DonorsChoose

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

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

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently 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, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this 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`
`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*

Feature	Description
`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 represents a 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 retrieve all 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 project was approved.

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 the 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 school 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 be NaN.

In [117]:

```
%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
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
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
```

1.1 Reading Data

In [118]:

```
project_data = pd.read_csv('train_data.csv' ,nrows = 20000)
resource_data = pd.read_csv('resources.csv')
```

In [119]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (20000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [120]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[120]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [121]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            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]))
```

1.3 preprocessing of `project_subject_subcategories`

In [122]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            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]))
```

1.3 Text preprocessing

In [123]:

```
# 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)
```

In [124]:

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

Out[124]:

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

In [125]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [126]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
```

develop early reading skills.\r\n\r\nStudents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade. This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==

IndexError   Traceback (most recent call last)
<ipython-input-126-009ab9740f10> in <module>
      6 print(project_data['essay'].values[1000])
      7 print("="*50)
----> 8 print(project_data['essay'].values[20000])
      9 print("="*50)
     10 print(project_data['essay'].values[99999])

IndexError: index 20000 is out of bounds for axis 0 with size 20000

In [127]:

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

In [128]:

```
sent = decontracted(project_data['essay'].values[19000])
print(sent)
print("="*50)
```

```
My room is filled with energetic and hungry learners. In this always changing world, it is importa
nt to be able to know how to run technology. At the end of the year, I want my students to be tech
-savvy 21st-century learners that can work and incorporate technology throughout their
studies.\r\nMy students are energetic, ready to learn kids that yearn for education.\r\nThey are a
lways wanting to try the newest apps or how they could do a project with more technology. Our scho
ol is filled with kids that have a thirst for learning and making a difference with the knowledge
they acquire. My school is filled with teachers like myself trying to come up with creative and ne
w ideas for kids to learn a specific lesson.My student will use the iPads to research and create p
rojects. They will use the the newest apps to grow their knowledge in certain subjects. Also stude
nts will use the iPads to create multi- media projects in all subjects. Also, the iPads will benef
it my kids by being able to download books to their iPads and read when they want. The goal in
that would to create a love for reading and learning.\r\nMy students will have an advantage by usi
ng technology in school that will boost them into the 21st century.\r\nAlso, instead of waiting to
get to a computer lab my students can pull out their iPads in a matter of seconds. My students wil
l also benefit from the newest apps in math, language arts, reading, and writing.nannan
==================================================
```

In [129]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My room is filled with energetic and hungry learners. In this always changing world, it is importa
nt to be able to know how to run technology. At the end of the year, I want my students to be tech
-savvy 21st-century learners that can work and incorporate technology throughout their studies.  M
y students are energetic, ready to learn kids that yearn for education.  They are always wanting t
o try the newest apps or how they could do a project with more technology. Our school is filled wi
th kids that have a thirst for learning and making a difference with the knowledge they acquire. M
y school is filled with teachers like myself trying to come up with creative and new ideas for kid
s to learn a specific lesson.My student will use the iPads to research and create projects. They w
ill use the the newest apps to grow their knowledge in certain subjects. Also students will use th
e iPads to create multi- media projects in all subjects. Also, the iPads will benefit my kids by b
eing able to download books to their iPads and read when they want. The goal in that would to
create a love for reading and learning.  My students will have an advantage by using technology in
school that will boost them into the 21st century.  Also, instead of waiting to get to a computer
lab my students can pull out their iPads in a matter of seconds. My students will also benefit fro
m the newest apps in math, language arts, reading, and writing.nannan
```

In [130]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My room is filled with energetic and hungry learners In this always changing world it is important
to be able to know how to run technology At the end of the year I want my students to be tech savv
y 21st century learners that can work and incorporate technology throughout their studies My
students are energetic ready to learn kids that yearn for education They are always wanting to try
the newest apps or how they could do a project with more technology Our school is filled with kids
that have a thirst for learning and making a difference with the knowledge they acquire My school
is filled with teachers like myself trying to come up with creative and new ideas for kids to lear
n a specific lesson My student will use the iPads to research and create projects They will use th
e the newest apps to grow their knowledge in certain subjects Also students will use the iPads to
create multi media projects in all subjects Also the iPads will benefit my kids by being able to d
ownload books to their iPads and read when they want The goal in that would to create a love for r
eading and learning My students will have an advantage by using technology in school that will boo
```

```
st them into the 21st century Also instead of waiting to get to a computer lab my students can pul
l out their iPads in a matter of seconds My students will also benefit from the newest apps in mat
h language arts reading and writing nannan
```

In [131]:

```
# https://gist.github.com/sebleier/554280
# 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', 'e
ach', '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', "d
esn'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"]
```

In [132]:

```
# 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%|████████████████████████████████████████████████████████████████| 20000/20000
[00:14<00:00, 1379.02it/s]
```

In [133]:

```
# after preprocesing
preprocessed_essays[19000]
```

Out[133]:

```
'my room filled energetic hungry learners in always changing world important able know run technol
ogy at end year i want students tech savvy 21st century learners work incorporate technology throu
ghout studies my students energetic ready learn kids yearn education they always wanting try
newest apps could project technology our school filled kids thirst learning making difference know
ledge acquire my school filled teachers like trying come creative new ideas kids learn specific le
sson my student use ipads research create projects they use newest apps grow knowledge certain sub
jects also students use ipads create multi media projects subjects also ipads benefit kids able do
wnload books ipads read want the goal would create love reading learning my students advantage usi
ng technology school boost 21st century also instead waiting get computer lab students pull ipads
matter seconds my students also benefit newest apps math language arts reading writing nannan'
```

1.4 Preprocessing of `project_title`

In [94]:

```
# similarly you can preprocess the titles also
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].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_titles.append(sent.lower().strip())
project_data['preprocessed_titles'] = preprocessed_titles
```

```
100%|████████████████████████████████████████████████████████████████| 20000/20000
[00:00<00:00, 26341.38it/s]
```

1.5 Preparing data for models

In [95]:

```
project_data.columns
```

Out[95]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_titles'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

In [96]:

```
Y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
```

In [97]:

```
X = project_data
```

In [98]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, stratify=Y)
```

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

In [99]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_categories = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
vectorizer_categories.fit(X_train['clean_categories'].values)


categories_one_hot_train = vectorizer_categories.fit_transform(X_train['clean_categories'].values)
categories_one_hot_test = vectorizer_categories.transform(X_test['clean_categories'].values)

print("After vectorizations")

print("Shape of Train data - one hot encoding ",categories_one_hot_train.shape)
print("Shape of Test data - one hot encoding ",categories_one_hot_test.shape)
print("="*100)
print(vectorizer_categories.get_feature_names())
print("="*100)
```

```
After vectorizations
Shape of Train data - one hot encoding  (13400, 9)
Shape of Test data - one hot encoding  (6600, 9)
====================================================================================================

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================================
```

In [100]:

```
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)

sub_cat_one_hot_train = vectorizer_sub_cat.fit_transform(X_train['clean_subcategories'].values)
sub_cat_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subcategories'].values)

print("After vectorizations")

print("Shape of Train data - one hot encoding ",sub_cat_one_hot_train.shape)
print("Shape of Test data - one hot encoding",sub_cat_one_hot_test.shape)
print("="*100)

print(vectorizer_sub_cat.get_feature_names())
print("="*100)
```

```
After vectorizations
Shape of Train data - one hot encoding  (13400, 30)
Shape of Test data - one hot encoding (6600, 30)
====================================================================================================

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'College_CareerPrep', 'Oth
er', 'Music', 'History_Geography', 'Health_LifeScience', 'ESL', 'EarlyDevelopment', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds',
'Literature_Writing', 'Mathematics', 'Literacy']
====================================================================================================
```

In [101]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

In [102]:

```
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [103]:

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

In [108]:

```
vectorizer_school_state = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), lo
wercase=False, binary=True)
vectorizer_school_state.fit(X_train['school_state'].values)

school_state_one_hot_train = vectorizer_school_state.fit_transform(X_train['school_state'].values)
school_state_one_hot_test = vectorizer_school_state.transform(X_test['school_state'].values)

print("After vectorizations")


print("Shape of Train data - one hot encoding",school_state_one_hot_train.shape)
print("Shape of Test data - one hot encoding",school_state_one_hot_test.shape)
print("="*100)
print(vectorizer_school_state.get_feature_names())
print("="*100)
```

```
After vectorizations
Shape of Train data - one hot encoding (13400, 51)
Shape of Test data - one hot encoding (6600, 51)
====================================================================================================

['VT', 'WY', 'ND', 'MT', 'NH', 'RI', 'DE', 'NE', 'SD', 'AK', 'NM', 'WV', 'HI', 'ME', 'DC', 'IA', 'I
D', 'KS', 'AR', 'MN', 'MS', 'CO', 'KY', 'OR', 'MD', 'NV', 'AL', 'UT', 'TN', 'WI', 'CT', 'VA', 'NJ',
'AZ', 'MA', 'OK', 'WA', 'LA', 'MO', 'IN', 'OH', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'TX', 'NY
', 'CA']
====================================================================================================
```

In [109]:

```
# Project_Grade_Category - replacing hyphens, spaces with Underscores
project_data['project_grade_category'] = project_data['project_grade_category'].map({'Grades PreK-
2': 'Grades_PreK_2',
                                                                                     'Grades 6-8' :
rades_6_8',
                                                                                     'Grades 3-5' :
rades_3_5',
                                                                                     'Grades 9-12'
Grades_9_12'})
project_data['teacher_prefix'] = project_data['teacher_prefix'].map({'Mrs.': 'Mrs', 'Ms.': 'Ms', 'M
r.' : 'Mr',
                                                                     'Teacher': 'Teacher', 'Dr.' :
'Dr'})
```

In [110]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('null')
```

In [111]:

```
my_counter = Counter()
```

```
my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [112]:

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

In [113]:

```
vectorizer_project_grade_cat = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys(
)), lowercase=False, binary=True)
vectorizer_project_grade_cat.fit(X_train['project_grade_category'].values)

project_grade_cat_one_hot_train =
vectorizer_project_grade_cat.fit_transform(X_train['project_grade_category'].values)
project_grade_cat_one_hot_test =
vectorizer_project_grade_cat.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print("="*100)
print("Shape of Train data - one hot encoding",project_grade_cat_one_hot_train.shape)
print("Shape of Test data - one hot encoding",project_grade_cat_one_hot_test.shape)
print("="*100)
print(vectorizer_project_grade_cat.get_feature_names())
```

```
After vectorizations
====================================================================================================

Shape of Train data - one hot encoding (13400, 4)
Shape of Test data - one hot encoding (6600, 4)
====================================================================================================

['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
```

In [114]:

```
my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [115]:

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

In [116]:

```
vectorizer_teacher_prefix_cat = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.key
s()), lowercase=False, binary=True)
vectorizer_teacher_prefix_cat.fit(X_train['teacher_prefix'].values.astype("U"))

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)

print("="*100)

teacher_prefix_cat_one_hot_train =
vectorizer_teacher_prefix_cat.fit_transform(X_train['teacher_prefix'].values.astype("U"))
teacher_prefix_cat_one_hot_test = vectorizer_teacher_prefix_cat.transform(X_test['teacher_prefix']
.values.astype("U"))
print("After vectorizations")
print("="*100)

print("Shape of Train data - one hot encoding",teacher_prefix_cat_one_hot_train.shape)
print("Shape of Test data - one hot encoding ",teacher_prefix_cat_one_hot_test.shape)
print("="*100)
```

```
print(vectorizer_teacher_prefix_cat.get_feature_names())
```

```
(13400, 18) (13400,)
(6600, 18) (6600,)
==================================================================================================

After vectorizations
==================================================================================================

Shape of Train data - one hot encoding (13400, 5)
Shape of Test data - one hot encoding  (6600, 5)
==================================================================================================

['null', 'Teacher', 'Mr', 'Ms', 'Mrs']
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [135]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_essay_bow = CountVectorizer(ngram_range=(2, 2),min_df=10,max_features=5000)
vectorizer_essay_bow.fit(X_train['essay'])

# BOW for essays Train Data
essay_bow_train = vectorizer_essay_bow.fit_transform(X_train['essay'])
print("Shape of matrix for TRAIN data ",essay_bow_train.shape)

# BOW for essays Test Data
essay_bow_test = vectorizer_essay_bow.transform(X_test['essay'])
print("Shape of matrix for TEST data",essay_bow_test.shape)
```

```
Shape of matrix for TRAIN data  (13400, 5000)
Shape of matrix for TEST data (6600, 5000)
```

In [24]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

In [136]:

```
vectorizer_title_bow = CountVectorizer(ngram_range=(2, 2),min_df=10,max_features=5000)
vectorizer_title_bow.fit(X_train['preprocessed_titles'])

# BOW for title Train Data
title_bow_train = vectorizer_title_bow.fit_transform(X_train['preprocessed_titles'])
print("Shape of matrix for TRAIN data ",title_bow_train.shape)

# BOW for title Test Data
title_bow_test = vectorizer_title_bow.transform(X_test['preprocessed_titles'])
print("Shape of matrix for TEST data",title_bow_test.shape)
```

```
Shape of matrix for TRAIN data  (13400, 329)
Shape of matrix for TEST data (6600, 329)
```

1.5.2.2 TFIDF vectorizer

In [138]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essay_tfidf = TfidfVectorizer(ngram_range=(2, 2),min_df=10,max_features=5000)
vectorizer_essay_tfidf.fit(X_train['essay'])

#tidf Train Data
essay_tfidf_train = vectorizer_essay_tfidf.fit_transform(X_train['essay'])
```

```
print("Shape of matrix for TRAIN data",essay_tfidf_train.shape)

#tidf Test Data
essay_tfidf_test = vectorizer_essay_tfidf.transform(X_test['essay'])
print("Shape of matrix for TEST data",essay_tfidf_test.shape)
```

```
Shape of matrix for TRAIN data (13400, 5000)
Shape of matrix for TEST data (6600, 5000)
```

In [139]:

```
vectorizer_title_tfidf = TfidfVectorizer(ngram_range=(2, 2),min_df=10,max_features=5000)
vectorizer_title_tfidf.fit(X_train['preprocessed_titles'])

#tidf Train Data
title_tfidf_train = vectorizer_title_tfidf.fit_transform(X_train['preprocessed_titles'])
print("Shape of matrix for TRAIN data",title_tfidf_train.shape)

#tidf Test Data
title_tfidf_test = vectorizer_title_tfidf.transform(X_test['preprocessed_titles'])
print("Shape of matrix for TEST data",title_tfidf_test.shape)
```

```
Shape of matrix for TRAIN data (13400, 329)
Shape of matrix for TEST data (6600, 329)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [140]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ==============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ==============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))
```

```
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)


'''
```

