In [1]:
```
#Group 8 - Wk6 - Clustering
#Group Member: Athena Zhang, Pratheek Praveen Kumar, Weifeng Li, Wenke Yu, Ziq
iao Wei
```

Make sure you installed ***sklearn***, ***matplotlib*** and ***numpy*** if you use your local machine

In [2]:
```
!pip install scikit-learn-extra
!pip install scipy
```

```
Requirement already satisfied: scikit-learn-extra in c:\users\student\anacond
a3\lib\site-packages (0.2.0)
Requirement already satisfied: scikit-learn>=0.23.0 in c:\users\student\anaco
nda3\lib\site-packages (from scikit-learn-extra) (1.0.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\student\anaconda3\li
b\site-packages (from scikit-learn-extra) (1.18.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\student\anaconda3\li
b\site-packages (from scikit-learn-extra) (1.4.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\student\anaco
nda3\lib\site-packages (from scikit-learn>=0.23.0->scikit-learn-extra) (3.1.
0)
Requirement already satisfied: joblib>=0.11 in c:\users\student\anaconda3\lib
\site-packages (from scikit-learn>=0.23.0->scikit-learn-extra) (0.14.1)
Requirement already satisfied: scipy in c:\users\student\anaconda3\lib\site-p
ackages (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\student\anaconda3\li
b\site-packages (from scipy) (1.18.1)
```

In [3]:
```
import random
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import sklearn_extra
import scipy
from sklearn import datasets
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import sklearn.cluster
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import MiniBatchKMeans
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn_extra.cluster import KMedoids
from scipy.cluster.hierarchy import dendrogram
```

In [5]:
```
categories = ['soc.religion.christian', 'sci.space', 'rec.sport.hockey', 'com
p.sys.mac.hardware', 'sci.med']
# categories = ['alt.atheism', 'soc.religion.christian']
# categories = ['comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware']
# categories = ['rec.sport.baseball', 'rec.sport.hockey']
# 'alt.atheism','comp.graphics','comp.os.ms-windows.misc','comp.sys.ibm.pc.har
dware',
# 'comp.sys.mac.hardware','comp.windows.x', 'misc.forsale', 'rec.autos',
# 'rec.motorcycles',  'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt',
# 'sci.electronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.p
olitics.guns',
# 'talk.politics.mideast', 'talk.politics.misc', 'talk.religion.misc'
train = sklearn.datasets.fetch_20newsgroups(subset='train', categories=categor
ies, remove=('headers', 'footers', 'quotes'),)
test = sklearn.datasets.fetch_20newsgroups(subset='test', categories=categorie
s, remove=('headers', 'footers', 'quotes'),)
print('train data size:', len(train.data))
print('test data size:', len(test.data))
```

```
train data size: 2964
test data size: 1972
```

Nearest Neighbors

In [6]:
```
import pandas as pd
pd.set_option('display.max_colwidth', -1)

idx = 200
inst = train.data[idx]
train.target_names[train.target[idx]]
pd.DataFrame.from_dict({'category':[train.target_names[train.target[idx]]], 'e
mail':[inst]})
```

```
C:\Users\student\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureW
arning: Passing a negative integer is deprecated in version 1.0 and will not
be supported in future version. Instead, use None to not limit the column wid
th.
```

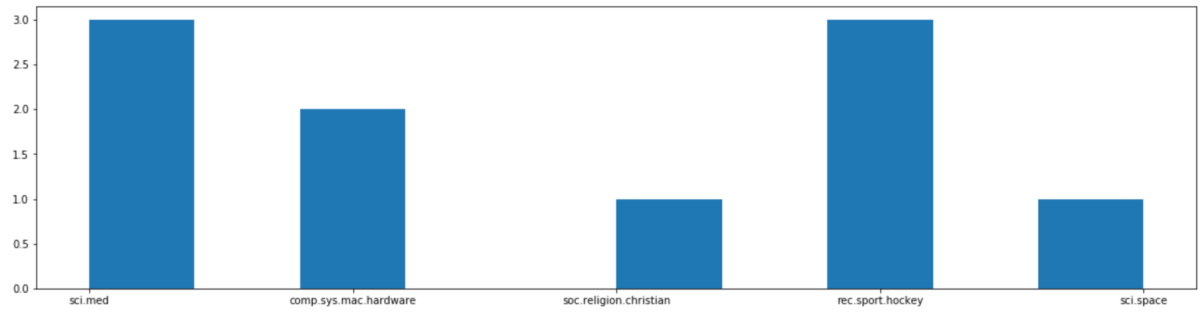
Out[6]:

	category	
0	comp.sys.mac.hardware	I believe it go:\n680060\npowerPC\nPentium\n680040\n486\n680030\n386\n680020\n286=68 a resent article in one of the macMags I think a 50mHz 030 accelerator was\n slig than a 25mHz 040 accel. But, this is using a system designed\n for the 030. So, reason that a system designed for an 040 ie\n quadra) would do better. So over 040 = 030 * 2.5 or so.\n Along the same lines the new POwerPC stuff is suppose system\n at the level of a fast quadra, but system 8 or whatever will allow 3 times t of a 040 in the powerPC based systems. and wait for the 680060. I think\ pentium.\n\npro-life pro-

In [7]:
```
num_neighs = 10

metric = 'l1' # or 'cosine', 'l1', 'l2'
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True)
train.vecs = features.fit_transform(train.data)
nbrs = NearestNeighbors(n_neighbors=num_neighs+1, algorithm='brute', metric=me
tric).fit(train.vecs)
distances, indices = nbrs.kneighbors(train.vecs[idx])
# for nidx in indices[indices!=idx]:
#   print('-----')
#   print(train.target_names[train.target[nidx]])
#   print(train.data[nidx])
plt.rcParams["figure.figsize"] = (20,5)
plt.hist([train.target_names[nidx] for nidx in train.target[indices[indices!=i
dx]]])
pd.DataFrame.from_dict({'category':[train.target_names[nidx] for nidx in train
.target[indices[indices!=idx]]], 'email':[train.data[nidx] for nidx in indices
[indices!=idx]]})
```

