 from “Computer Science”, hence the weights with **binary** representation of “convolutional” in the document #28 from “Mathematics” and document #1 from “Physics” are 0. Furthermore, because the **binary** representation only can choose 1 and 0, the weights with **binary** representation are always between those with **TF** representation and **TF*IDF** representation.

*2.4.2.2 **TF** representation*

When the Word Vector with **TF** representation, some weights are larger than **binary** representation and **TF*IDF** representation. This is because the **TF** (Term Frequency) representation refers to the Term Frequency which refers to the appears time of word. To be specific, the weight of each Rare Word is 0 in the document which does not belong to specific domain which is same as the **binary** representation. For example, “orthogonal” only appears once time in document #28 from “Mathematics”, hence the weights with **TF** representation of “convolutional” in document #49 from “Computer Science” and document #1 from “Physics” are 0. Furthermore, because the Term Frequency is not limited and does not consider the Rare Words which is meaningful but occurs only few times in all documents, the weights of Rare Words with **TF** representation are often small which is not a good idea.

*2.4.2.3 **TF*IDF** representation*

When the Word Vector with **TF*IDF** representation, the weights of most Stop Words, the weights of most Stop Words are smaller than **binary** representation and **TF*IDF** representation. This is because the **TF*IDF** representation introduce the Inverse Document Frequency is a measure of how much information the word provides to increase the positive impact caused by the words which are unique in each domain. The formula of **TF*IDF** is:

$$W_{t,d} = TF_{t,d} \times log(N\ /DF_t)$$

According to the formula, the weights of Rare Words became higher because the DF_t of them is relatively low. Takes “gauge” in document #1 from “Physics” as an example, its weight with **TF*IDF** representation is:

$$W_{gauge,\#1} = 5 \times log(60/1) \approx 20.47$$

2.4.3 Other Words

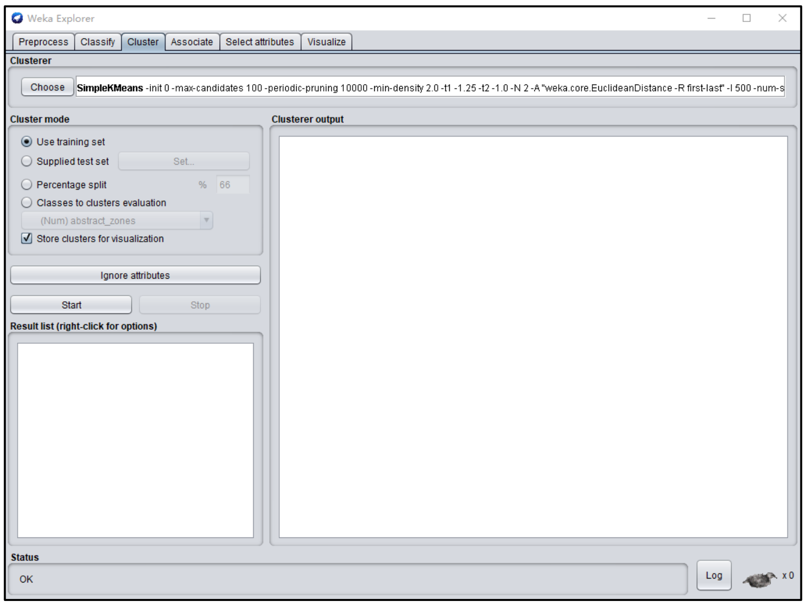
Paper# (domain)	**Representation**	**algorithm**	**system**	**demonstrate**
1 (Physics)	binary	1	1	1
	TF	8	13	2
	TF*IDF	0	0	0.03
28 (Mathematics)	binary	1	1	1
	TF	15	16	3
	TF*IDF	0	0	0.05
49 (Computer Science)	binary	1	1	1
	TF	14	12	4
	TF*IDF	0	0	0.06

3. Task 2: **Data Clustering**

3.1 Clustering with the three representations

Load each of the three processed **arff** files, namely abstracts-binary.arff, abstracts-tf.arff, and abstracts-tfidf.arff in Weka and select the Cluster tab to perform the following clustering:

3.1.1 Choose **SimpleKMeans** for Clusterer;

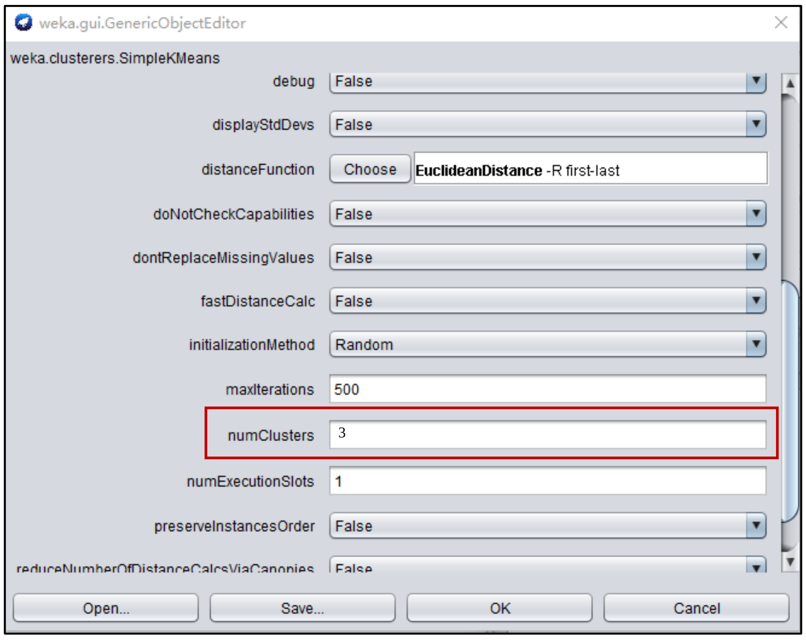


SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-s

● Use training set
○ Supplied test set Set...
○ Percentage split % 66
○ Classes to clusters evaluation
(Num) abstract_zones
☑ Store clusters for visualization

OK

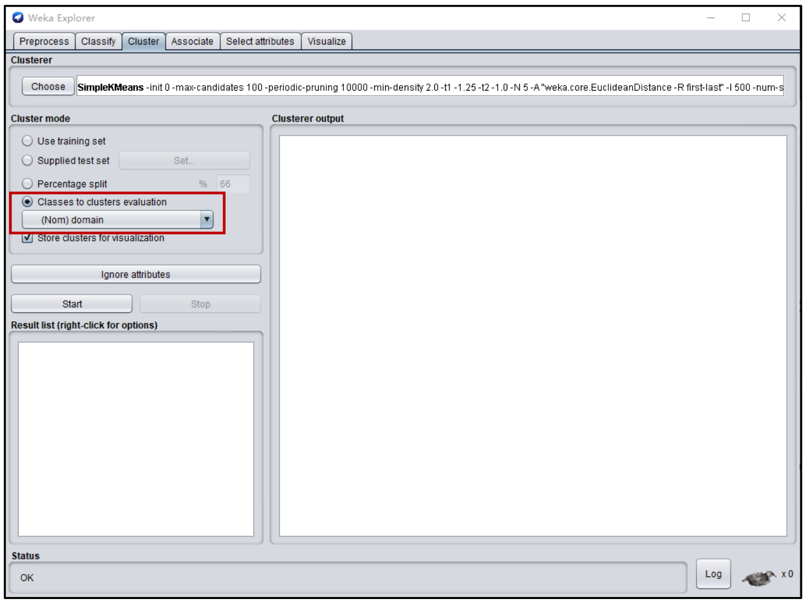
3.1.2 Click the text box with "SimpleKMeans –N …" and set **numClusters** to **the number of domains/areas** you have in the data (3 to 5).



weka.clusterers.SimpleKMeans

debug	False
displayStdDevs	False
distanceFunction	**EuclideanDistance** -R first-last
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	3
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False