Out[140]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nr
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [141]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [143]:

```
# average Word2Vec Function
# compute average word2vec for each review.
# the avg-w2v for each sentence/review is stored in this list
def avg_w2v_vectors_func(sentance):
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    return vector
```

In [145]:

```
essay_avg_w2v_train = []
essay_avg_w2v_test  = []

for sentence in tqdm(X_train['essay']):
    essay_avg_w2v_train.append(avg_w2v_vectors_func(sentance)) # Avg-w2v for Train data

# Avg-w2v for Train data
print("len(essay_avg_w2v_train):",len(essay_avg_w2v_train))
print("len(essay_avg_w2v_train[0])",len(essay_avg_w2v_train[0]))

for sentence in tqdm(X_test['essay']):
    essay_avg_w2v_test.append(avg_w2v_vectors_func(sentance)) # Avg-w2v for Test data

# Avg-w2v for Test data
```

```
print("len(essay_avg_w2v_test):",len(essay_avg_w2v_test))
print("len(essay_avg_w2v_test[0])",len(essay_avg_w2v_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 13400/13400
[00:06<00:00, 1928.59it/s]
```

```
len(essay_avg_w2v_train): 13400
len(essay_avg_w2v_train[0]) 300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 6600/6600
[00:03<00:00, 1918.54it/s]
```

```
len(essay_avg_w2v_test): 6600
len(essay_avg_w2v_test[0]) 300
```

In [146]:

```
title_avg_w2v_train = []
title_avg_w2v_test  = []

for sentence in tqdm(X_train['preprocessed_titles']):
    title_avg_w2v_train.append(avg_w2v_vectors_func(sentance)) # Avg-w2v for Train data

# Avg-w2v for Train data
print("len(title_avg_w2v_train):",len(title_avg_w2v_train))
print("len(title_avg_w2v_train[0])",len(title_avg_w2v_train[0]))

for sentence in tqdm(X_test['preprocessed_titles']):
    title_avg_w2v_test.append(avg_w2v_vectors_func(sentance)) # Avg-w2v for Test data

# Avg-w2v for Test data
print("len(title_avg_w2v_test):",len(title_avg_w2v_test))
print("len(title_avg_w2v_test[0])",len(title_avg_w2v_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 13400/13400
[00:00<00:00, 45303.66it/s]
```

```
len(title_avg_w2v_train): 13400
len(title_avg_w2v_train[0]) 300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 6600/6600
[00:00<00:00, 45192.15it/s]
```

```
len(title_avg_w2v_test): 6600
len(title_avg_w2v_test[0]) 300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [147]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [148]:

```
# Compute  TFIDF weighted W2V for each sentence of the review.

def tf_idf_weight_func(sentence): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
```

```
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    return vector
```

In [31]:

```
# Similarly you can vectorize for title also
```

In [150]:

```
essay_tfidf_w2v_train = []
essay_tfidf_w2v_test  = []

for sentence in tqdm(X_train['essay']):
    essay_tfidf_w2v_train.append(tf_idf_weight_func(sentance)) #  TFIDF weighted W2V for Train
data
print("len(essay_tfidf_w2v_train)",len(essay_tfidf_w2v_train))
print("len(essay_tfidf_w2v_train[0])",len(essay_tfidf_w2v_train[0]))

for sentence in tqdm(X_test['essay']):
    essay_tfidf_w2v_test.append(tf_idf_weight_func(sentance)) #  TFIDF weighted W2V for Test data
print("len(essay_tfidf_w2v_test)",len(essay_tfidf_w2v_test))
print("len(essay_tfidf_w2v_test[0])",len(essay_tfidf_w2v_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 13400/13400 [00:
43<00:00, 311.24it/s]
```

```
len(essay_tfidf_w2v_train) 13400
len(essay_tfidf_w2v_train[0]) 300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 6600/6600
[00:25<00:00, 261.87it/s]
```

```
len(essay_tfidf_w2v_test) 6600
len(essay_tfidf_w2v_test[0]) 300
```

In [151]:

```
title_tfidf_w2v_train  = []
title_tfidf_w2v_test  = []

for sentence in tqdm(X_train['preprocessed_titles']):
    title_tfidf_w2v_train.append(tf_idf_weight_func(sentance)) #  TFIDF weighted W2V for Train
data
print("len(title_tfidf_w2v_train)",len(title_tfidf_w2v_train))
print("len(title_tfidf_w2v_train[0])",len(title_tfidf_w2v_train[0]))

for sentence in tqdm(X_test['preprocessed_titles']):
    title_tfidf_w2v_test.append(tf_idf_weight_func(sentance)) #  TFIDF weighted W2V for Test data
print("len(title_tfidf_w2v_test)",len(title_tfidf_w2v_test))
print("len(title_tfidf_w2v_test[0])",len(title_tfidf_w2v_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 13400/13400 [00:
49<00:00, 272.19it/s]
```

```
len(title_tfidf_w2v_train) 13400
len(title_tfidf_w2v_train[0]) 300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 6600/6600
[00:23<00:00, 277.04it/s]
```

```
len(title_tfidf_w2v_test) 6600
len(title_tfidf_w2v_test[0]) 300
```

1.5.3 Vectorizing Numerical features

In [152]:

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

In [153]:

```
from sklearn.preprocessing import Normalizer

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)

print("="*100)
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['price'].values.reshape(-1,1))

price_data_train = normalizer.fit_transform(X_train['price'].values.reshape(-1,1))

price_data_test = normalizer.transform(X_test['price'].values.reshape(-1,1))


print("After vectorizations")
print("="*100)
print(price_data_train.shape, Y_train.shape)
print(price_data_test.shape, Y_test.shape)
print("="*100)
```

```
(13400, 20) (13400,)
(6600, 20) (6600,)
====================================================================================================

After vectorizations
====================================================================================================

(13400, 1) (13400,)
(6600, 1) (6600,)
====================================================================================================
```

In [154]:

```
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)

print("="*100)
normalizer.fit(X_train['quantity'].values.reshape(-1,1))

quant_train = normalizer.fit_transform(X_train['quantity'].values.reshape(-1,1))
quant_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("="*100)
print("After vectorizations")
```

```
print("After vectorizations")
print(quant_train.shape, Y_train.shape)
print(quant_test.shape, Y_test.shape)
print("="*100)
```

```
(13400, 20) (13400,)
(6600, 20) (6600,)
====================================================================================================

====================================================================================================

After vectorizations
(13400, 1) (13400,)
(6600, 1) (6600,)
====================================================================================================
```

In [155]:

```
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)

print("="*100)
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_no_projects_train =
normalizer.fit_transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1
,1))
prev_no_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))

print("="*100)
print("After vectorizations")
print(prev_no_projects_train.shape, Y_train.shape)
print(prev_no_projects_test.shape, Y_test.shape)
print("="*100)
```

```
(13400, 20) (13400,)
(6600, 20) (6600,)
====================================================================================================

====================================================================================================

After vectorizations
(13400, 1) (13400,)
(6600, 1) (6600,)
====================================================================================================
```

1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [158]:

```
from scipy.sparse import hstack

X_train_merge = hstack((categories_one_hot_train, sub_cat_one_hot_train,
school_state_one_hot_train, project_grade_cat_one_hot_train, teacher_prefix_cat_one_hot_train, pri
ce_data_train, quant_train, prev_no_projects_train,title_tfidf_train, essay_tfidf_train)).tocsr()
X_test_merge = hstack((categories_one_hot_test, sub_cat_one_hot_test, school_state_one_hot_test, p
roject_grade_cat_one_hot_test, teacher_prefix_cat_one_hot_test, price_data_test, quant_test,
prev_no_projects_test,title_tfidf_test, essay_tfidf_test)).tocsr()
```

```
print("Final Data matrix")
print("="*100)
print(X_train_merge.shape, Y_train.shape)
print(X_test_merge.shape, Y_test.shape)
print("="*100)
```

```
Final Data matrix
====================================================================================================

(13400, 5431) (13400,)
(6600, 5431) (6600,)
====================================================================================================
```

In [159]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Computing Sentiment Scores

In [160]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students w
ith the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelli
gences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of differen
t backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a carin
g community of successful \
learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice
a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role pla
y in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
and create common core cooking lessons where we learn important math and writing concepts while co
oking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into maki
ng the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project woul
d expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade apple
sauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cook
books to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoymen
t for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
```

```
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 10: Clustering

- step 1: Choose any vectorizer (data matrix) that you have worked in any of the assignments, and got the best AUC value.
- step 2: Choose any of the feature selection/reduction algorithms ex: selectkbest features, pretrained word vectors, model based feature selection etc and reduce the number of features to 5k features
- step 3: Apply all three kmeans, Agglomerative clustering, DBSCAN
  - **K-Means Clustering:**
    - Find the best 'k' using the elbow-knee method (plot k vs inertia_)
  - **Agglomerative Clustering:**
    - Apply agglomerative algorithm and try a different number of clusters like 2,5 etc.
    - You can take less data points (as this is very computationally expensive one) to perform hierarchical clustering because they do take a considerable amount of time to run.
  - **DBSCAN Clustering:**
    - Find the best 'eps' using the elbow-knee method.
    - You can take a smaller sample size for this as well.
- step 4: Summarize each cluster by manually observing few points from each cluster.
- step 5: You need to plot the word cloud with essay text for each cluster for each of algorithms mentioned in step 3.

2. Clustering

2.1 Choose the best data matrix on which you got the best AUC

In [39]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [161]:

```
a = vectorizer_categories.get_feature_names()
b = vectorizer_sub_cat.get_feature_names()
c = vectorizer_school_state.get_feature_names()
d = vectorizer_project_grade_cat.get_feature_names()
e = vectorizer_teacher_prefix_cat.get_feature_names()
f = vectorizer_title_tfidf.get_feature_names()
g = vectorizer_essay_tfidf.get_feature_names()
```

In [162]:

```
from itertools import chain

feature_names_tfidf = list(chain(
a,
b,
c,
d,
e,
["Price","Quantity","Prec_no_projts"],
f,
g))
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2 Make Data Model Ready: encoding numerical, categorical features

In [40]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [ ]:

```
#handled above
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [41]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [ ]:

```
#handled above
```

2.4 Dimensionality Reduction on the selected features

In [42]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [163]:

```
#https://scikit-learn.org/stable/modules/feature_selection.html

from sklearn.ensemble import ExtraTreesClassifier
import pandas as pd

clf = ExtraTreesClassifier()

df_tfidf_5k = pd.DataFrame(X_train_merge.todense())
df_tfidf_5k.columns = feature_names_tfidf

clf = clf.fit(df_tfidf_5k,Y_train)
```

In [164]:

```
# https://datascience.stackexchange.com/questions/31406/tree-decisiontree-feature-importances-numbers-correspond-to-how-features

tfidf_5k_fimpt = {}
tfidf_5k_fimpt = dict(zip(feature_names_tfidf, clf.feature_importances_))
#https://stackoverflow.com/questions/16772071/sort-dict-by-value-python
tfidf_5k_fimpt = sorted(tfidf_5k_fimpt.items(), key=lambda x: x[1], reverse=True)

tfidf_5k_fimpt = tfidf_5k_fimpt[:5000]
#https://stackoverflow.com/questions/22412258/get-the-first-element-of-each-tuple-in-a-list-in-python


tfidf_5k_fimpt = [ seq[0] for seq in tfidf_5k_fimpt if seq[1] > 0.0 ] # choosing features greater than 0
df_tfidf_5k = df_tfidf_5k[tfidf_5k_fimpt]


df_5k_test = pd.DataFrame(X_test_merge.todense(),columns = feature_names_tfidf)

df_5k_test = df_5k_test[tfidf_5k_fimpt]
```

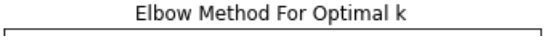
2.5 Apply Kmeans

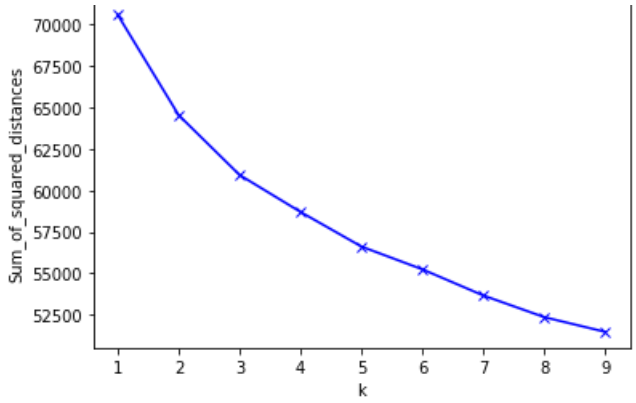
In [165]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [167]:

```
# https://pythonprogramminglanguage.com/kmeans-elbow-method/
# https://www.datasciencecentral.com/profiles/blogs/python-implementing-a-k-means-algorithm-with-sklearn
# https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import cmath as math
import sys

Sum_of_squared_distances = []

K = range(1,10)

#Find the K-Mean

for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(df_tfidf_5k)
    Sum_of_squared_distances.append(km.inertia_)


#Plotting k vs inertia

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```





In [173]:

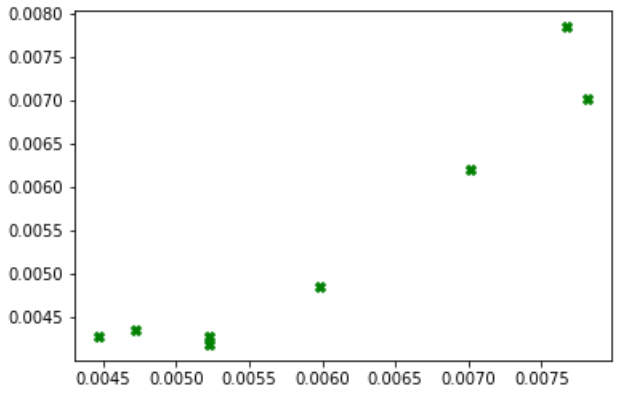
```
# Optimal k value =8

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
model = KMeans(n_clusters = 8 , n_jobs = -1)
model.fit(df_tfidf_5k)

# Cluster label
clust_labels = model.labels_

# Centroid values
cent = np.array(model.cluster_centers_)
```

In [174]:

```
#https://pythonprogramminglanguage.com/kmeans-clustering-centroid/

#Plotting the centroids
plt.scatter(cent[:,0], cent[:,1], marker="X", color='g')
```

Out[174]:

```
<matplotlib.collections.PathCollection at 0x1fab6f34f88>
```



In [175]:

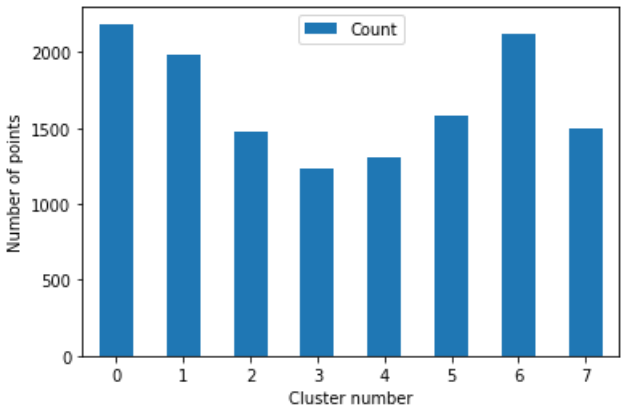
```
df_tfidf_5k_tr = df_tfidf_5k

df_tfidf_5k_tr['clusters'] = clust_labels

feature_names_tfidf_tr = feature_names_tfidf
#Adding the column to our list
feature_names_tfidf_tr.extend(['clusters'])
```

In [176]:

```
#Data Points per cluster
CountStatus = pd.value_counts(df_tfidf_5k_tr['clusters'].values, sort=False)
```

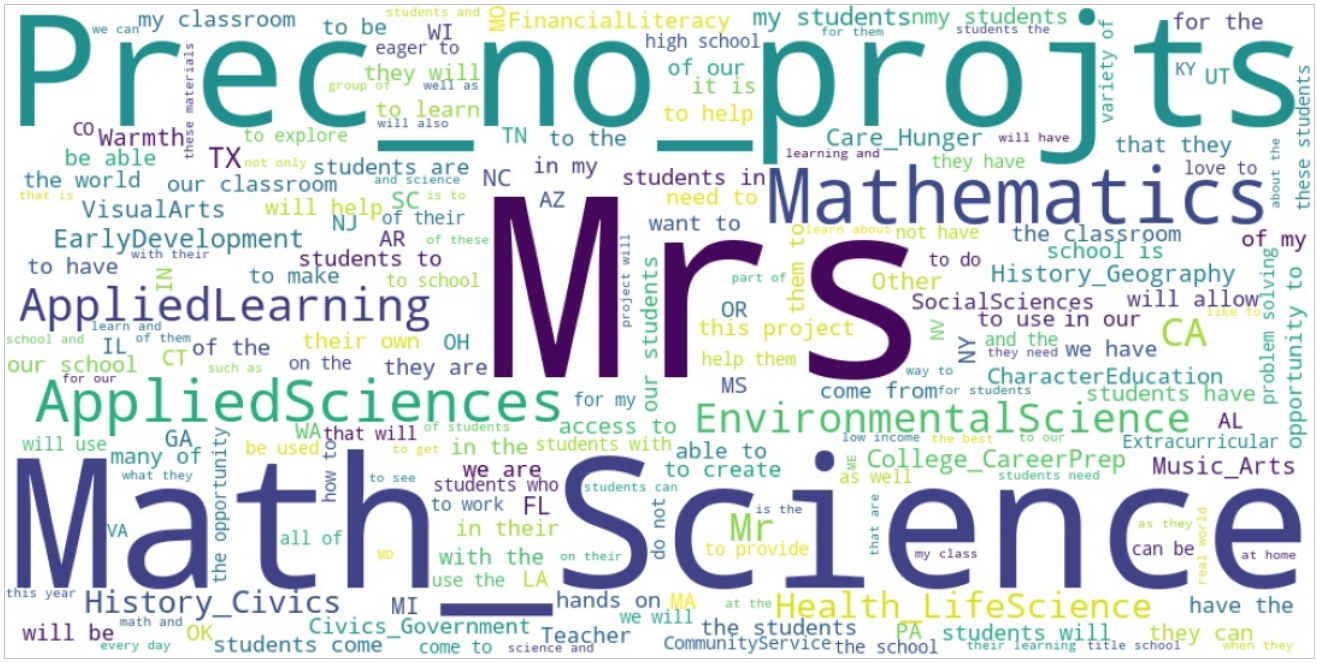
In [177]:

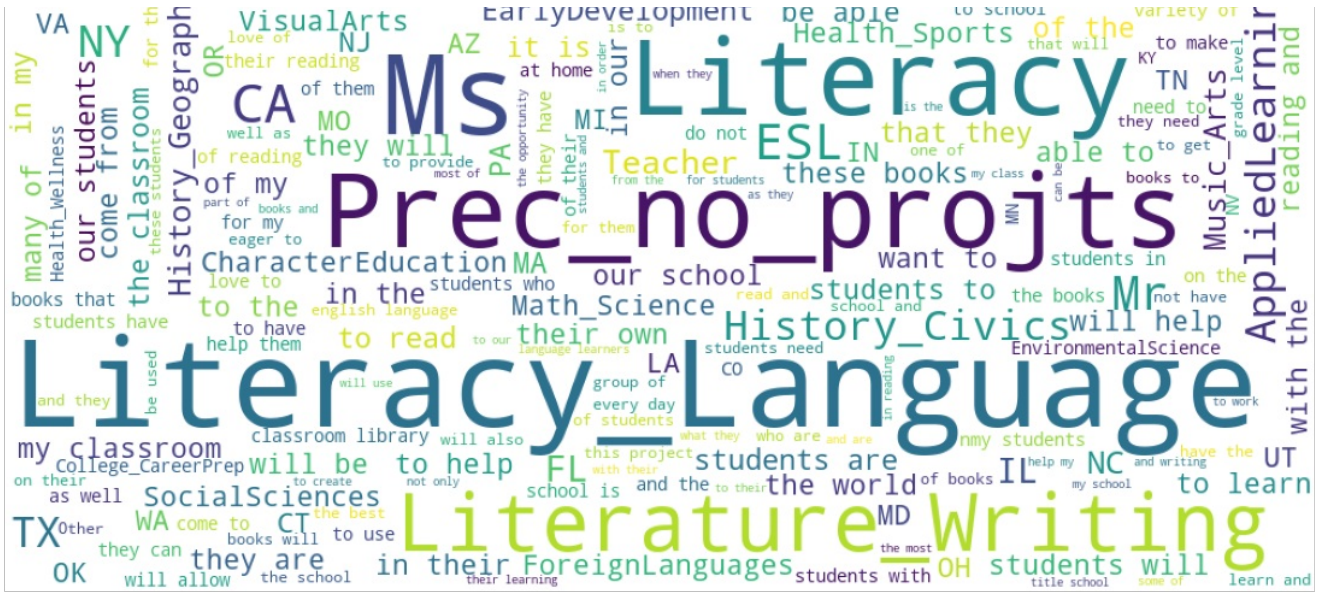
```
#Bar plot with counts of points in each cluster
cls_plt = pd.DataFrame({'Cluster': range(0,8),'Count':CountStatus})

ax = cls_plt.plot.bar(x='Cluster', y='Count', rot=0)
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```

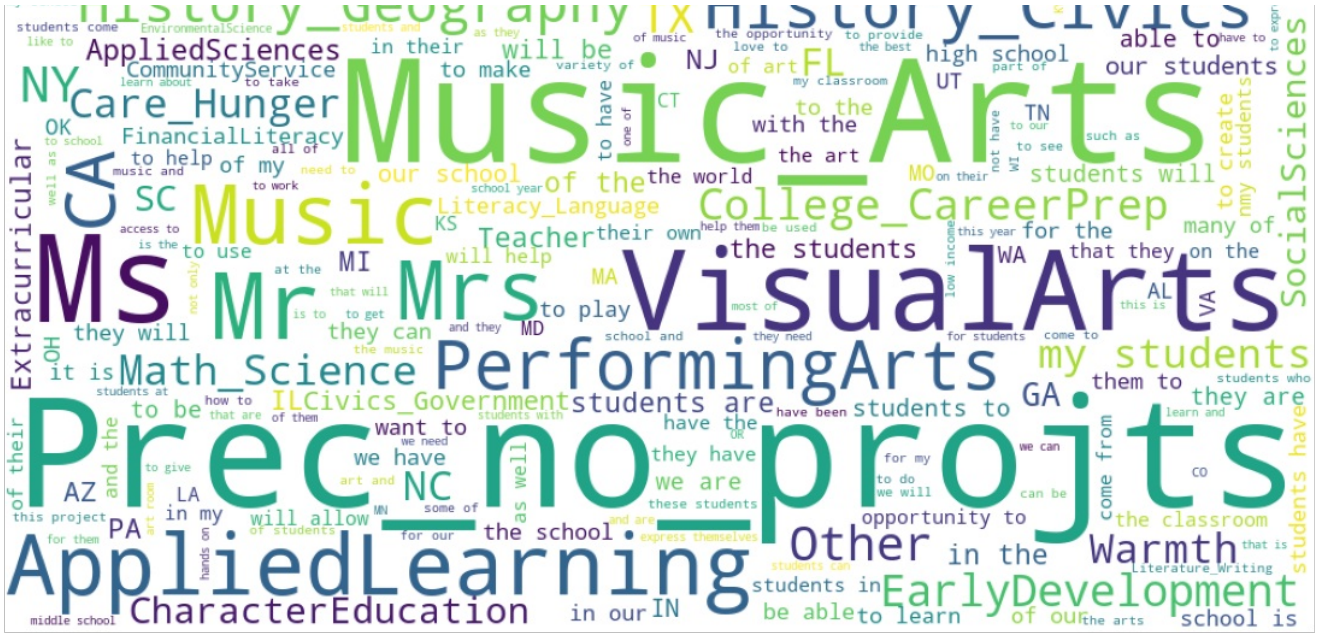


In [178]:

```
#https://stackoverflow.com/questions/43606339/generate-word-cloud-from-single-column-pandas-datafr
ame

from wordcloud import WordCloud

for i in range(0,8):
    clust = df_tfidf_5k_tr[df_tfidf_5k_tr['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```




Ms
Literacy
Prec_no_projts
Literacy_Language
Literature
Writing
Mr
Mrs
Special_Needs
Math_Science
Literacy_Language
Mathematics
Literacy
Math_Science
Literature_Writing
History_Civics


Music_Arts
Prec_no_projts
VisualArts
PerformingArts
AppliedLearning
Ms
Mr
Mrs
Music
Math_Science
College_CareerPrep
Care_Hunger
CommunityService
FinancialLiteracy
Literacy_Language
Teacher
Civics_Government
CharacterEducation
EarlyDevelopment
Other
Warmth
SocialSciences
Extracurricular
AppliedSciences
History_Geography
History_Civics
Literature_Writing
Mathematics
Literacy
Health_LifeScience
EnvironmentalScience
SpecialNeeds
Health_Sports
ForeignLanguages

In [179]:

```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Cluster based on"]

x_pretty_table.add_row([ "K-Means","Cluster 0",2702,'Arts and History'])
x_pretty_table.add_row([ "K-Means","Cluster 1",2128,'Physical education'])
x_pretty_table.add_row([ "K-Means","Cluster 2",2188,'Maths,Science and literature'])
x_pretty_table.add_row([ "K-Means","Cluster 3",2125,'Extracurricular activities and sports'])
x_pretty_table.add_row([ "K-Means","Cluster 4",2246,'Emotional intelligence and Social care'])
x_pretty_table.add_row([ "K-Means","Cluster 5",2447,'Maths and Science'])
x_pretty_table.add_row([ "K-Means","Cluster 6",2319,'Daily used words'])
x_pretty_table.add_row([ "K-Means","Cluster 7",1801,'Maths,Technology and literature'])


print(x_pretty_table)
```

```
+------------+-----------+-------------+-----------------------------------------+
| Model Type |  Cluster  | Data points |            Cluster based on             |
+------------+-----------+-------------+-----------------------------------------+
|  K-Means   | Cluster 0 |     2702    |            Arts and History             |
|  K-Means   | Cluster 1 |     2128    |           Physical education            |
|  K-Means   | Cluster 2 |     2188    |      Maths,Science and literature       |
|  K-Means   | Cluster 3 |     2125    | Extracurricular activities and sports   |
|  K-Means   | Cluster 4 |     2246    | Emotional intelligence and Social care  |
|  K-Means   | Cluster 5 |     2447    |           Maths and Science             |
|  K-Means   | Cluster 6 |     2319    |            Daily used words             |
|  K-Means   | Cluster 7 |     1801    |    Maths,Technology and literature      |
+------------+-----------+-------------+-----------------------------------------+
```

2.6 Apply AgglomerativeClustering

In [180]:

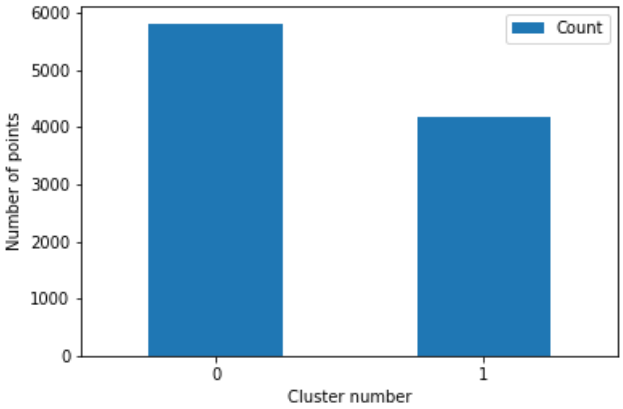
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [181]:

```
from sklearn.cluster import AgglomerativeClustering

amodel2 = AgglomerativeClustering(n_clusters = 2 ,affinity='euclidean',linkage='ward')
amodel2.fit(df_tfidf_5k[0:10000])

# Getting the cluster labels
clust_labels2 = amodel2.labels_
```

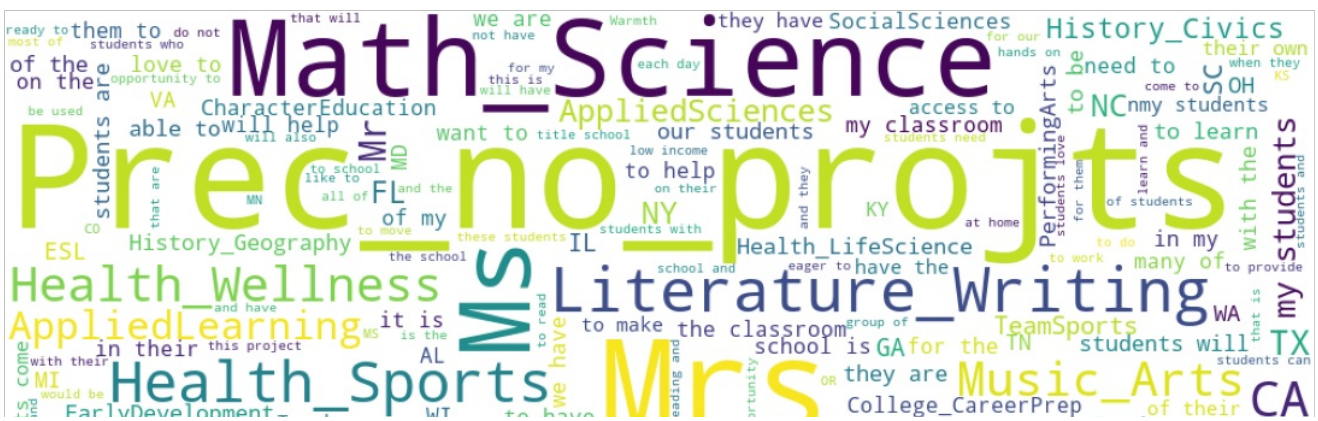
In [182]:

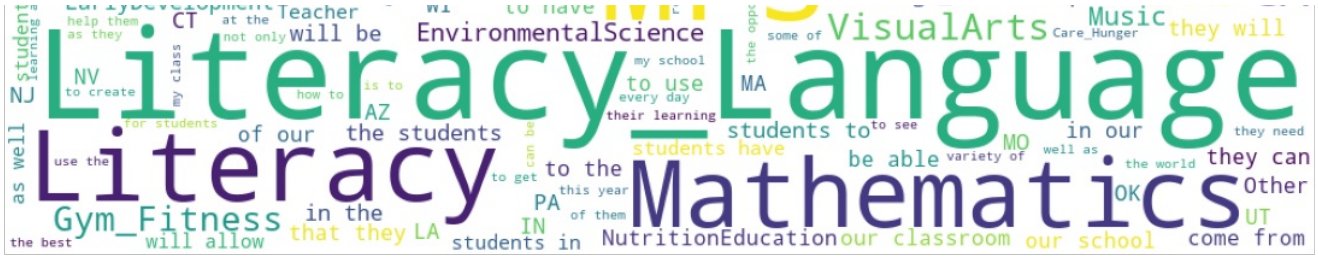
```
df_tfidf_5k_tr_a = df_tfidf_5k[:10000]

df_tfidf_5k_tr_a['clusters'] = clust_labels2

#Data Points per cluster.
CountStatus2 = pd.value_counts(df_tfidf_5k_tr_a['clusters'].values, sort=False)

#Bar plot with counts of points in each cluster
clus_agg_2 = pd.DataFrame({'Cluster': range(0,2),'Count':CountStatus2})

ax = clus_agg_2.plot.bar(x='Cluster', y='Count', rot=0)
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```



In [183]:

```
for i in range(0,2):
    clust = df_tfidf_5k_tr_a[df_tfidf_5k_tr_a['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```





In [184]:

```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Cluster based on"]

x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 0",5132,'Words related to special needs'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 1",4868,'Regularly used words in daily student life'])

print(x_pretty_table)
```

```
+--------------------------+-----------+-------------+-----------------------------------------+
|        Model Type        |  Cluster  | Data points |              Cluster based on           |
+--------------------------+-----------+-------------+-----------------------------------------+
| Agglomerative Clustering | Cluster 0 |     5132    |       Words related to special needs    |
| Agglomerative Clustering | Cluster 1 |     4868    | Regularly used words in daily student life |
+--------------------------+-----------+-------------+-----------------------------------------+
```

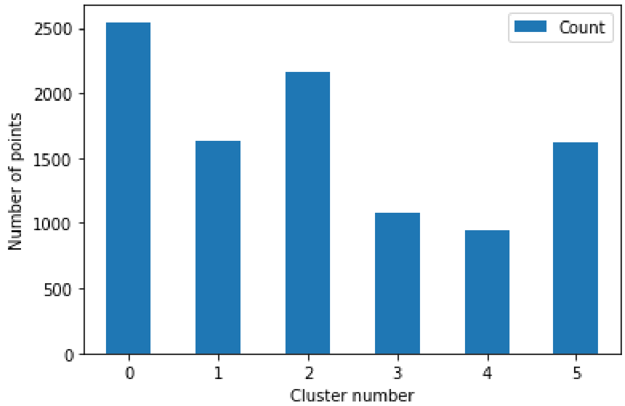
In [185]:

```
aggmodel_6 = AgglomerativeClustering(n_clusters = 6 ,affinity='euclidean',linkage='ward')
aggmodel_6.fit(df_tfidf_5k[0:10000])

# Getting the cluster labels
clust_labels6 = aggmodel_6.labels_
```

In [186]:

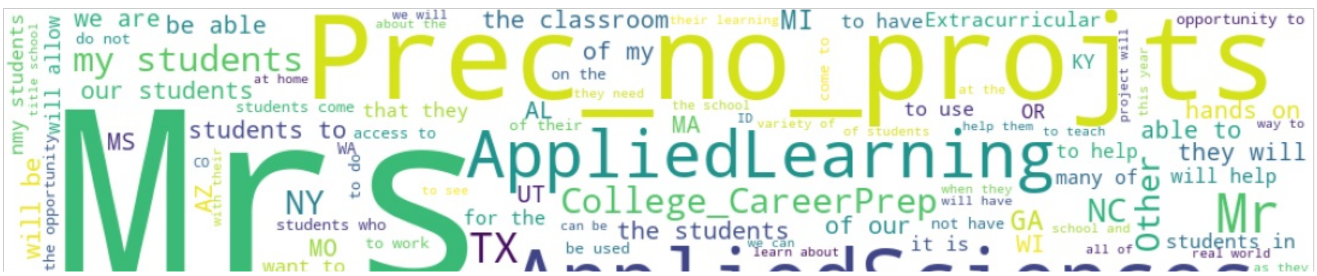
```
df_tfidf_5k_tr_a['clusters'] = clust_labels6

#Data Points per cluster.
CountStatus6 = pd.value_counts(df_tfidf_5k_tr_a['clusters'].values, sort=False)
```

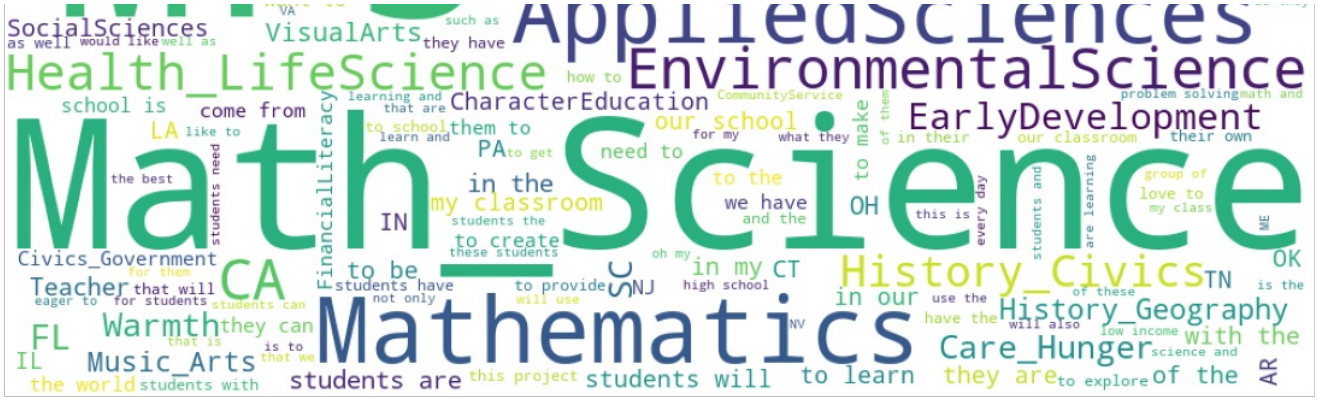
```
#Bar plot with counts of points in each cluster
t = pd.DataFrame({'Cluster': range(0,6),'Count':CountStatus6})

ax = t.plot.bar(x='Cluster', y='Count', rot=0)
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```



In [187]:

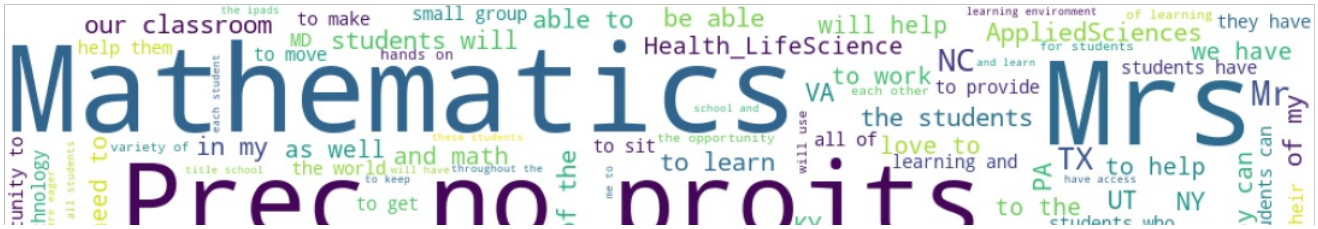
```
for i in range(0,6):
    clust = df_tfidf_5k_tr_a[df_tfidf_5k_tr_a['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```

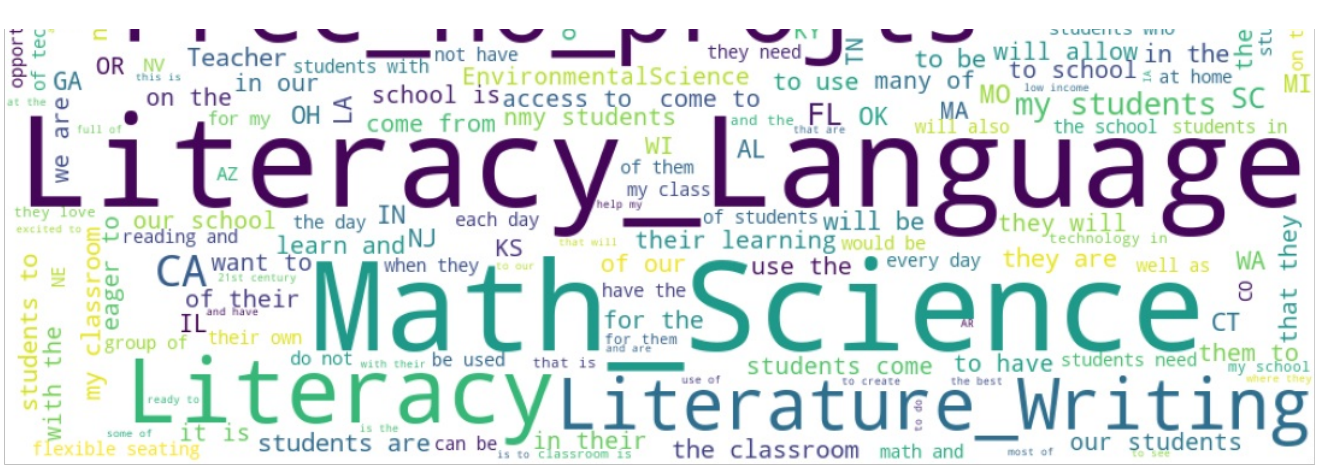



Math_Science
Mathematics
AppliedSciences
EnvironmentalScience
Health_LifeScience
EarlyDevelopment
CharacterEducation
SocialSciences
VisualArts
History_Civics
History_Geography
Care_Hunger
Music_Arts
Civics_Government
Teacher
Warmth
Financialliteracy
CA
FL
IL
AR
OK
TN


Prec_no_projts
Literacy_Language
Math_Science
Mathematics
Music_Arts
AppliedLearning
History_Civics
History_Geography
PerformingArts
CharacterEducation
College_CareerPrep
Care_Hunger
SocialSciences
Music
Literacy
Health_LifeScience
VisualArts
Literature_Writing
Mrs
Ms
CA
Teacher
Civics_Government
EnvironmentalScience
AppliedSciences
EarlyDevelopment


Prec_no_projts
NutritionEducation
Mrs
Gym_Fitness
Health_Wellness
Health_Sports
Ms
Mr
TeamSports
AppliedLearning
CharacterEducation
Math_Science
Health_LifeScience
EarlyDevelopment
CA


Mathematics
Mrs
Prec_no_projts
Health_LifeScience
AppliedSciences
Mr



In [188]:

```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Cluster based on"]

x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 0",2267,'Maths and Social studies'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 1",2601,'Maths and Science'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 2",1192,'Fitness and Sports'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 3",1215,'Regularly used words in
daily student life'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 4",1497,'Arts and Science'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 5",1228,'Words related to special
needs '])

print(x_pretty_table)
```

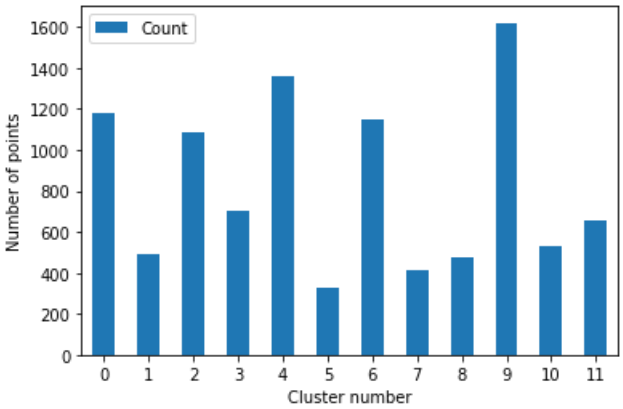
Model Type	Cluster	Data points	Cluster based on
Agglomerative Clustering	Cluster 0	2267	Maths and Social studies
Agglomerative Clustering	Cluster 1	2601	Maths and Science
Agglomerative Clustering	Cluster 2	1192	Fitness and Sports
Agglomerative Clustering	Cluster 3	1215	Regularly used words in daily student life
Agglomerative Clustering	Cluster 4	1497	Arts and Science
Agglomerative Clustering	Cluster 5	1228	Words related to special needs

In [189]:

```
aggclus_12 = AgglomerativeClustering(n_clusters = 12 ,affinity='euclidean',linkage='ward')
aggclus_12.fit(df_tfidf_5k[0:10000])

# Getting the cluster labels
clust_labels12 = aggclus_12.labels_
```

In [190]:

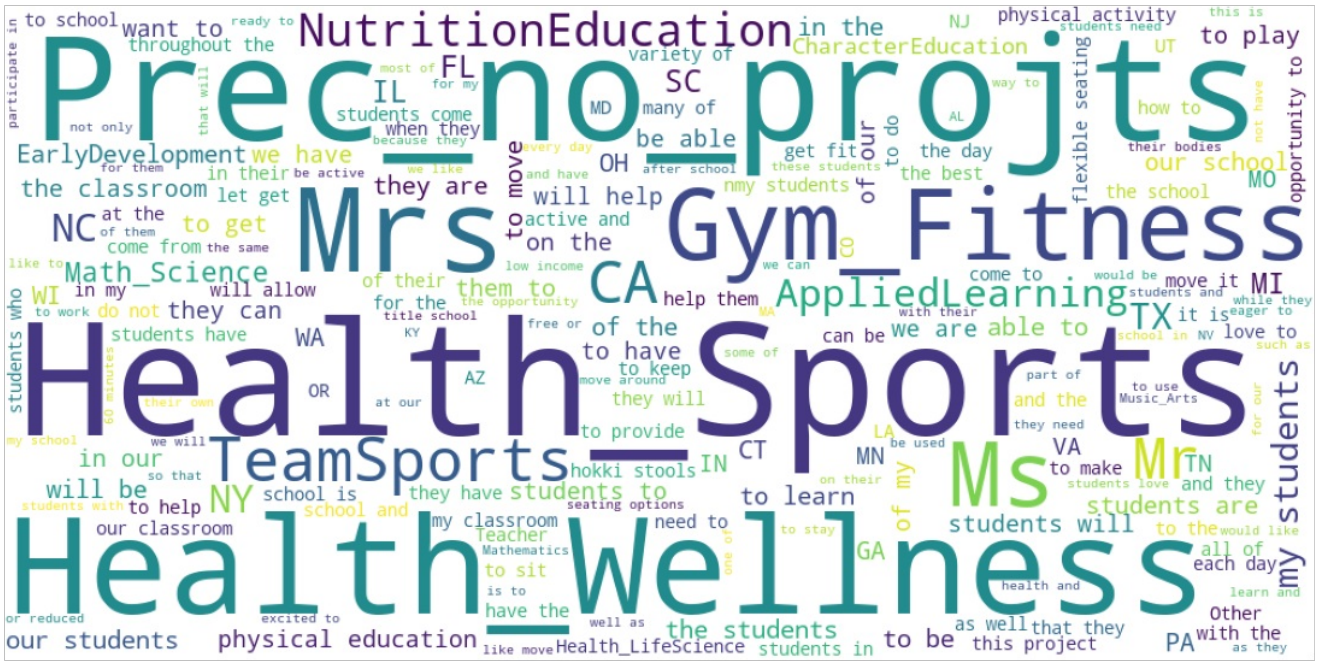
```
df_tfidf_5k_tr_a['clusters'] = clust_labels12

#Data Points per cluster.
CountStatus12 = pd.value_counts(df_tfidf_5k_tr_a['clusters'].values, sort=False)

#Bar plot with counts of points in each cluster
clust_12 = pd.DataFrame({'Cluster': range(0,12),'Count':CountStatus12})

ax = clust_12.plot.bar(x='Cluster', y='Count', rot=0)
plt.xlabel("Cluster number")
plt.ylabel("Number of points")
plt.show()
```

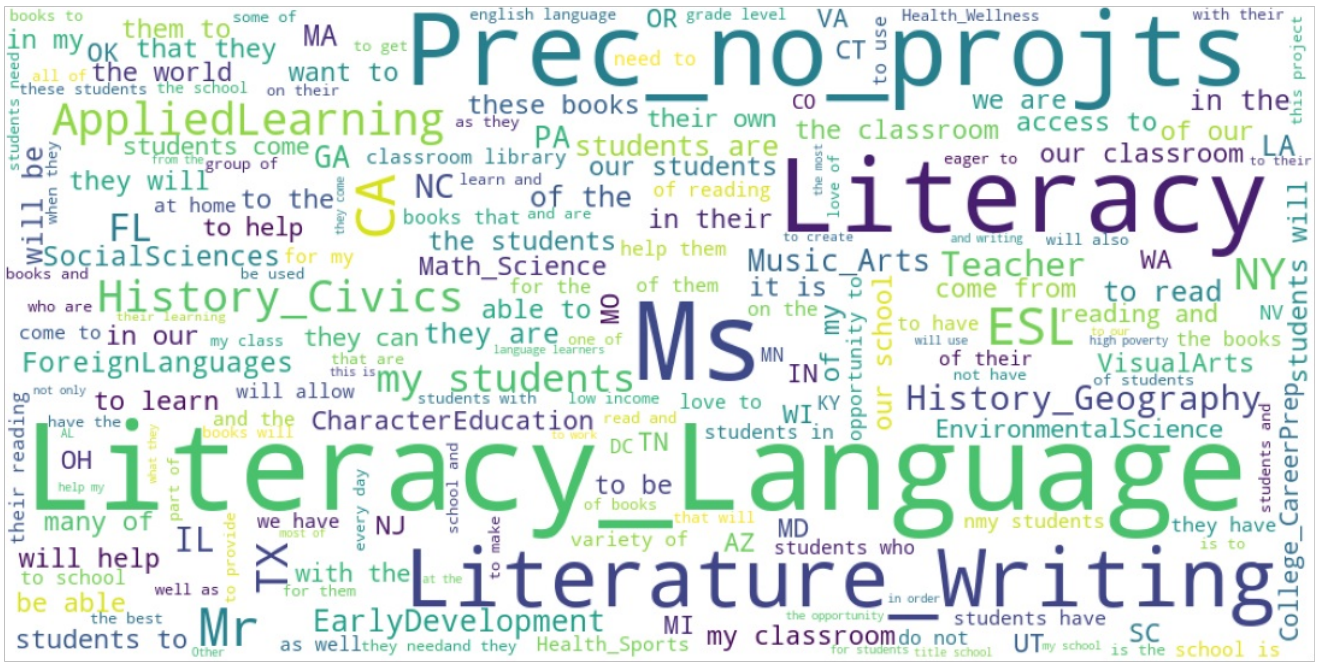


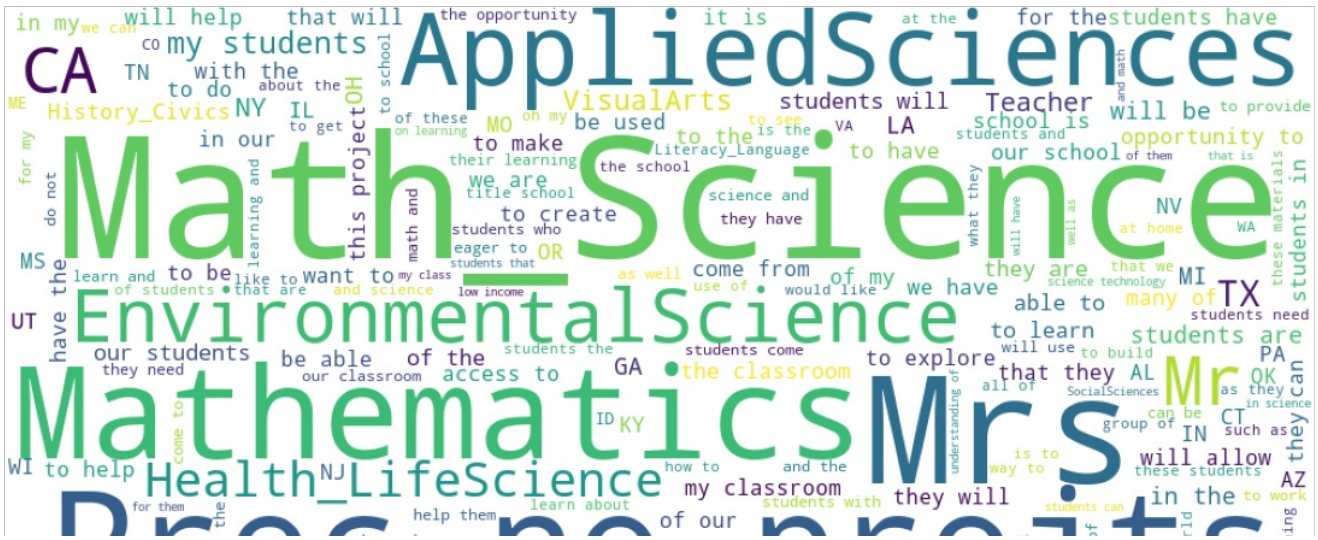
In [191]:

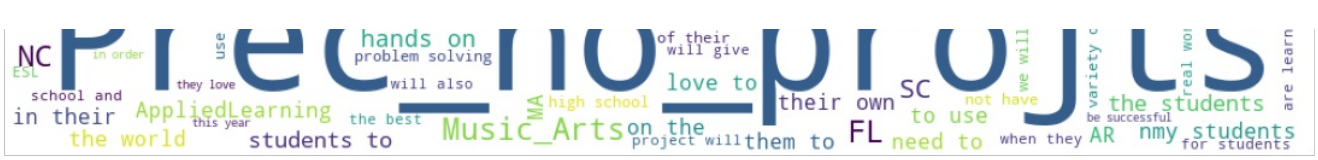
```
for i in range(0,12):
    clust = df_tfidf_5k_tr_a[df_tfidf_5k_tr_a['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```

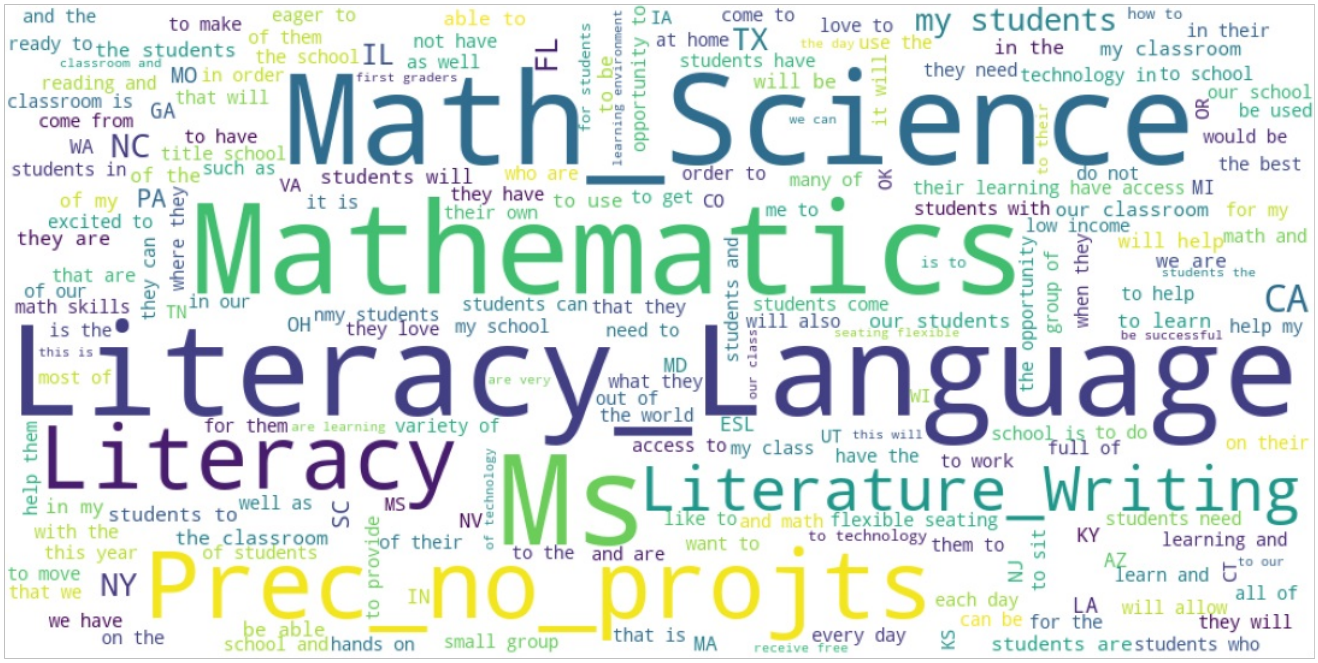

AppliedLearning
EarlyDevelopment
my students
them to
will help
special education
the students
NY

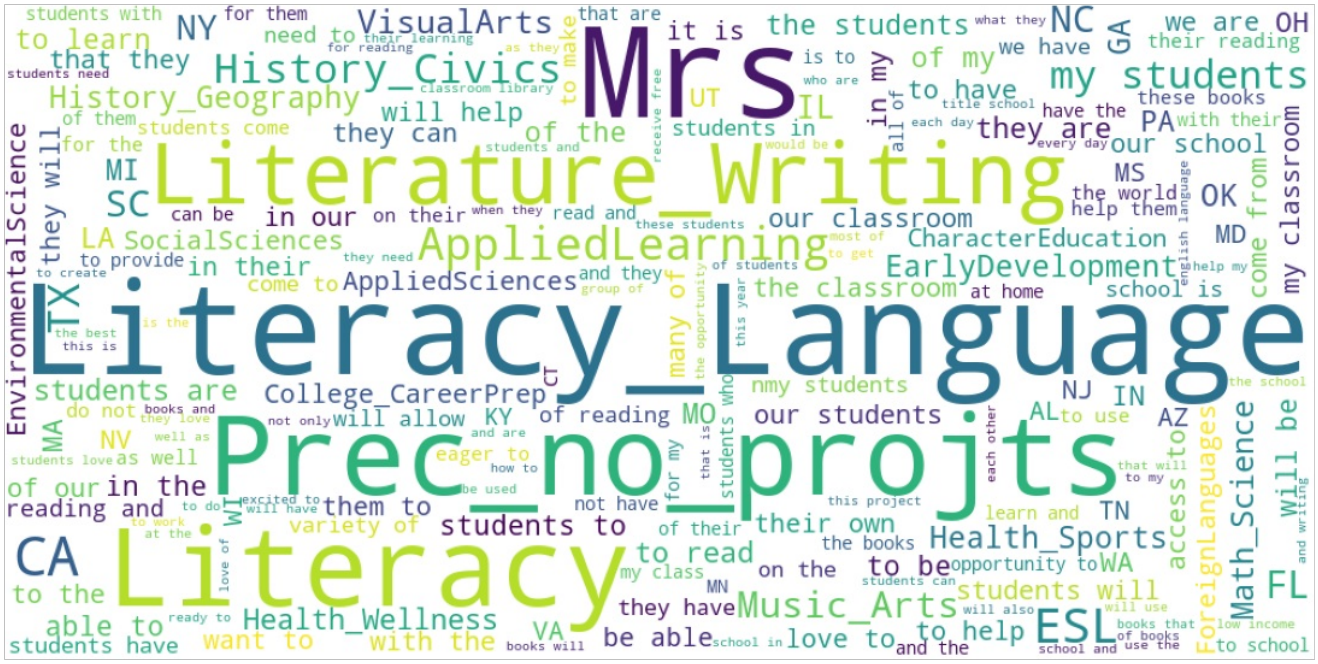

Prec_no_projts
Mrs_Gym_Fitness
Health_Sports
Health_Wellness
TeamSports_
Ms
Mr
NutritionEducation
CharacterEducation
AppliedLearning
EarlyDevelopment
Math_Science
physical education
my students

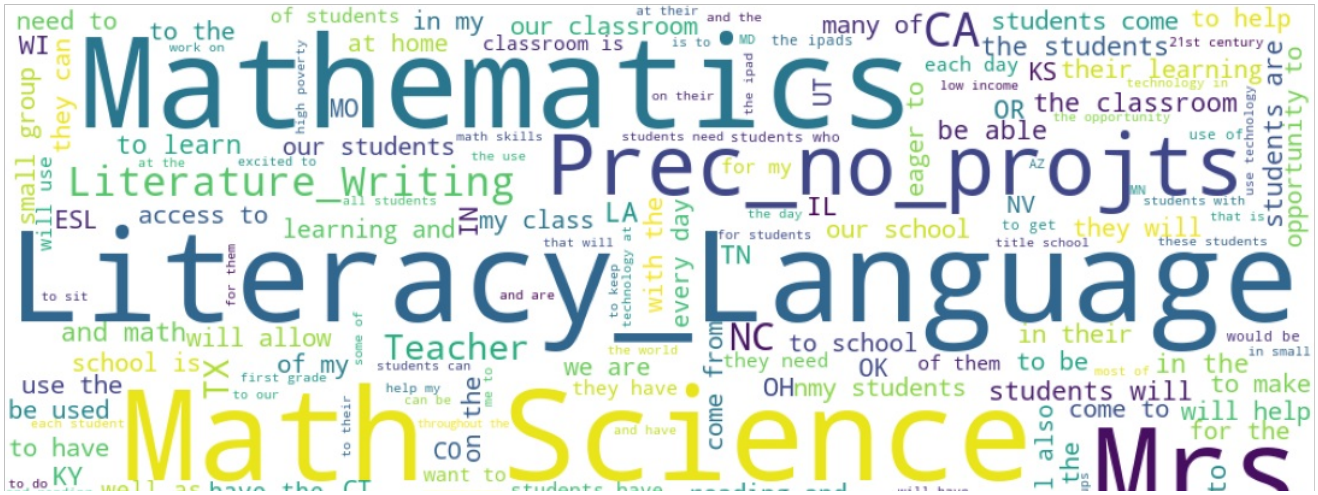

Prec_no_projts
Literacy_Language
Literature_Writing
Literacy
Ms
AppliedLearning
History_Civics
SocialSciences
ForeignLanguages
Math_Science
Music_Arts
Teacher
ESL
VisualArts
History_Geography
EnvironmentalScience
CharacterEducation
College_CareerPrep
Mr
EarlyDevelopment


AppliedSciences
Math_Science
EnvironmentalScience
Mathematics
Mrs
Health_LifeScience
Prec_no_projts
my students
VisualArts
History_Civics
Teacher
Mr

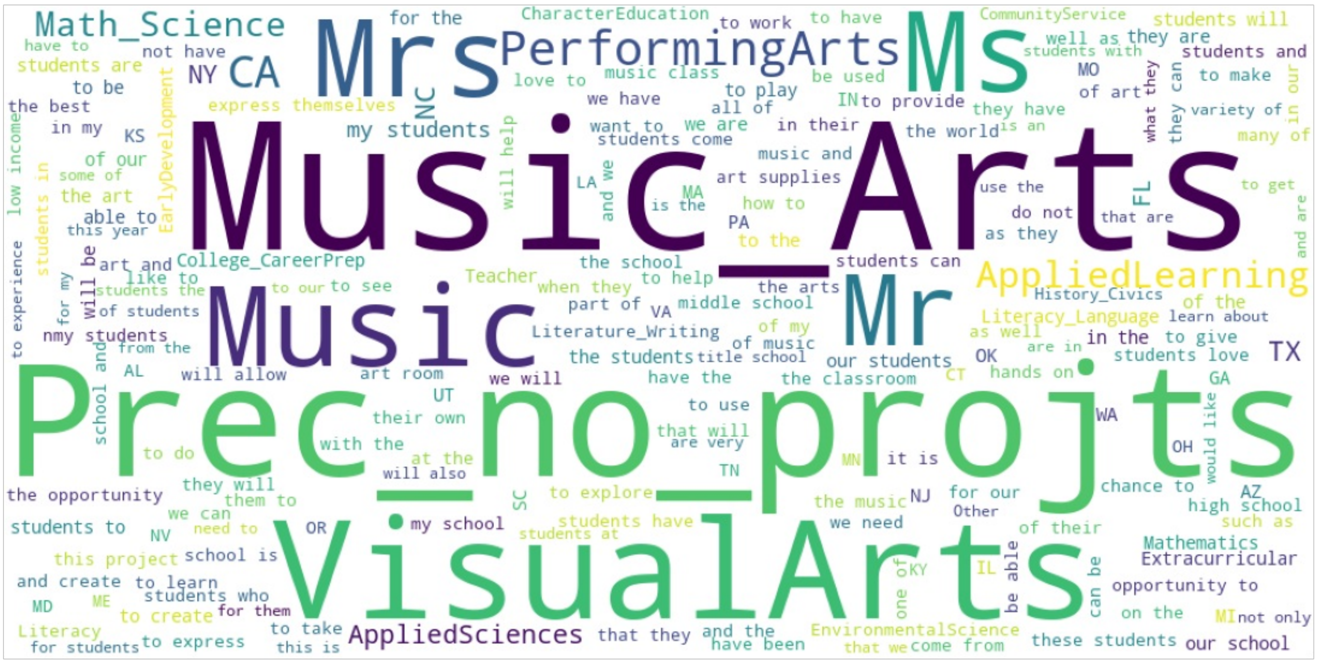

Prec_no_projts
Music_Arts

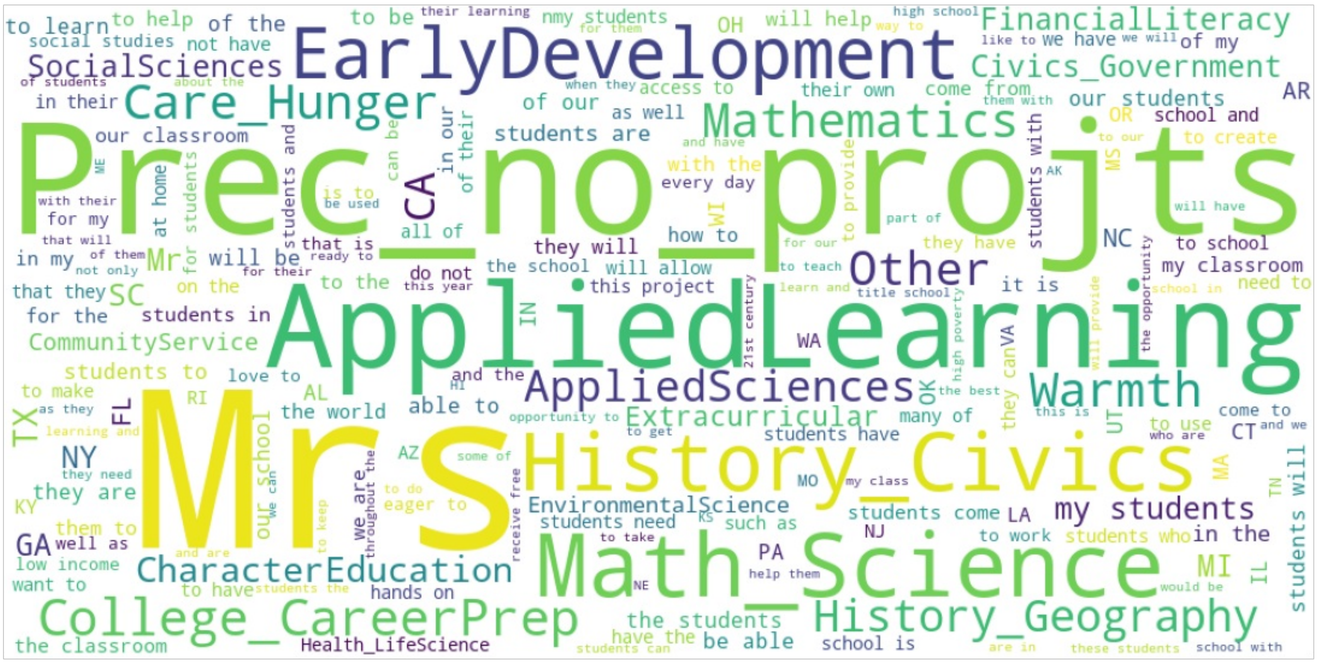

Math_Science
Mathematics
Literacy_Language
Literacy
Ms
Literature_Writing
Prec_no_projts

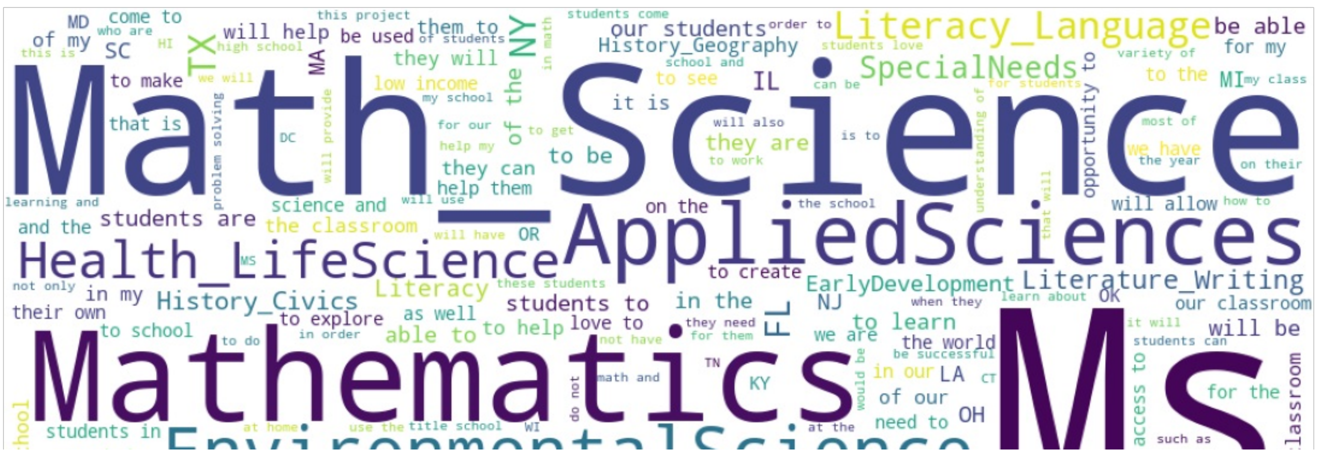

Mrs
Literature_Writing
Literacy_Language
Prec_no_projts
Literacy
AppliedLearning
History_Civics
VisualArts
Music_Arts
Health_Sports
College_CareerPrep

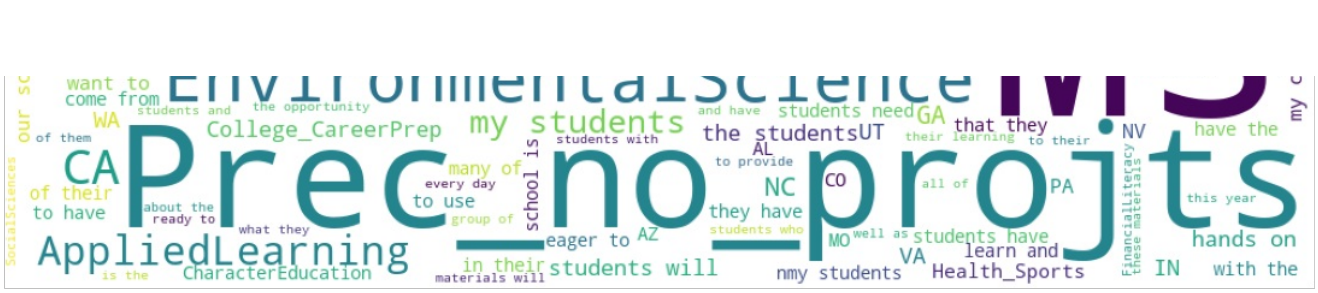

Mathematics
Prec_no_projts
Literacy_Language
Literature_Writing
Math_Science
Mrs

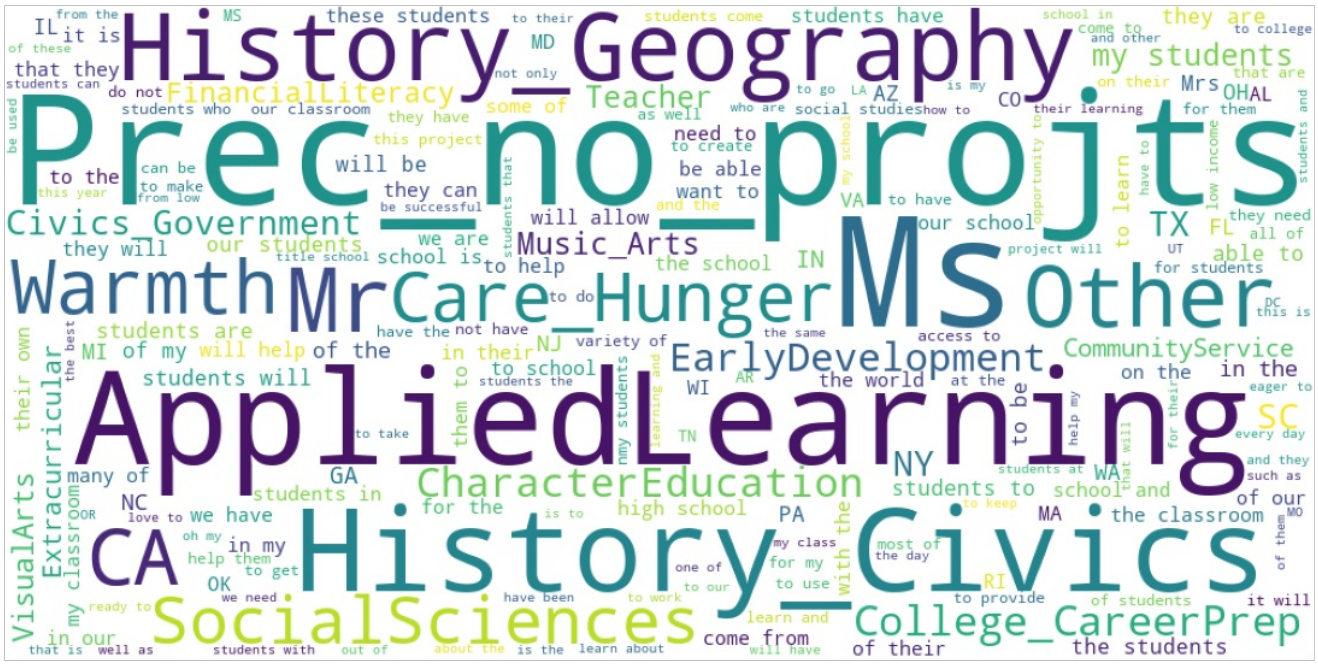

Literacy
Mr
my students
my classroom

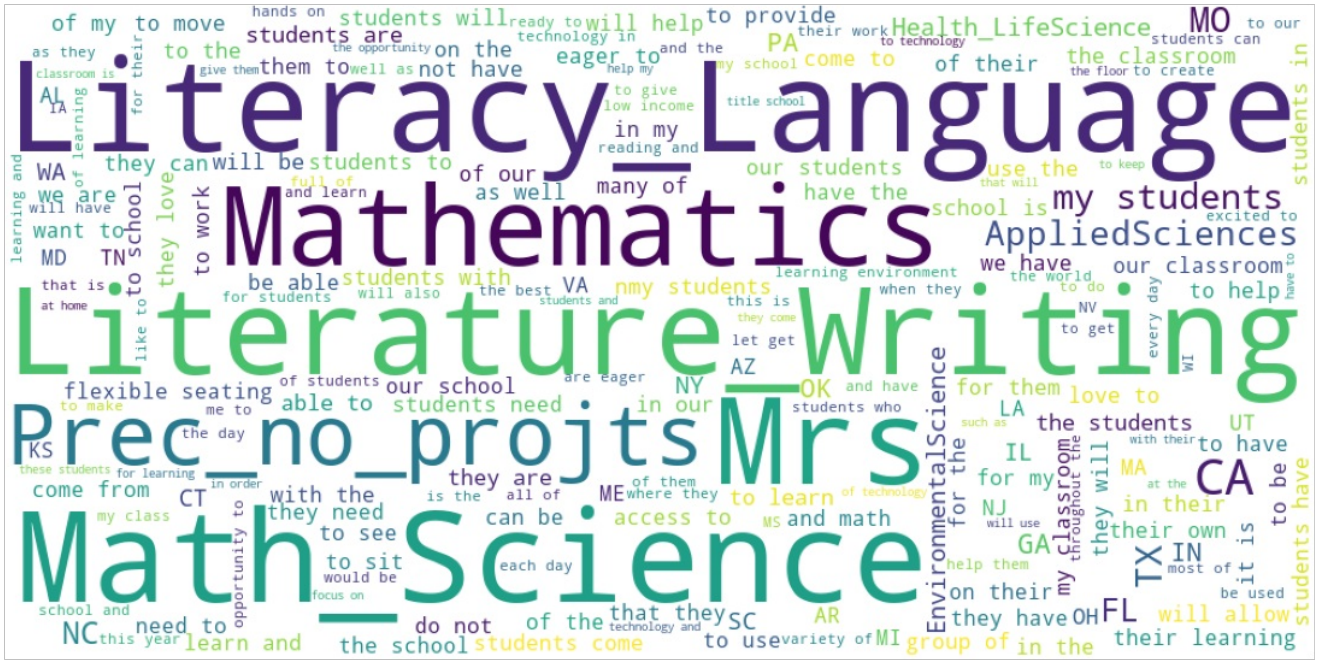

Music_Arts
Prec_no_projts
VisualArts
Music
Mrs
Ms
Mr
PerformingArts
AppliedLearning
Math_Science
AppliedSciences


Prec_no_projts
AppliedLearning
Mrs
EarlyDevelopment
History_Civics
Math_Science
College_CareerPrep
History_Geography
CharacterEducation
AppliedSciences
Warmth
Other
Mathematics
Care_Hunger
SocialSciences
FinancialLiteracy
Civics_Government
Extracurricular
EnvironmentalScience
CommunityService
Health_LifeScience


Math_Science
AppliedSciences
Mathematics
Ms
Health_LifeScience
Literacy_Language
SpecialNeeds
EarlyDevelopment
Literature_Writing
History_Civics
History_Geography







In [192]:

```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Cluster based on"]

x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 0",1228,'Regularly used words in
daily student life '])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 1",573,'Maths,Science and Literature'
])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 2",992,'Arts and Literatur'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 3",141,'Maths and science'])
```

```
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 4",1275,'Literature and History'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 5",694,'Maths,Science and Literature'
])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 6",1351,'Sports and fitness'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 7",712,'Personal Development'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 8",1497,'Music and arts'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 9",521,'Care and humanity'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 10",480,'Maths and Science'])
x_pretty_table.add_row([ "Agglomerative Clustering","Cluster 11",536,'Health and Wellness'])

print(x_pretty_table)
```

```
+--------------------------+------------+-------------+-----------------------------------------
|        Model Type        |  Cluster   | Data points |               Cluster based on
|
+--------------------------+------------+-------------+-----------------------------------------
| Agglomerative Clustering | Cluster 0  |     1228    | Regularly used words in daily student life
|
| Agglomerative Clustering | Cluster 1  |     573     |         Maths,Science and Literature
|
| Agglomerative Clustering | Cluster 2  |     992     |              Arts and Literatur
|
| Agglomerative Clustering | Cluster 3  |     141     |              Maths and science
|
| Agglomerative Clustering | Cluster 4  |     1275    |            Literature and History
|
| Agglomerative Clustering | Cluster 5  |     694     |         Maths,Science and Literature
|
| Agglomerative Clustering | Cluster 6  |     1351    |              Sports and fitness
|
| Agglomerative Clustering | Cluster 7  |     712     |             Personal Development
|
| Agglomerative Clustering | Cluster 8  |     1497    |                Music and arts
|
| Agglomerative Clustering | Cluster 9  |     521     |              Care and humanity
|
| Agglomerative Clustering | Cluster 10 |     480     |              Maths and Science
|
| Agglomerative Clustering | Cluster 11 |     536     |             Health and Wellness
|
+--------------------------+------------+-------------+-----------------------------------------
```

2.7 Apply DBSCAN

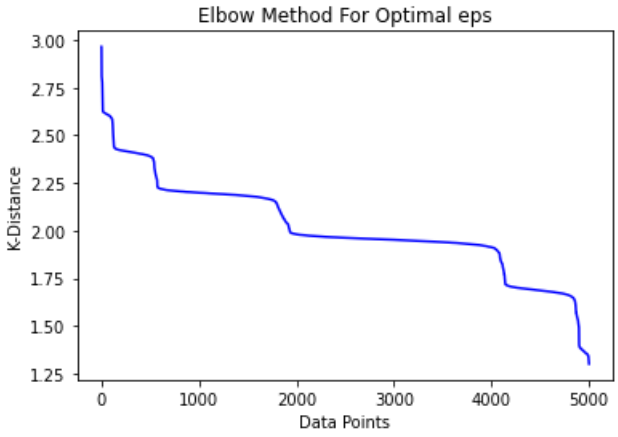
In [193]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [194]:

```
#https://stackoverflow.com/questions/48010276/how-to-estimate-eps-using-knn-distance-plot-in-dbsca
n

from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors

ns = 8
nbrs = NearestNeighbors(n_neighbors=ns).fit(df_tfidf_5k[0:5000])
distances, indices = nbrs.kneighbors(df_tfidf_5k[0:5000])
distanceDec = sorted(distances[:,ns-1], reverse=True)

plt.plot(list(range(1,5000+1)), distanceDec,'b-')

plt.xlabel('Data Points')
```

```
plt.ylabel('K-Distance')
plt.title('Elbow Method For Optimal eps')
plt.show()
```



In [195]:

```
#eps=2.2

dmodel = DBSCAN(eps=2.2, min_samples=15, metric='euclidean',n_jobs= -1)

dmodel.fit(df_tfidf_5k[0:5000])

# Getting the cluster labels
clust_labels_d = dmodel.labels_
```

In [196]:

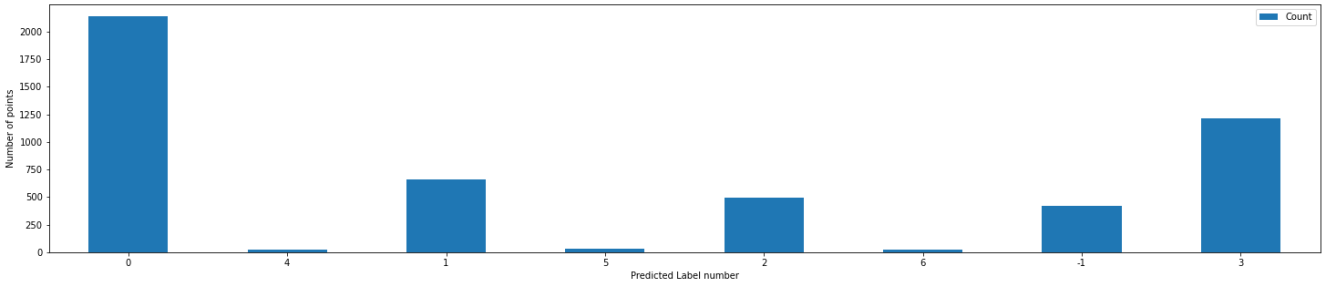
```
df_tfidf_5k_tr_d = df_tfidf_5k[0:5000]

df_tfidf_5k_tr_d['clusters'] = clust_labels_d

#Data Points per cluster.
CountStatus_d = pd.value_counts(df_tfidf_5k_tr_d['clusters'].values, sort=False)
```

In [197]:

```
labels = np.array(list(dict(CountStatus_d).keys()))
count  = np.array(list(dict(CountStatus_d).values()))

clust_dbs = pd.DataFrame(labels,columns=['Label'])
clust_dbs['Count'] = pd.DataFrame(count,columns=['Count'])
```

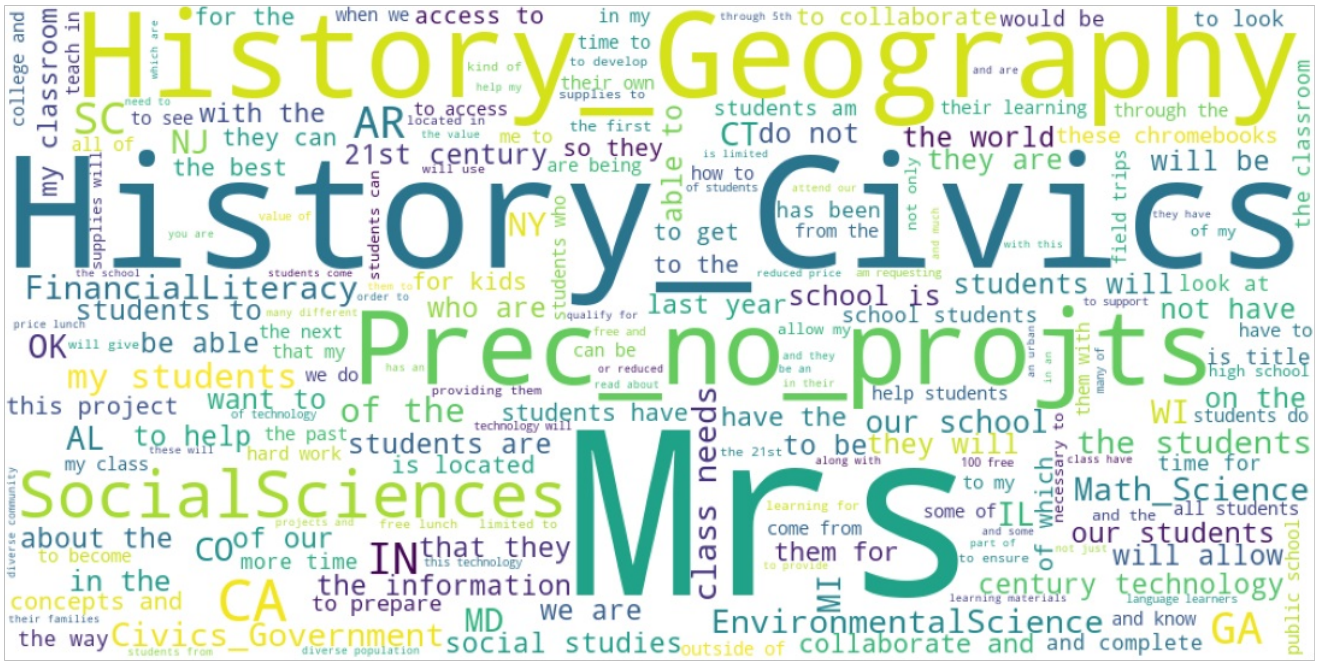
In [198]:

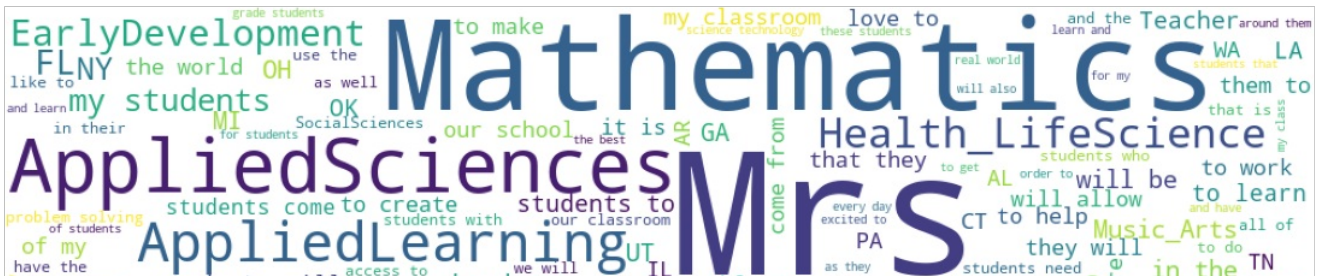
```
#Bar plot with counts of points in each cluster

