SVD and Matrix Factorization on DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org websi

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they nee solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as ef as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use t information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`

Feature	Description
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represe resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retriev resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the p

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to t following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their s lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will b

```
ad&authuser=0&nonce=4jft1g4q59abm&user=03671904864144121079&hash=486rrlnvdfnbi5i5vdkq4ol6bnqk57be" -O "11_DonorsChoose_Tru
```

```
--2019-11-14 07:02:23--  https://doc-0g-5o-docs.googleusercontent.com/docs/securesc/gkcko5omvjm8g0ffrjouskdfeosq6vei/0h
Resolving doc-0g-5o-docs.googleusercontent.com (doc-0g-5o-docs.googleusercontent.com)... 173.194.218.132, 2607:f8b0:400
Connecting to doc-0g-5o-docs.googleusercontent.com (doc-0g-5o-docs.googleusercontent.com)|173.194.218.132|:443... conne
HTTP request sent, awaiting response... 403 Forbidden
2019-11-14 07:02:23 ERROR 403: Forbidden.
```

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
import re
import string
from nltk.corpus import stopwords
```

```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
# from gensim.models import Word2Vec
# from gensim.models import KeyedVectors
import pickle
from tqdm import tqdm
import os
# from plotly import plotly
# import plotly.offline as offline
# import plotly.graph_objs as go
# offline.init_notebook_mode()
# from collections import Counter
```

```
! wget --header="Host: doc-0k-5o-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x
```

```
--2019-11-14 07:02:26--  https://doc-0k-5o-docs.googleusercontent.com/docs/securesc/gkcko5omvjm8g0ffrjouskdfeosq6vei/cr
Resolving doc-0k-5o-docs.googleusercontent.com (doc-0k-5o-docs.googleusercontent.com)... 173.194.218.132, 2607:f8b0:400
Connecting to doc-0k-5o-docs.googleusercontent.com (doc-0k-5o-docs.googleusercontent.com)|173.194.218.132|:443... conne
HTTP request sent, awaiting response... 403 Forbidden
2019-11-14 07:02:26 ERROR 403: Forbidden.
```

```
! wget --header="Host: doc-0c-5o-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x
```

```
--2019-11-14 07:02:26--  https://doc-0c-5o-docs.googleusercontent.com/docs/securesc/gkcko5omvjm8g0ffrjouskdfeosq6vei/2g
Resolving doc-0c-5o-docs.googleusercontent.com (doc-0c-5o-docs.googleusercontent.com)... 173.194.218.132, 2607:f8b0:400
Connecting to doc-0c-5o-docs.googleusercontent.com (doc-0c-5o-docs.googleusercontent.com)|173.194.218.132|:443... conne
HTTP request sent, awaiting response... 403 Forbidden
2019-11-14 07:02:26 ERROR 403: Forbidden.
```

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	

```
# We are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
```

```
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
            'won', "won't", 'wouldn', "wouldn't"]
```

```
from tqdm import tqdm
clean_titles = []
for sent in tqdm(project_data["project_title"].values):
    titles= " "
    for w in sent.lower().split():
        if w not in stopwords:
            titles = titles + w +" "
    clean_titles.append(titles.strip())

project_data["clean_titles"]=clean_titles
project_data.drop(["project_title"],axis =1 , inplace = True )
```

```
100%|██████████| 109248/109248 [00:01<00:00, 94158.42it/s]
```

```
# Replace the NAN VAlues :
project_data = project_data.dropna(subset=["teacher_prefix"])
sujit = project_data[project_data["teacher_prefix"].isnull()]
sujit
```

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_s

```
project_data = project_data[0 : 50000]
```

```
print(project_data.shape)
```

```
(50000, 17)
```

```
Y=project_data["project_is_approved"]
len(Y)
```

```
50000
```

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

```
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	

```
# https://stackoverflow.com/a/47091490/4084039
import re
```

```
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:26<00:00, 1898.74it/s]
```

```
project_data['preprocessed_essays'] = preprocessed_essays
```

```
project_data['combined'] = project_data['preprocessed_essays'] + ' ' + project_data['clean_titles']
```

```
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Scie
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

```

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))

```

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Scie
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

```
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

Computing Sentiment Scores

2 cells hidden

Adding new feature Number of words in Essay and Title

6 cells hidden

Split the Data-Metrix into Train,Test and Cross-validation

1 cell hidden

Selecting the top-most features from essay and title .

1. The reasons behind the best 2000 features is , as the word corpus will have lot of words . so we will select the top 2000 words for the Matrix factoriz

7 cells hidden

TruncatedSVD on calculated Co-Occurence matrix

4 cells hidden

steps creating the word vector :

1. we have to ceate a dictonary of all the index with the key as the corrosponding vectors .

5 cells hidden

Vectorizing Textual Data

4 cells hidden

Same has to be done with the cv data and the test data .

as in case we use the Glov vector for example :

here we have to give our oun custom words for the list of words that we have to use for making the AVG word to vector

1. X_test_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
2. for sentence in tqdm(X_test['clean_titles'].values): # for each review/sentence
3. vector = np.zeros(300) # as word vectors are of zero length
4. cnt_words =0; # num of words with a valid vector in the sentence/review
5. for word in sentence.split(): # for each word in a review/sentence
6. if word in glove_words:
7. vector += model[word]
8. cnt_words += 1
9. if cnt_words != 0:
10. vector /= cnt_words
11. X_test_avg_w2v_vectors.append(vector)

```
#Word count for Cross validation data
cvfinal = calculating(xcv['combined'])
print("Number of words in CV data",len( cvfinal ))
tefinal = calculating(xtest['combined'])
print("Number of words in test data",len(tefinal))
```

```
100%|██████████| 11055/11055 [00:00<00:00, 14339.00it/s]
```

```
# Avg word vector of the cv data
xcvAW2V = avgw2v(cvfinal,svdFeat)
xcvAW2V=np.asarray(xcvAW2V)
print("shape of the AVG word Vector of CV data ",xcvAW2V.shape)
xteAW2V = np.asarray(avgw2v(tefinal,svdFeat))
print("shape of the AVG word Vector of test data ",xteAW2V.shape)
```

```
100%|██████████| 11055/11055 [00:20<00:00, 546.97it/s]
  0%|          | 57/16500 [00:00<00:29, 563.47it/s]shape of the AVG word Vector of CV data  (11055, 999)
100%|██████████| 16500/16500 [00:30<00:00, 548.78it/s]
shape of the AVG word Vector of test data  (16500, 999)
```

Vectorising the catogorial data

23 cells hidden

Heatmap representaion of the confusion metrix

1 cell hidden

Conclusion

2 cells hidden