3.1.3 Select **(Nom) domain** attribute for **Classes to clusters evaluation** and Start.



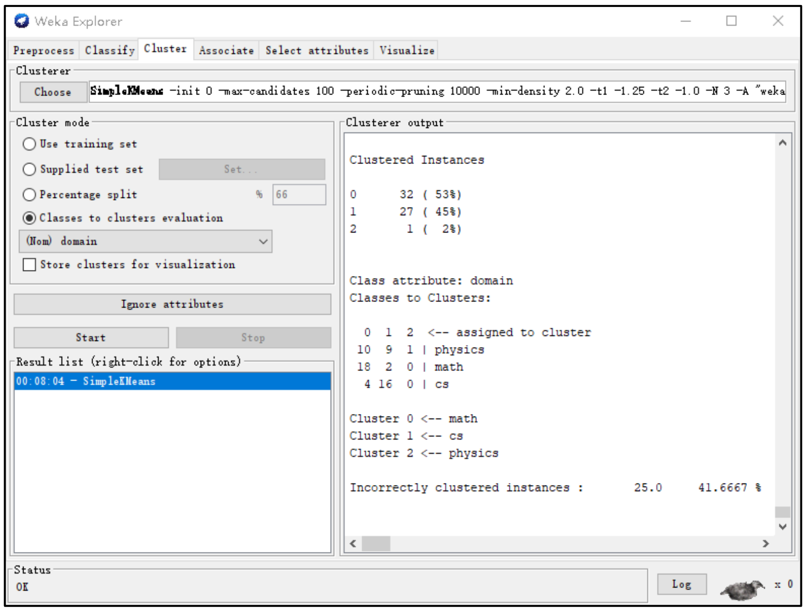
Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-s

Cluster mode: Classes to clusters evaluation, (Nom) domain; Store clusters for visualization [✓]

Status: OK

3.1.4 Take note of **percentage** of **incorrectly clustered instances** at the end of the output.

3.1.4.1 Binary



Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka

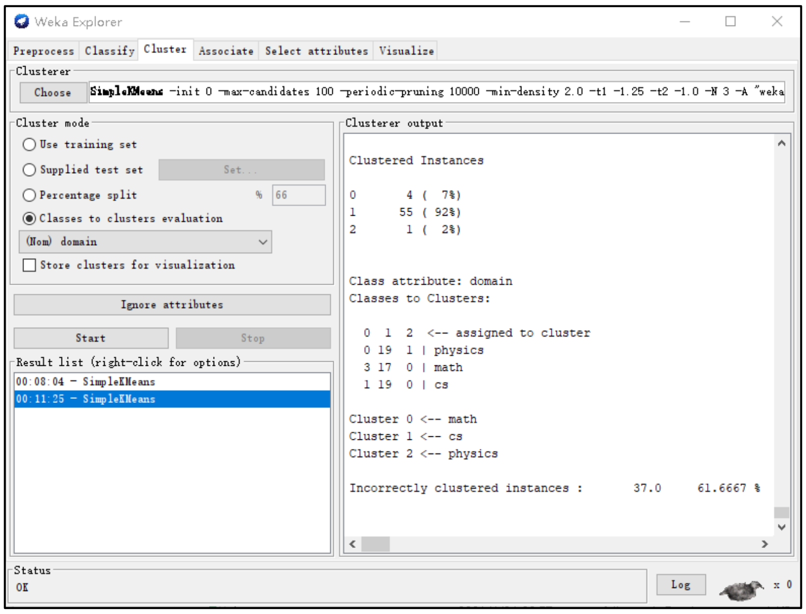
Cluster mode: Classes to clusters evaluation, (Nom) domain

Result list: 00:08:04 - SimpleKMeans

```
Clustered Instances

0       32 ( 53%)
1       27 ( 45%)
2        1 (  2%)


Class attribute: domain
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 10  9  1 | physics
 18  2  0 | math
  4 16  0 | cs

Cluster 0 <-- math
Cluster 1 <-- cs
Cluster 2 <-- physics

Incorrectly clustered instances :       25.0      41.6667 %
```