Out[7]:

	category	email
0	sci.med	
1	sci.med	
2	sci.med	
3	comp.sys.mac.hardware	\n
4	comp.sys.mac.hardware	each
5	soc.religion.christian	\n\n\n\n\n\n
6	rec.sport.hockey	
7	rec.sport.hockey	
8	sci.space	
9	rec.sport.hockey	



In [8]:
```
np.shape(distances)
np.mean(distances)
print(indices)
print(distances)
print(sklearn.metrics.pairwise.cosine_similarity(train.vecs[200], train.vecs[9
2]))
```

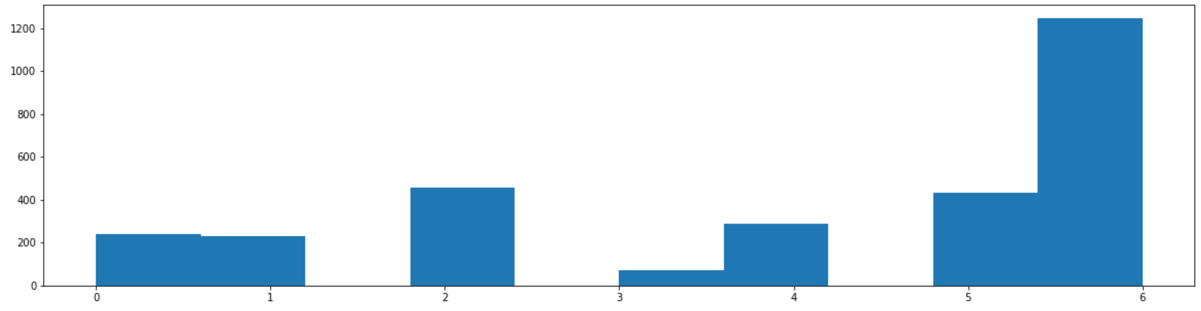
```
[[ 200 1326 1330   93  602 2095 1282 2900 2332 1340 2184]]
[[0.         9.80042532 9.80042532 9.80042532 9.80042532 9.80042532
  9.80042532 9.80042532 9.80042532 9.80042532 9.80042532]]
[[0.]]
```

KMeans

In [9]:
```
#random.seed(a = 200)
features = TfidfVectorizer(ngram_range=(1,1), stop_words= 'english', lowercase
=True, max_features=2000, max_df=0.9)
train.vecs = features.fit_transform(train.data)
```

In [10]:
```
clusterer = KMeans(n_clusters=7, init='k-means++', max_iter=500, n_init=5, alg
orithm="full")
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

Out[10]:
```
(array([ 239.,  231.,    0.,  455.,    0.,   71.,  288.,    0.,  433.,
        1247.]),
 array([0. , 0.6, 1.2, 1.8, 2.4, 3. , 3.6, 4.2, 4.8, 5.4, 6. ]),
 <a list of 10 Patch objects>)
```



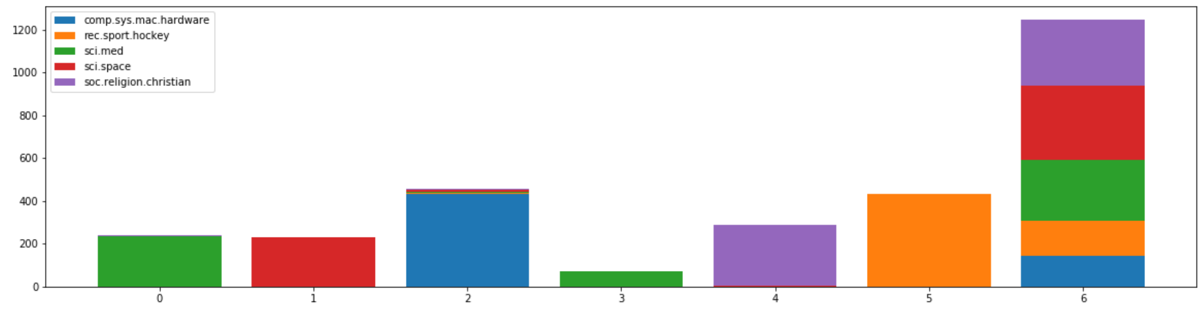
In [11]:
```
clusts[1:20]
```

Out[11]:
```
array([6, 3, 6, 4, 3, 5, 1, 6, 6, 6, 5, 6, 2, 3, 6, 6, 2, 6, 0])
```

In [12]:
```
def purity_score(c, y):
  '''From: http://www.caner.io/purity-in-python.html'''
  A = np.c_[(c,y)]
  n_accurate = 0.
  for j in np.unique(A[:,0]):
    z = A[A[:,0] == j, 1]
    x = np.argmax(np.bincount(z))
    n_accurate += len(z[z == x])
  return n_accurate / A.shape[0]

def cluster_purity(c, y):
  numy = len(set(y))
  cvals = list(set(c)) #[str(ce) for ce in list(set(c))]
  numc = len(cvals)
  ind = [str(cval) for cval in cvals] #np.arange(numc)
  bottom = np.zeros(numc)
  for yidx in range(numy):
    counts = np.zeros(numc)
    for cidx in range(numc):
      num = len(list(filter(lambda p: p[0]==cvals[cidx] and p[1]==yidx, zip(c,
y))))
      counts[cidx] = num
    plt.bar(ind, counts,label=train.target_names[yidx],bottom=bottom)
    bottom = bottom + counts
  plt.legend()
```

In [13]:
```
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

