Assignment: Data Types and Data Representation

Sergei Rogov

All the files belonging to this assignment can be found in my public repository: https://github.com/SergeiRogov/Data_representation.git

Activity 1

(a) Time in terms of AM or PM.
Binary, qualitative, nominal (If we only care if it's first half of the day or second)
Discrete, qualitative, ordinal (If we measure time down to hh:mm:ss)

(b) Brightness as measured by a light meter.
Continuous, quantitative, ratio (0 is appoint of reference, presumably brightness can't be negative)

(c) Brightness as measured by people's judgments.
Discrete, qualitative, ordinal (if people only have discrete ranking)

(d) Angles as measured in degrees between 0 and 360.
Discrete, quantitative, ratio (If measured only down to angles)
Continuous, quantitative, ratio (If measured continuously)

(e) Bronze, Silver, and Gold medals as awarded at the Olympics.
Discrete, qualitative, ordinal (can rank medals from worst to best)

(f) Height above sea level.
Continuous, quantitative, ratio (0 is appoint of reference. Even though height above sea level might be negative, we can state "the altitude of this place is twice as big as the altitude of the other place")

(g) Number of patients in a hospital.
Discrete, quantitative, ratio

(h) ISBN numbers for books.
Discrete, qualitative, nominal (Even though it involves some ordering like book edition, as a whole ISBN number is nominal)

(i) Ability to pass light in terms of the following values: opaque, translucent, transparent.
Discrete, qualitative, ordinal

(j) Military rank.
Discrete, qualitative, ordinal

(k) Distance from the center of campus.
Continuous, quantitative, ratio (0 is appoint of reference, distance can't be negative)

Activity 2

a) Disadvantages of this representation:

1) The tables (matrices) representing texts in this way will be sparse. We can expect a lot of zeros because the majority of words in one text will not be presented in the others.

2) This approach doesn't take the order of the words into account. Some pieces of information will be lost.
Solution: treat some words not alone, but in pairs or in groups of three to extract more meaning.

3) It does not recognize semantics of the word.
Solution: use special libraries which are "aware" of semantics and can handle it.

4) It does not know if some words are more important than the others.
Solution: analyze the number of word inclusion in a set of documents and calculate some multiplier which scales the "importance" of a certain words.

b) I implemented this method of text representation in two ways: with and without scikit-learn library involvement.

Here are results of this method performed on a small set of sentences:

Doc1: Cats like milk.
Doc2: Dogs like meat.
Doc3: Birds like to sing.
Doc4: The sky is blue.

No scikit (order – as words appear in files):

```
cats,like,milk,dogs,meat,birds,sing,sky,blue
1,1,1,0,0,0,0,0,0
0,1,0,1,1,0,0,0,0
0,1,0,0,0,1,1,0,0
0,0,0,0,0,0,0,1,1
```

With scikit (presented in alphabetic order):

```
birds,blue,cats,dogs,like,meat,milk,sing,sky
0,0,1,0,1,0,1,0,0
0,0,0,1,1,1,0,0,0
1,0,0,0,1,0,0,1,0
0,1,0,0,0,0,0,0,1
```

The result on a bigger set of text files (first columns of the result):

No scikit:

```
december,2016,edgar,m,welch,drove,six,hours,from,his,home,washington,dc,where,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,1,1,1,0,
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,1,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,1,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,1,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,
```

With scikit:

```
000,15th,2016,250,30,abusers,addition,advertising,affairs,amazing,
0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,
0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,
1,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,
1,0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,
```

Implementation without scikit:

```
import pandas as pd
import os
import re

# a set of unnecessary words
words_to_cut = {'to', 'is', 'a', 'an', 'the', 'of', 'as', 'and', 'in',
'on', 'at'}
vocabulary = []
all_files_tokens = []


def unique_words(sequence):
    '''Function cuts repeated words from a list of words'''
    seen = set()
    return [x for x in sequence if not (x in seen or seen.add(x))]


def cut_stopwords(sequence):
    '''Function cuts words contained in a set "words_to_cut" from a list of
words'''
    return [word for word in sequence if not (word in words_to_cut)]


def vectorize(tokens):
    '''Function forms a vector representing words from dictionary contained
in a file'''
    vect = []
    for word in vocabulary:
        vect.append(1 if word in tokens else 0)
    return vect


# assign directory
directory = "/Users/macbookair/Documents/UNIC STUDIES/Machine Learning and
Data Mining I/Data_representation/files"

# making sure a list of files in a directory doesn't contain files like
.DS_store
files_in_directory = filter(lambda c: c[0] != '.' and c[0] != '~',
os.listdir(directory))
# iterate over files in this directory
for file_index, filename in enumerate(sorted(files_in_directory)):
    f = os.path.join(directory, filename)
    # checking if it is a file
    print(filename)
    if os.path.isfile(f):
        with open(f) as f:
            string = f.read().lower()  # making text lower case
            # splitting into tokens

all_files_tokens.extend([re.findall(r"\w+|[^.,!?;:'*`~@=(){}\w\s]", string,
re.UNICODE)])
            # getting rid of non-essential words
            all_files_tokens[file_index] =
cut_stopwords(all_files_tokens[file_index])
            # cutting off repeated words
            all_files_tokens[file_index] =
unique_words(all_files_tokens[file_index])
            # adding words to our dictionary
            vocabulary += all_files_tokens[file_index]
```

```
# cutting off repeated words in final version of dictionary
vocabulary = unique_words(vocabulary)
# starting to form a table to export - filling first line with words from a
dictionary
table_to_csv = [vocabulary]

# adding lines-vectors representing words contained in files
for file_tokens in all_files_tokens:
    table_to_csv.append(vectorize(file_tokens))
df = pd.DataFrame(table_to_csv)
df.to_csv(r"/Users/macbookair/Documents/UNIC STUDIES/Machine Learning and
Data Mining I/Data_representation/answers/anwser.csv", sep=',',
index=False, header=False)
print(df)
```

Implementation with scikit:

```
import os
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer

list_of_texts = []
# assign directory
directory = "/Users/macbookair/Documents/UNIC STUDIES/Machine Learning and
Data Mining I/Data_representation/files"

# making sure a list of files in a directory doesn't contain files like
.DS_store
files_in_directory = filter(lambda c: c[0] != '.' and c[0] != '~',
os.listdir(directory))
# iterate over files in this directory
for file_index, filename in enumerate(sorted(files_in_directory)):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
        with open(f) as f:
            string = f.read()
            list_of_texts.append(string)

CountVec = CountVectorizer(ngram_range=(1, 1),
                           stop_words='english')
# transform
data_to_csv = CountVec.fit_transform(list_of_texts)
# create dataframe
cv_dataframe = pd.DataFrame(data_to_csv.toarray(),
columns=CountVec.get_feature_names_out())
cv_dataframe.to_csv(r"/Users/macbookair/Documents/UNIC STUDIES/Machine
Learning and Data Mining I/Data_representation/answers/anwser_scikit.csv",
sep=',', index=False, header=True)
print(cv_dataframe)
```