Status: OK

3.1.4.2 Term Frequency



Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka

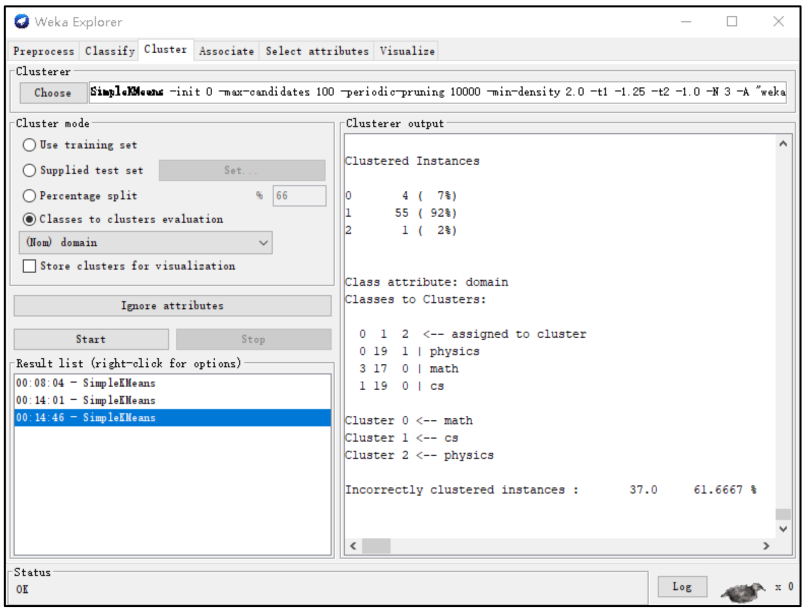
Cluster mode: Classes to clusters evaluation, (Nom) domain; Percentage split % 66

Result list (right-click for options):
00:08:04 - SimpleKMeans
00:11:25 - SimpleKMeans

Status: OK

```
Clustered Instances

0       4 (  7%)
1      55 ( 92%)
2       1 (  2%)


Class attribute: domain
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 19  1 | physics
  3 17  0 | math
  1 19  0 | cs

Cluster 0 <-- math
Cluster 1 <-- cs
Cluster 2 <-- physics

Incorrectly clustered instances :       37.0      61.6667 %
```

3.1.4.3 Term Frequency-inverse document frequency



Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka

Cluster mode: Classes to clusters evaluation, (Nom) domain; Percentage split % 66

Result list (right-click for options):
00:08:04 - SimpleKMeans
00:14:01 - SimpleKMeans
00:14:46 - SimpleKMeans

Status: OK

```
Clustered Instances

0       4 (  7%)
1      55 ( 92%)
2       1 (  2%)


Class attribute: domain
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 19  1 | physics
  3 17  0 | math
  1 19  0 | cs

Cluster 0 <-- math
Cluster 1 <-- cs
Cluster 2 <-- physics

Incorrectly clustered instances :       37.0      61.6667 %
```

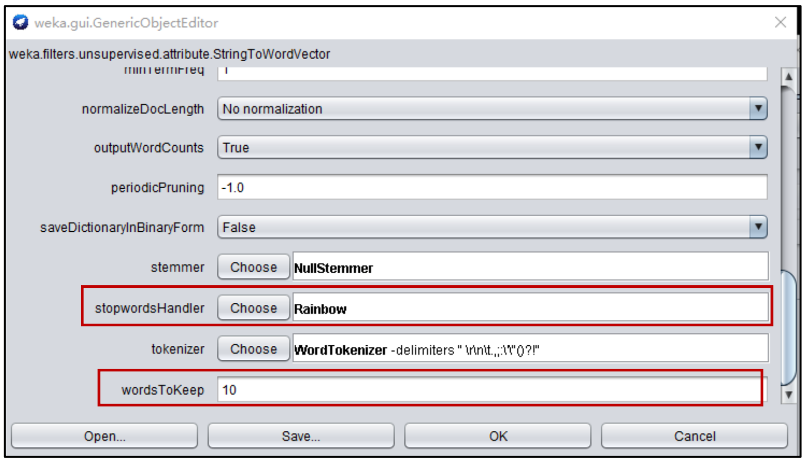
3.1.5 Create the following table summarizing clustering results based the three different representations:

	Clustering effectiveness with different vector representations		
	Binary	**TF**	**TFIDF**
Incorrectly clustered (%):	41.67%	61.667%	61.667%

3.2 Clustering with different numbers of words/attributes

3.2.1 Choose **Rainbow** as the Stopword Handler (for stopword removal);

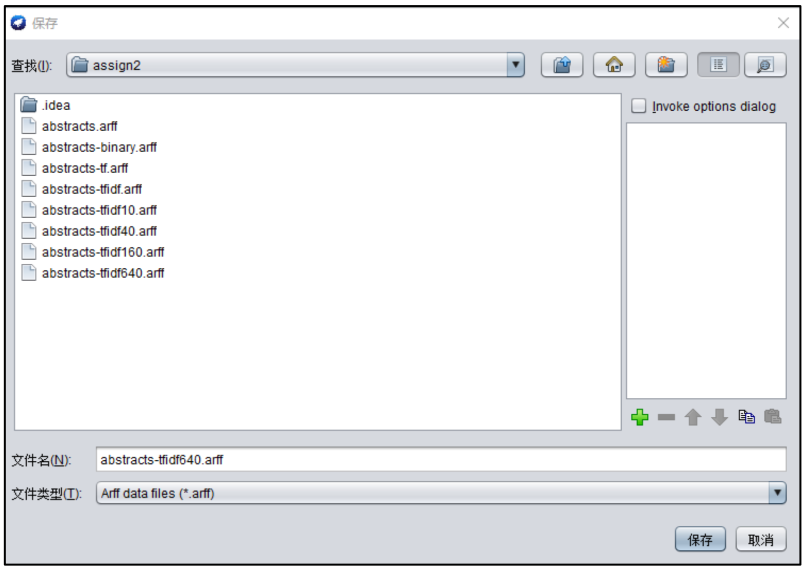
3.2.2 Change the **wordsToKeep** (per class) to 10, 40, 160, 640;



weka.filters.unsupervised.attribute.StringToWordVector

minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	True
periodicPruning	-1.0
saveDictionaryInBinaryForm	False
stemmer	**NullStemmer**
stopwordsHandler	**Rainbow**
tokenizer	**WordTokenizer** -delimiters " \r\n\t.,;:\'\"()?!"
wordsToKeep	10

3.2.3 Please save each **vectorized data** as abstracts-tfidf10.arff, abstracts-tfidf40.arff, etc.



查找(I): assign2

.idea
abstracts.arff
abstracts-binary.arff
abstracts-tf.arff
abstracts-tfidf.arff
abstracts-tfidf10.arff
abstracts-tfidf40.arff
abstracts-tfidf160.arff
abstracts-tfidf640.arff

文件名(N): abstracts-tfidf640.arff

文件类型(T): Arff data files (*.arff)

3.2.4 For each wordsToKeep setting, vectorize the data, and perform the same KMeans clustering as in **Task 2.A**;

3.2.5 Report the results using the following table:

	Clustering effectiveness with different number of attributes			
	10	40	160	640
#Attributes	31+domain	121+domain	489+domain	3328+domain
Incorrectly clustered (%):	56.67 %	45%	43.33%	56.67%

3.3 Compare and discuss the difference with reason/insight

Compare results with binary, TF, TF*IDF, and with the different number of attributes (words to keep). Discuss and explain **what** may have contributed to the results (in terms of the number of incorrectly clustered instances) and **why**.