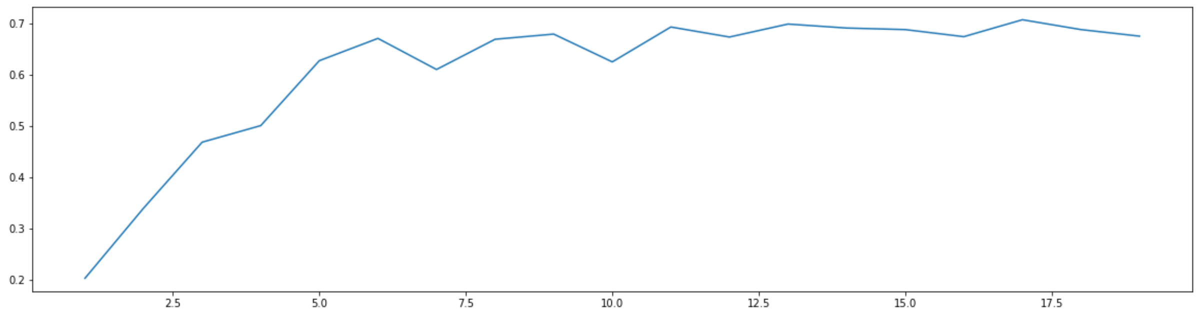
```
Purity: 0.6852226720647774
```



In [14]:
```
import nltk

purity = []
wcss=[]

krange = range(1,20)
for k in krange:
  tclusterer = KMeans(n_clusters=k, init='k-means++', max_iter=500, n_init=5,
algorithm="full")
  clusts = tclusterer.fit_predict(train.vecs)
  purity.append(purity_score(clusts, train.target))
  wcss.append(tclusterer.inertia_)
  print('k=',k,'done, purity:', purity[k-1])

#plt.plot(krange, wcss)
plt.plot(krange, purity)
```

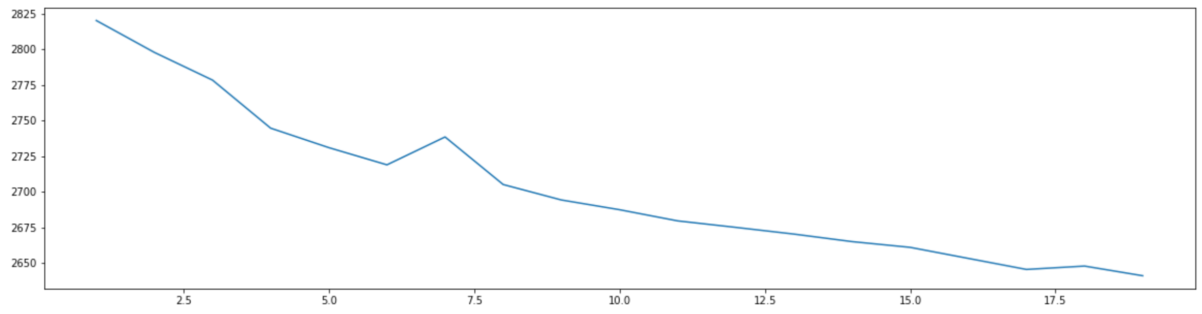
```
k= 1 done, purity: 0.20242914979757085
k= 2 done, purity: 0.33940620782726044
k= 3 done, purity: 0.4682860998650472
k= 4 done, purity: 0.5006747638326585
k= 5 done, purity: 0.6275303643724697
k= 6 done, purity: 0.6710526315789473
k= 7 done, purity: 0.6103238866396761
k= 8 done, purity: 0.6693657219973009
k= 9 done, purity: 0.6794871794871795
k= 10 done, purity: 0.6251686909581646
k= 11 done, purity: 0.6933198380566802
k= 12 done, purity: 0.6737516869095816
k= 13 done, purity: 0.699055330634278
k= 14 done, purity: 0.6912955465587044
k= 15 done, purity: 0.6882591093117408
k= 16 done, purity: 0.6744264507422402
k= 17 done, purity: 0.7074898785425101
k= 18 done, purity: 0.6882591093117408
k= 19 done, purity: 0.6754385964912281
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x2af470ce288>]
```



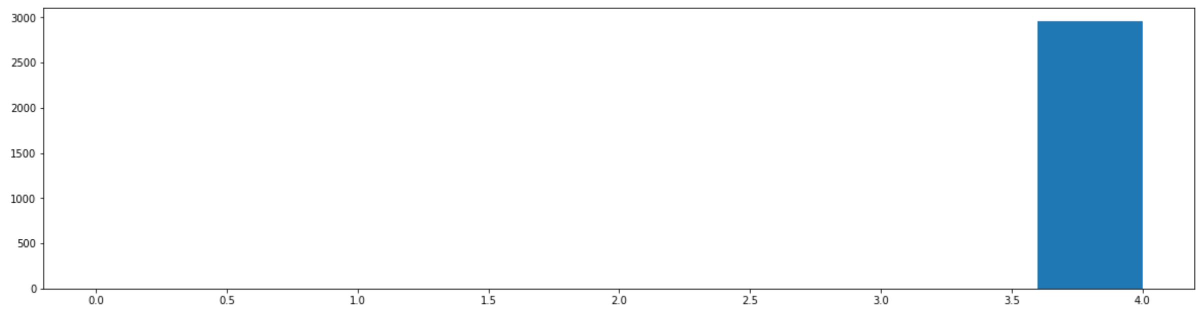
In [15]:
```
plt.plot(krange, wcss)
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x2af449da488>]
```