ax = clust_dbs.plot.bar(x='Label', y='Count', rot=0,figsize=(25,5))
plt.xlabel("Predicted Label number")
plt.ylabel("Number of points")
plt.show()
```

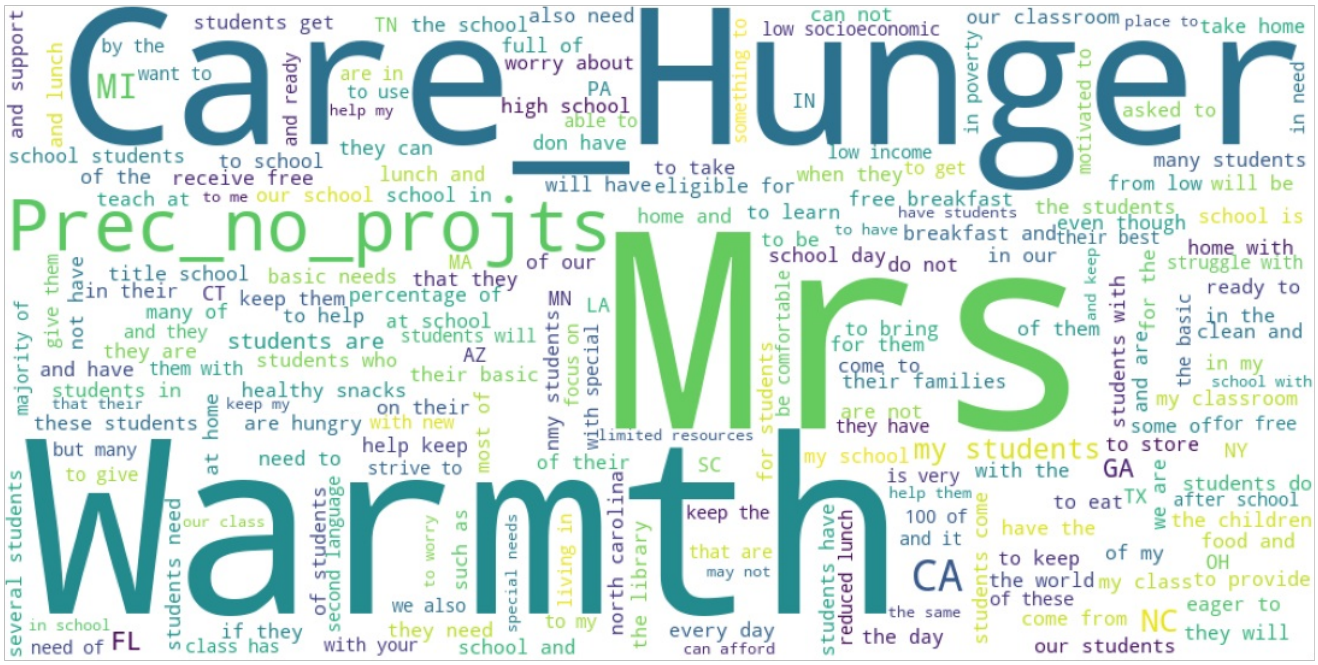


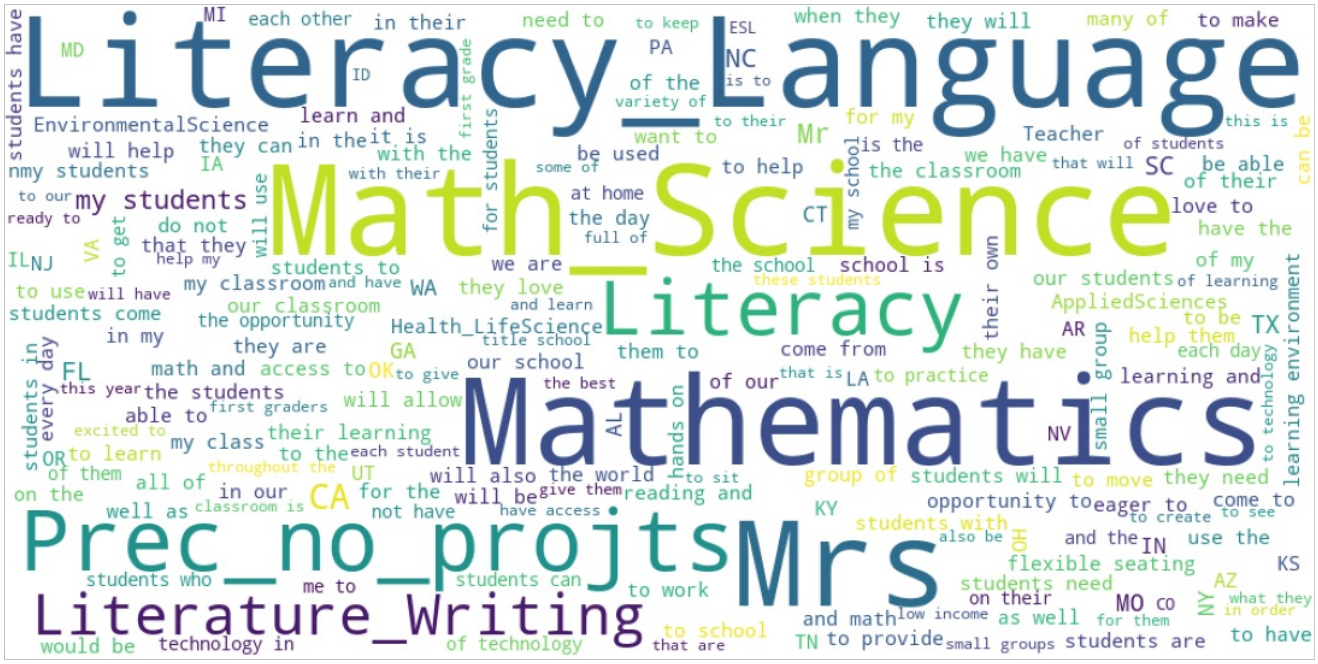
In [199]:

```
from wordcloud import WordCloud

for i in labels[0:5]:
    clust = df_tfidf_5k_tr_d[df_tfidf_5k_tr_d['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```









In [200]:

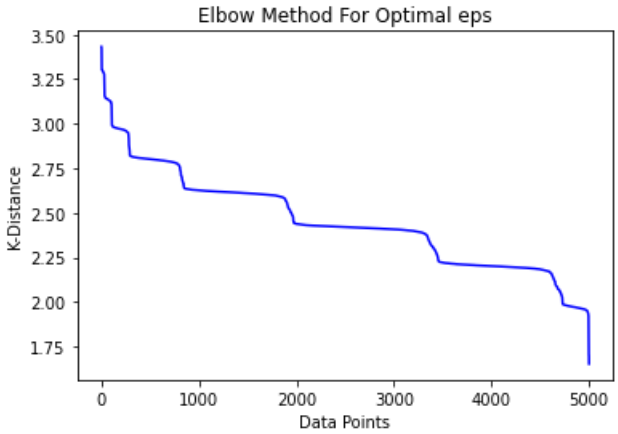
```
from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Eps","Cluster based on"]

x_pretty_table.add_row([ "DB Scan","Cluster 0",4394,2.2,'Regularly used words in daily student
```

```
life '])
x_pretty_table.add_row([ "DB Scan","Cluster 1",49,2.2,'Care and humanity'])
x_pretty_table.add_row([ "DB Scan","Cluster 2",14,2.2,'Arts and science'])
x_pretty_table.add_row([ "DB Scan","Cluster -1",543,2.2,'Regularly used words in daily student
life  and applied science'])
print(x_pretty_table)
```

```
+------------+------------+-------------+-----+-----------------------------------------------------
----------+
| Model Type |  Cluster   | Data points | Eps |                          Cluster based on
|
+------------+------------+-------------+-----+-----------------------------------------------------
----------+
|  DB Scan   | Cluster 0  |     4394    | 2.2 |           Regularly used words in daily student lif
e            |
|  DB Scan   | Cluster 1  |      49     | 2.2 |                         Care and humanity
|
|  DB Scan   | Cluster 2  |      14     | 2.2 |                          Arts and science
|
|  DB Scan   | Cluster -1 |     543     | 2.2 | Regularly used words in daily student life  and app
lied science |
+------------+------------+-------------+-----+-----------------------------------------------------
----------+
```

In [201]:

```
#https://stackoverflow.com/questions/48010276/how-to-estimate-eps-using-knn-distance-plot-in-dbsca
n

from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors

ns = 100
nbrs = NearestNeighbors(n_neighbors=ns).fit(df_tfidf_5k[0:5000])
distances, indices = nbrs.kneighbors(df_tfidf_5k[0:5000])
distanceDec = sorted(distances[:,ns-1], reverse=True)

plt.plot(list(range(1,5000+1)), distanceDec,'b-')

plt.xlabel('Data Points')
plt.ylabel('K-Distance')
plt.title('Elbow Method For Optimal eps')
plt.show()
```



In [202]:

```
#eps=2.5

dmodel = DBSCAN(eps=2.5, min_samples=20, metric='euclidean',n_jobs= -1)

dmodel.fit(df_tfidf_5k[0:5000])

# Getting the cluster labels
clust_labels_d = dmodel.labels_
```

In [203]:

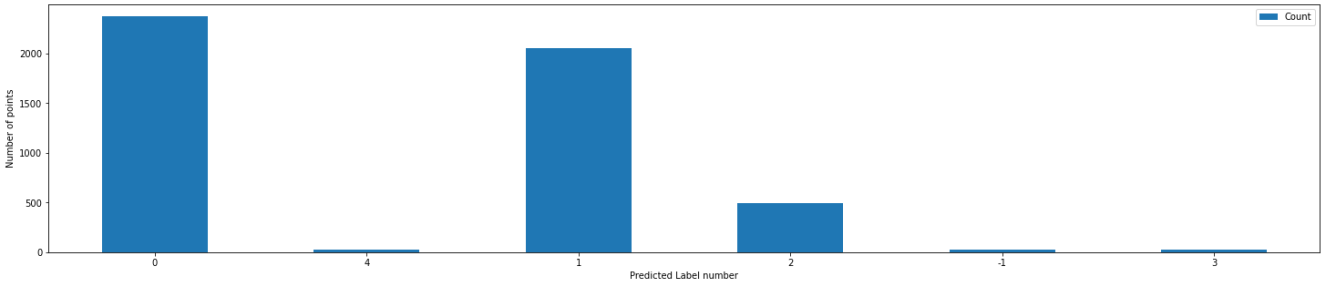
```
df_tfidf_5k_tr_d = df_tfidf_5k[0:5000]

df_tfidf_5k_tr_d['clusters'] = clust_labels_d

#Data Points per cluster.
CountStatus_d = pd.value_counts(df_tfidf_5k_tr_d['clusters'].values, sort=False)
```

In [204]:

```
labels = np.array(list(dict(CountStatus_d).keys()))
count  = np.array(list(dict(CountStatus_d).values()))

clust_dbs = pd.DataFrame(labels,columns=['Label'])
clust_dbs['Count'] = pd.DataFrame(count,columns=['Count'])
```

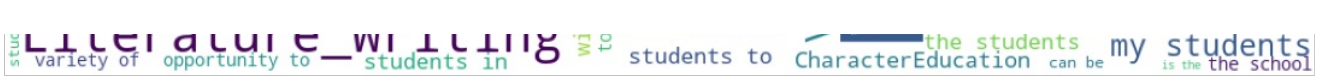
In [205]:

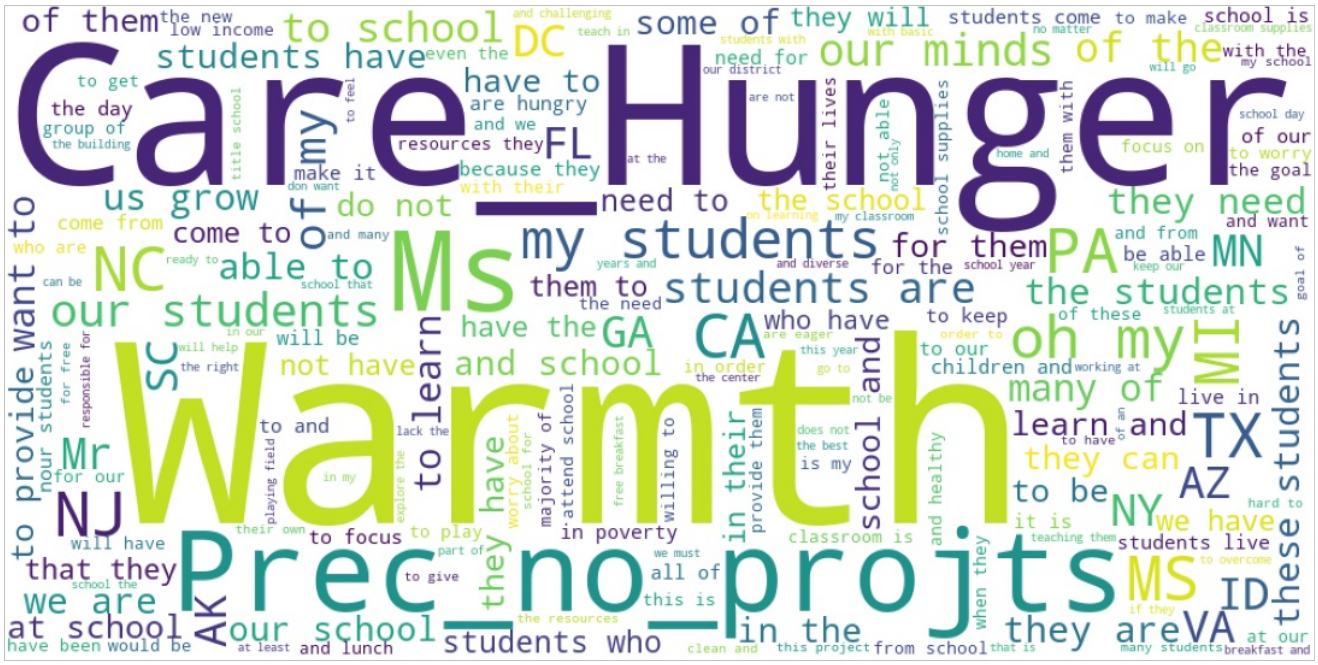
```
#Bar plot with counts of points in each cluster

ax = clust_dbs.plot.bar(x='Label', y='Count', rot=0,figsize=(25,5))
plt.xlabel("Predicted Label number")
plt.ylabel("Number of points")
plt.show()
```



In [206]:

```
for i in labels[0:3]:
    clust = df_tfidf_5k_tr_d[df_tfidf_5k_tr_d['clusters'] == i].drop(['clusters'],axis=1)
    tp_freq = (clust.sum()).to_dict()
    wordcloud = WordCloud(width = 1000, height = 500, background_color
='white').generate_from_frequencies(tp_freq)
    plt.figure(figsize=(25,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
    plt.close()
```





In [207]:

```
from prettytable import PrettyTable

x_pretty_table = PrettyTable()
x_pretty_table.field_names = ["Model Type","Cluster","Data points","Eps","Cluster based on"]

x_pretty_table.add_row([ "DB Scan","Cluster 0",4923,2.5,'Regularly used words in daily student
life '])
x_pretty_table.add_row([ "DB Scan","Cluster 1",53,2.5,'Care and humanity'])
x_pretty_table.add_row([ "DB Scan","Cluster -1",24,2.5,'Arts and science'])
print(x_pretty_table)
```

```
+------------+------------+-------------+-----+-----------------------------------------+
| Model Type |  Cluster   | Data points | Eps |              Cluster based on           |
+------------+------------+-------------+-----+-----------------------------------------+
|  DB Scan   | Cluster 0  |     4923    | 2.5 | Regularly used words in daily student life  |
|  DB Scan   | Cluster 1  |      53     | 2.5 |              Care and humanity          |
|  DB Scan   | Cluster -1 |      24     | 2.5 |               Arts and science          |
+------------+------------+-------------+-----+-----------------------------------------+
```

3. Cocnlusions

Please write down few lines of your observations on this assignment.