3.3.1 Word Representation

	Clustering effectiveness with different vector representations		
	Binary	**TF**	**TFIDF**
Incorrectly clustered (%):	41.67%	61.67%	61.67%

	Clustering result								
	Binary			**TF**			**TFIDF**		
	physics	math	cs	physics	math	cs	physics	math	cs
Clustered Instances	53%	45%	2%	7%	92%	2%	7%	92%	2%

According to the above table, the proportion of ***Incorrectly clustered instances*** with **binary** representation is 41.67% which refers it has the best ***clustering effectiveness***. However, the proportion of ***Incorrectly clustered instances*** with **TF** representation and **TF*IDF** representation are both 61.67% which refers they have worse ***clustering effectiveness*** than **binary** representation. Moreover, almost 98% instances are assigned to 2 clusters and only 2% instances are assigned to the 3rd cluster with **binary** representation. Similarly, almost 92% instances are assigned to the 2nd cluster with **TF** representation and **TF*IDF** representation. In most case, above situation is abnormal, because people think the **TF** representation and **TF*IDF** representation are advanced method which have better data expression ability than **binary** representation. In my opinion, the reasons cause this unexpected situation are as follows:

3.3.1.1 The dataset is too sparse

In this dataset, the number of attributes is 3601 which is a relatively high-dimensional and the too sparse which means there are a lot of zero values in the dataset. A lot of zero values in dataset which cause extremely incomplete in dataset will make the results of clustering become more susceptible to errors, so the above conclusion of ***clustering effectiveness*** with different vector representations is not accurate.

3.3.1.2 Distance of attributes between instances is close

In this dataset, the number of instances is 60 which is extremely smaller than the number of attributes which means the dataset cannot provide enough information to the model for clustering. Moreover, most words provided by instances are useless to the clustering model, because most domain specific words which have great contribute to model only appear once in the instance which provide them. Therefore, the words which appear in several domains at same time makes the distance between each instance became closer which misleading the model to assigned most instances into only one or two clusters.

3.3.1.3 There may be overlapping features between domains

In the previous process, I have filtered out the abstracts which belong to several domains at same time. However, the three fields "physics", "Mathematics" and "computer science" I selected belong to science and engineering disciplines and have strong similarity. Therefore, there will be some noise attributes that make the distance between each instance closer, resulting in inappropriate clustering results.

3.3.2 Attributes Number

	Clustering effectiveness with different number of attributes			
	10	40	160	640
#Attributes	31+domain	121+domain	489+domain	3328+domain
Incorrectly clustered (%):	56.67 %	45%	43.33%	56.67%

According to the above table and material from Internet, “**wordsToKeep**” refers to that for each document, if the word frequency is greater than “**minTermFreq”** (defult 1) are kept. Because the documents in each ARFF is reserved, the final number of attributes is usually larger than the “**wordsToKeep**” value we set. Moreover, the Stop word Handler named “**Rainbow**” which is a program based on the *Bow* library that performs statistical text classification. According to the above table, the proportion of ***Incorrectly clustered instances*** with 160 “**wordsToKeep**” is 43.33% which refers it has the best ***clustering effectiveness***. The proportion of ***Incorrectly clustered instances*** with 40 “**wordsToKeep**” is 45% which refers it has the second-best ***clustering effectiveness***. The proportion of ***Incorrectly clustered instances*** with 10 “**wordsToKeep**” and 640 “**wordsToKeep**” are both 56.67% which refers they have worse ***clustering effectiveness***. According to the above analysis, it can be well concluded that with the increase of the number, the ***clustering effectiveness*** of the model first increases and then decreases.

3.3.2.1 Increase

The reason why the ***clustering effectiveness*** increases from 10 “**wordsToKeep**” to 40 “**wordsToKeep**” is more information are provided to the clustering model which make the border of each cluster clearer when the attributes increase**.** Moreover, the number of attributes is smaller than the default “**wordsToKeep**” (1000) which decrease the degree of dataset sparse.

3.3.2.2 Decrease

The reasons why the ***clustering effectiveness*** decreases from 160 “**wordsToKeep**” to 640 “**wordsToKeep**” are:

① The increasing of attributes introduce more zero values into the model, which increase the degree of dataset sparse. This will make the result become sensitive to the errors.

② The increasing of attributes will introduce more noise attributes into the model, which will misleading the clustered result.