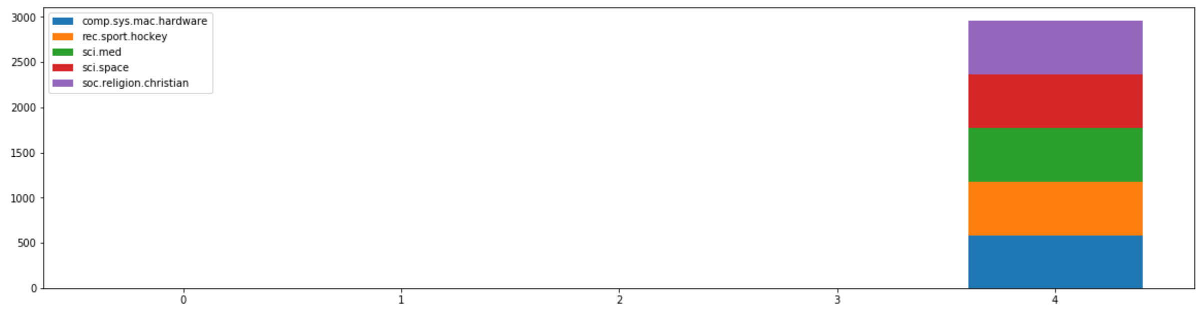
In [16]:
```
random.seed(a = 200)
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True)
train.vecs = features.fit_transform(train.data)

clusterer = KMeans(n_clusters=5, init='k-means++', max_iter=100, n_init=1)
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

Out[16]:
```
(array([1.00e+00, 0.00e+00, 1.00e+00, 0.00e+00, 0.00e+00, 1.00e+00,
        0.00e+00, 1.00e+00, 0.00e+00, 2.96e+03]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```



In [17]:
```
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

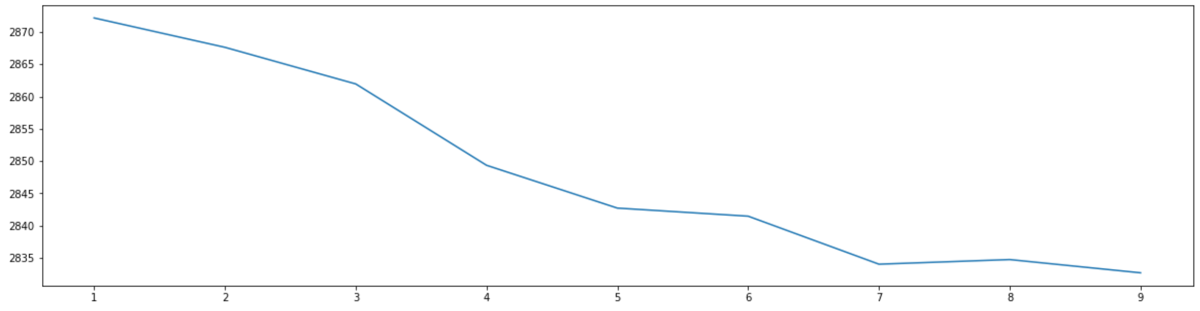
```
Purity: 0.2034412955465587
```



In [18]:
```
inertias = []
krange = range(1,10)
for k in krange:
  tclusterer = KMeans(n_clusters=k, init='k-means++', max_iter=100, n_init=1)
  tclusterer.fit_predict(train.vecs)
  inertias.append(tclusterer.inertia_)
  print('k=',k,'done, inertia:', tclusterer.inertia_)
plt.plot(krange, inertias)
```

```
k= 1 done, inertia: 2872.1832799486715
k= 2 done, inertia: 2867.6324779555475
k= 3 done, inertia: 2861.954417351781
k= 4 done, inertia: 2849.341667642632
k= 5 done, inertia: 2842.7048318195166
k= 6 done, inertia: 2841.448827154473
k= 7 done, inertia: 2834.0087124240786
k= 8 done, inertia: 2834.721975967095
k= 9 done, inertia: 2832.671181485172
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x2af44555f88>]
```



In [19]:
```
centroids = clusterer.cluster_centers_
```

In [20]:
```
train.vecs[0].toarray()
```

Out[20]:
```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

In [21]:
```
print(centroids)
```

```
[[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.21542679e-03 2.41134703e-05 1.39883766e-05 ... 1.01706475e-04
  4.01890626e-05 4.01890626e-05]]
```

In [22]:
```
nbrs = NearestNeighbors(n_neighbors=num_neighs+1, algorithm='brute', metric=me
tric).fit(train.vecs)
distance, indices = nbrs.kneighbors(centroids)
```

```
In [23]: indices
```

```
Out[23]: array([[1966,  151, 2567,  260, 1576,  637, 2118, 1883, 1554,  446, 1543],
               [1549, 2414,  759,  780,  776, 2264,   92, 2652,  446,  406, 2163],
               [1084,  881, 2484, 1576,  341,  682, 2521, 1543, 2524,   92, 1554],
               [1892,  602, 2163, 1765, 2877, 2303,  151, 1478, 2567, 2180, 1465],
               [ 759, 1465,  780,  776,  151, 2877, 2303, 1379,  682, 2652, 2264]],
              dtype=int64)
```

In [24]:
```
print('Document closest to Cluster 1:\n', train.data[1126])
```

```
Document closest to Cluster 1:
 I have come across what I consider to be an excellent tract. It is a
bit lengthy for a posting, but I thought I'd share it with all of you
anyway. Feel free to pass it along to anyone whom you feel might
benefit from what it says. May God richly bless those who read it.

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

                       D O E S   G O D   L O V E   Y O U ?


Q. What  kind  of  question  is that?   Anyone who can read sees signs,
   tracts, books, and bumper stickers that say, "God Loves You."  Isn't
   that true?

A. It  is  true that God offers His love to the whole world, as we read
   in one of the most quoted verses in the Bible:

      For  God  so  loved  the world, that he gave his only begotten
      Son, that whosoever believeth in him should  not  perish,  but
      have everlasting life.                                 John 3:16


      The way of the wicked is an abomination unto the LORD:  but he
      loveth him that followeth after righteousness.   Proverbs 15:9

      For  the LORD knoweth the way of the righteous: but the way of
      the ungodly shall perish.                              Psalm 1:6


Q. But  I am not wicked.  I am a decent, moral person.  Surely the good
   I have done in my life far outweighs whatever bad I have done.   How
   can these verses apply to me?

A. By  God's  standard  of  righteousness even the most moral person is
   looked upon by God as a desperate sinner on his way to  Hell.    The
   Bible teaches that no one is good enough in himself to go to Heaven.
   On  the  contrary,  we  are all sinners and we are all guilty before
   God.

      As it is written, There is none righteous, no, not one:  There
      is  none  that understandeth, there is none that seeketh after
      God.                                              Romans 3:10-11

      The  heart  is  deceitful  above  all  things, and desperately
      wicked: who can know it?                         Jeremiah 17:9


Q. If I am such a wicked person in God's sight, what will God do to me?

A. The  Bible  teaches that at the end of the world all the wicked will
   come under eternal punishment in a place called Hell.

      For  a  fire is kindled in mine anger, and shall burn unto the
      lowest hell, and shall consume the earth  with  her  increase,
      and set on fire the foundations of the mountains.  I will heap
      mischiefs upon them; I will spend mine arrows upon them.  They
```

> shall be burnt with hunger, and devoured with burning heat, and with bitter destruction: I will also send the teeth of beasts upon them, with the poison of serpents of the dust.
> Deuteronomy 32:22-24

Q. Oh, come on now! Hell is not real, is it? Surely things are not that bad.

A. Indeed, Hell is very real, and things are that bad for the individual who does not know the Lord Jesus Christ as Savior. The Bible makes many references to Hell, indicating that it is both eternal and consists of perpetual suffering.

> And whosoever was not found written in the book of life was cast into the lake of fire. Revelation 20:15

> So shall it be at the end of the world: the angels shall come forth, and sever the wicked from among the just, And shall cast them into the furnace of fire: there shall be wailing and gnashing of teeth. Matthew 13:49-50

> ... when the Lord Jesus shall be revealed from heaven with his mighty angels, In flaming fire taking vengeance on them that know not God, and that obey not the gospel of our Lord Jesus Christ: Who shall be punished with everlasting destruction from the presence of the Lord, and from the glory of his power; 2 Thessalonians 1:7-9

Q. That is terrible! Why would God create a Hell?

A. Hell is terrible, and it exists because God created man to be accountable to God for his actions. God's perfect justice demands payment for sin.

> For the wages of sin is death; Romans 6:23

> For we must all appear before the judgment seat of Christ; that every one may receive the things done in his body, according to that he hath done, whether it be good or bad.
> 2 Corinthians 5:10

> But I say unto you, That every idle word that men shall speak, they shall give account thereof in the day of judgment.
> Matthew 12:36

Q. Does that mean that at the end of the world everyone will be brought to life again to be judged and then to be sent to Hell?

A. Indeed it does; that is, unless we can find someone to be our substitute in bearing the punishment of eternal damnation for our sins. That someone is God Himself, who came to earth as Jesus Christ to bear the wrath of God for all who believe in Him.

> All we like sheep have gone astray; we have turned every one

to his own way; and the LORD hath laid on him the iniquity of us all. Isaiah 53:6

But he was wounded for our transgressions, he was bruised for our iniquities: the chastisement of our peace was upon him; and with his stripes we are healed. Isaiah 53:5

For I delivered unto you first of all that which I also received, how that Christ died for our sins according to the scriptures; And that he was buried, and that he rose again the third day according to the scriptures: 1 Corinthians 15:3-4

For he hath made him to be sin for us, who knew no sin; that we might be made the righteousness of God in him.
2 Corinthians 5:21

Q. Are you saying that if I trust in Christ as my substitute, Who was already punished for my sins, then I will not have to worry about Hell anymore?

A. Yes, this is so! If I have believed in Christ as my Savior, then it is as if I have already stood before the Judgment Throne of God. Christ as my substitute has already paid for my sins.

He that believeth on the Son hath everlasting life: and he that believeth not the Son shall not see life; but the wrath of God abideth on him. John 3:36

Q. But what does it mean to believe on Him? If I agree with all that the Bible says about Christ as Savior, then am I saved from going to Hell?

A. Believing on Christ means a whole lot more than agreeing in our minds with the truths of the Bible. It means that we hang our whole lives on Him. It means that we entrust every part of our lives to the truths of the Bible. It means that we turn away from our sins and serve Christ as our Lord.

No man can serve two masters: for either he will hate the one, and love the other; or else he will hold to the one, and despise the other. Ye cannot serve God and mammon.
Matthew 6:24

Repent ye therefore, and be converted, that your sins may be blotted out, when the times of refreshing shall come from the presence of the Lord; Acts 3:19

Q. Are you saying that there is no other way to escape Hell except through Jesus? What about all the other religions? Will their followers also go to Hell?

A. Yes, indeed. They cannot escape the fact that God holds us account-able for our sins. God demands that we pay for our sins. Other religions cannot provide a substitute to bear the sins of their followers. Christ is the only one who is able to bear our guilt and save us.

> Neither is there salvation in any other: for there is none other name under heaven given among men, whereby we must be saved. Acts 4:12

> I am the way, the truth, and the life: no man cometh unto the Father, but by me. John 14:6

> If we confess our sins, he is faithful and just to forgive us our sins, and to cleanse us from all unrighteousness.
> 1 John 1:9

Q. Now I am desperate. I do not want to go to Hell. What can I do?

A. You must remember that God is the only one who can help you. You must throw yourself altogether on the mercies of God. As you see your hopeless condition as a sinner, cry out to God to save you.

> And the publican, standing afar off, would not lift up so much as his eyes unto heaven, but smote upon his breast, saying, God be merciful to me a sinner. Luke 18:13

> ... Sirs, what must I do to be saved? And they said, Believe on the Lord Jesus Christ, and thou shalt be saved, ...
> Acts 16:30-31

Q. But how can I believe on Christ if I know so little about Him?

A. Wonderfully, God not only saves us through the Lord Jesus, but He also gives us the faith to believe on Him. You can pray to God that He will give you faith in Jesus Christ as your Savior.

> For by grace are ye saved through faith; and that not of yourselves: it is the gift of God: Ephesians 2:8

God works particularly through the Bible to give us that faith. So, if you really mean business with God about your salvation, you should use every opportunity to hear and study the Bible, which is the only Word of God.
In this brochure, all verses from the Bible are within indented paragraphs. Give heed to them with all your heart.

> So then faith cometh by hearing, and hearing by the word of God. Romans 10:17

Q. But does this mean that I have to surrender everything to God?

A. Yes. God wants us to come to Him in total humility, acknowledging our sinfulness and our helplessness, trusting totally in Him.

> The sacrifices of God are a broken spirit: a broken and a

contrite heart, O God, thou wilt not despise. Psalm 51:17

Because we are sinners we love our sins. Therefore, we must begin to pray to God for an intense hatred of our sins. And if we sincerely desire salvation, we will also begin to turn from our sins as God strengthens us. We know that our sins are sending us to Hell.

> Unto you first God, having raised up his Son Jesus, sent him to bless you, in turning away every one of you from his iniquities. Acts 3:26

Q. Doesn't the Bible teach that I must attend church regularly and be baptized? Will these save me?

A. If possible, we should do these things, but they will not save us. No work of any kind can secure our salvation. Salvation is God's sovereign gift of grace given according to His mercy and good pleasure. Salvation is

> Not of works, lest any man should boast. Ephesians 2:9

Q. What else will happen at the end of the world?

A. Those who have trusted in Jesus as their Savior will be transformed into their glorious eternal bodies and will be with Christ forevermore.

> For the Lord himself shall descend from heaven with a shout, with the voice of the archangel, and with the trump of God: and the dead in Christ shall rise first: Then we which are alive and remain shall be caught up together with them in the clouds, to meet the Lord in the air: and so shall we ever be with the Lord. 1 Thessalonians 4:16-17

Q. What will happen to the earth at that time?

A. God will destroy the entire universe by fire and create new heavens and a new earth where Christ will reign with His believers forevermore.

> But the day of the Lord will come as a thief in the night; in the which the heavens shall pass away with a great noise, and the elements shall melt with fervent heat, the earth also and the works that are therein shall be burned up. ... Nevertheless we, according to his promise, look for new heavens and a new earth, wherein dwelleth righteousness.
> 2 Peter 3:10,13

Q. Does the Bible give us any idea of when the end of the earth will come?

A. Yes! The end will come when Christ has saved all whom He plans to save.

> And this gospel of the kingdom shall be preached in all the world for a witness unto all nations; and then shall the end come. Matthew 24:14

Q. Can we know how close to the end of the world we might be?

A. Yes! God gives much information in the Bible concerning the timing of the history of the world and tells us that while the Day of the Lord will come as a thief in the night for the unsaved, it will not come as a thief for the believers. There is much evidence in the Bible that the end of the world and the return of Christ may be very, very close.* All the time clues in the Bible point to this.

> For when they shall say, Peace and safety; then sudden destruction cometh upon them, as travail upon a woman with child; and they shall not escape. 1 Thessalonians 5:3

> Surely the Lord GOD will do nothing, but he revealeth his secret unto his servants the prophets. Amos 3:7

Q. But that means Judgment Day is almost here.

A. Yes, it does. God warned ancient Nineveh that He was going to destroy that great city and He gave them forty days warning.

> And Jonah began to enter into the city a day's journey, and he cried, and said, Yet forty days, and Nineveh shall be overthrown. Jonah 3:4

Q. What did the people of Nineveh do?

A. From the king on down they humbled themselves before God, repented of their sins, and cried to God for mercy.

> But let man and beast be covered with sackcloth, and cry mightily unto God: yea, let them turn every one from his evil way, and from the violence that is in their hands. Who can tell if God will turn and repent, and turn away from his fierce anger, that we perish not? Jonah 3:8-9

Q. Did God hear their prayers?

A. Yes. God saved a great many people of Nineveh.

Q. Can I still cry to God for mercy so that I will not come into judgment?

A. Yes. There is still time to become saved even though that time has become very short.

> How shall we escape, if we neglect so great salvation; which at the first began to be spoken by the Lord, and was confirmed unto us by them that heard him; Hebrews 2:3

> In God is my salvation and my glory: the rock of my strength, and my refuge, is in God. Trust in him at all times; ye people, pour out your heart before him: God is a refuge for us. Psalm 62:7-8

A R E Y O U R E A D Y T O M E E T G O D ?

A book entitled 1994?, written by Harold Camping, presents Biblical information that we may be very near the end of time. For information on how to obtain a copy or to receive a free program guide and list of radio stations on which you can hear our Gospel programs, please write to Family Radio, Oakland, California, 94621 (The United States of America), or call 1-800-543-1495.

The foregoing is a copy of the "Does God Love You?" tract printed by, and available free of charge from, Family Radio. A number of minor changes have been made to its layout to facilitate computer printing and distribution. The only change to the text itself is the paragraph which describes the way in which Biblical passages appear within the text. In the original tract they appear in italic lettering; they appear here as indented paragraphs.

I have read Mr. Camping's book, compared it with what the Bible actually says, find it to be the most credible research with respect to what the future holds that I have ever come across, and agree with him that there is just too much data to ignore. While none of us is guaranteed one more second of life, and while we, therefore, should take these matters very seriously regardless of when Christ will actually return, it would appear that our natural tendency to postpone caring about our eternal destiny until we feel that our death is imminent is even more senseless now because, in all likelihood, the law of averages with respect to life expectancy no longer applies. If you wish to obtain a copy of this book so that you can check out these facts for yourself, you may find the following information helpful:

title:	1994?
author:	Harold Camping
publisher:	Vantage Press
distributor:	Baker and Taylor
ISBN:	0-533-10368-1

I have chosen to share this tract with you because I whole-heartedly agree with everything it declares and feel that now, perhaps more than

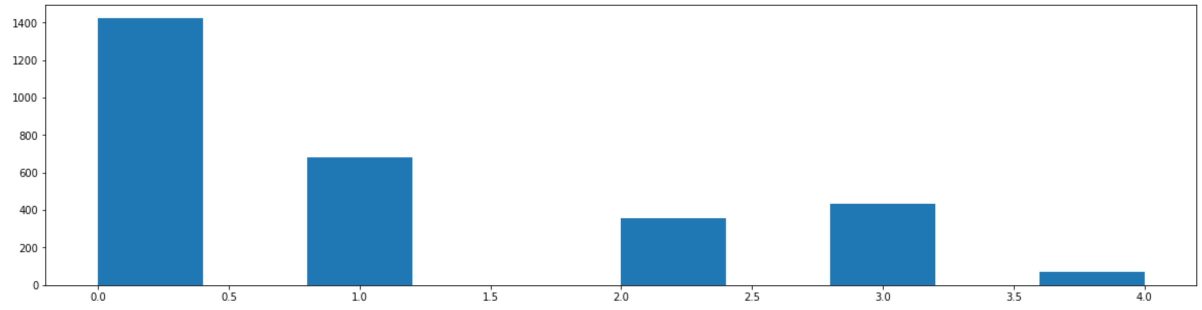
```
ever  before,  this information must be made known.  To paraphrase Acts
20:27,  it  does not shun to declare unto us all the counsel of God.  I
am  always  willing  to  discuss  the  eternal truths of the Bible with
anyone who is interested as I believe them to be the only issues of any
real  importance since we will spend, comparatively speaking, so little
time  on this side of the grave and so much on the other.  Feel free to
get in touch with me at any time:

      e-mail: davem@bnr.ca
      office: 1-613-765-4671
      home:   1-613-726-0014
```

In [25]:

```
#random.seed(a = 200)
#Adding new features: max_features=2000 and max_df=0.9
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True, max_features=2000, max_df=0.9)
train.vecs = features.fit_transform(train.data)

clusterer = KMeans(n_clusters=5, init='k-means++', max_iter=100, n_init=1)
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

Out[25]:

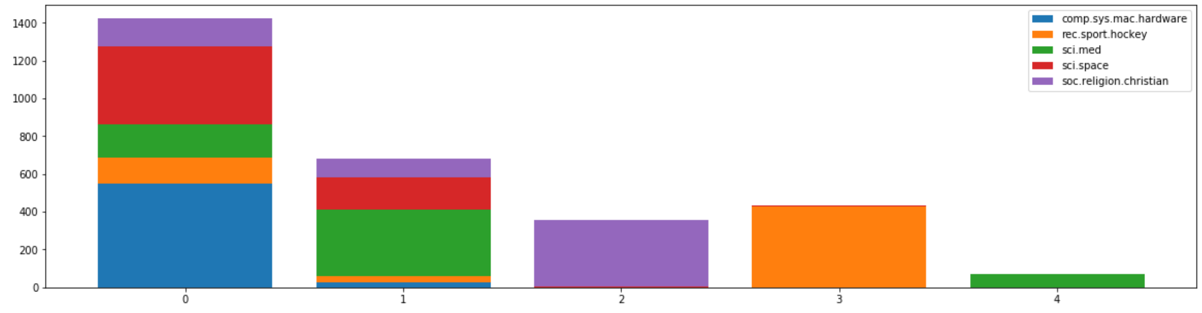
```
(array([1424.,    0.,  683.,    0.,    0.,  355.,    0.,  431.,    0.,
          71.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```



In [26]:

```
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

```
Purity: 0.5914304993252362
```



In [28]:

```
# interpretation:
After adding two features: max_features=2000 and max_df=0.9, the purity score
increased from 0.203 to 0.591
```

DBScan

In [29]:
```
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True)
train.vecs = features.fit_transform(train.data)
eps_range = np.arange(0.86,0.88,0.001)
max_purity = 0
for i in range(len(eps_range)):
    for j in range(10, 100, 10):
        clusterer = sklearn.cluster.DBSCAN(eps=eps_range[i], min_samples=j, me
tric='cosine')
        clusts = clusterer.fit_predict(train.vecs)
        score = purity_score(clusts, train.target)
        print('Purity:', eps_range[i], ",", j, "::",  score)
        if( score > max_purity):
            opt_eps = eps_range[i]
            opt_min_sample = j
            max_purity = score

#opt_min_sample = 10
#opt_eps = 0.87

clusterer = sklearn.cluster.DBSCAN(eps=opt_eps, min_samples=opt_min_sample, me
tric='cosine')
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

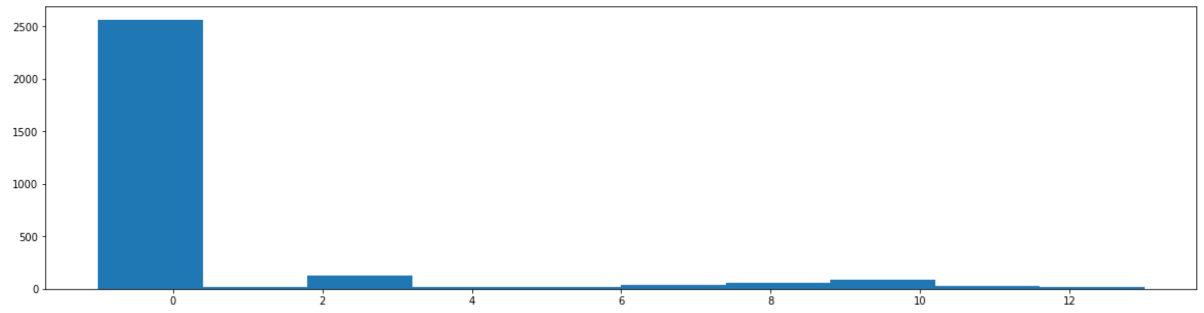
```
Purity: 0.86 , 10 :: 0.28609986504723345
Purity: 0.86 , 20 :: 0.24358974358974358
Purity: 0.86 , 30 :: 0.22840755735492577
Purity: 0.86 , 40 :: 0.22807017543859648
Purity: 0.86 , 50 :: 0.22739541160593793
Purity: 0.86 , 60 :: 0.22705802968960864
Purity: 0.86 , 70 :: 0.22705802968960864
Purity: 0.86 , 80 :: 0.20242914979757085
Purity: 0.86 , 90 :: 0.20242914979757085
Purity: 0.861 , 10 :: 0.2881241565452092
Purity: 0.861 , 20 :: 0.24493927125506074
Purity: 0.861 , 30 :: 0.22874493927125505
Purity: 0.861 , 40 :: 0.22840755735492577
Purity: 0.861 , 50 :: 0.22807017543859648
Purity: 0.861 , 60 :: 0.22739541160593793
Purity: 0.861 , 70 :: 0.22739541160593793
Purity: 0.861 , 80 :: 0.20242914979757085
Purity: 0.861 , 90 :: 0.20242914979757085
Purity: 0.862 , 10 :: 0.29048582995951416
Purity: 0.862 , 20 :: 0.24527665317139002
Purity: 0.862 , 30 :: 0.22874493927125505
Purity: 0.862 , 40 :: 0.22840755735492577
Purity: 0.862 , 50 :: 0.22840755735492577
Purity: 0.862 , 60 :: 0.22739541160593793
Purity: 0.862 , 70 :: 0.22739541160593793
Purity: 0.862 , 80 :: 0.20242914979757085
Purity: 0.862 , 90 :: 0.20242914979757085
Purity: 0.863 , 10 :: 0.29116059379217274
Purity: 0.863 , 20 :: 0.25033738191632926
Purity: 0.863 , 30 :: 0.22874493927125505
Purity: 0.863 , 40 :: 0.22840755735492577
Purity: 0.863 , 50 :: 0.22840755735492577
Purity: 0.863 , 60 :: 0.22739541160593793
Purity: 0.863 , 70 :: 0.22739541160593793
Purity: 0.863 , 80 :: 0.20242914979757085
Purity: 0.863 , 90 :: 0.20242914979757085
Purity: 0.864 , 10 :: 0.29284750337381915
Purity: 0.864 , 20 :: 0.25134952766531715
Purity: 0.864 , 30 :: 0.22908232118758434
Purity: 0.864 , 40 :: 0.22874493927125505
Purity: 0.864 , 50 :: 0.22874493927125505
Purity: 0.864 , 60 :: 0.22773279352226722
Purity: 0.864 , 70 :: 0.22773279352226722
Purity: 0.864 , 80 :: 0.20242914979757085
Purity: 0.864 , 90 :: 0.20242914979757085
Purity: 0.865 , 10 :: 0.2975708502024291
Purity: 0.865 , 20 :: 0.252361673414305
Purity: 0.865 , 30 :: 0.22908232118758434
Purity: 0.865 , 40 :: 0.22874493927125505
Purity: 0.865 , 50 :: 0.22874493927125505
Purity: 0.865 , 60 :: 0.22773279352226722
Purity: 0.865 , 70 :: 0.22773279352226722
Purity: 0.865 , 80 :: 0.20242914979757085
Purity: 0.865 , 90 :: 0.20242914979757085
Purity: 0.866 , 10 :: 0.2982456140350877
Purity: 0.866 , 20 :: 0.2526990553306343
Purity: 0.866 , 30 :: 0.22908232118758434
```

```
Purity: 0.866 , 40 :: 0.22874493927125505
Purity: 0.866 , 50 :: 0.22874493927125505
Purity: 0.866 , 60 :: 0.22773279352226722
Purity: 0.866 , 70 :: 0.22773279352226722
Purity: 0.866 , 80 :: 0.20242914979757085
Purity: 0.866 , 90 :: 0.20242914979757085
Purity: 0.867 , 10 :: 0.29959514170040485
Purity: 0.867 , 20 :: 0.25944669365721995
Purity: 0.867 , 30 :: 0.22908232118758434
Purity: 0.867 , 40 :: 0.22874493927125505
Purity: 0.867 , 50 :: 0.22874493927125505
Purity: 0.867 , 60 :: 0.22773279352226722
Purity: 0.867 , 70 :: 0.22773279352226722
Purity: 0.867 , 80 :: 0.20242914979757085
Purity: 0.867 , 90 :: 0.20242914979757085
Purity: 0.868 , 10 :: 0.3006072874493927
Purity: 0.868 , 20 :: 0.2658569500674764
Purity: 0.868 , 30 :: 0.22908232118758434
Purity: 0.868 , 40 :: 0.22908232118758434
Purity: 0.868 , 50 :: 0.22874493927125505
Purity: 0.868 , 60 :: 0.22773279352226722
Purity: 0.868 , 70 :: 0.22773279352226722
Purity: 0.868 , 80 :: 0.20242914979757085
Purity: 0.868 , 90 :: 0.20242914979757085
Purity: 0.869 , 10 :: 0.30195681511470984
Purity: 0.869 , 20 :: 0.26889338731443996
Purity: 0.869 , 30 :: 0.22908232118758434
Purity: 0.869 , 40 :: 0.22908232118758434
Purity: 0.869 , 50 :: 0.22874493927125505
Purity: 0.869 , 60 :: 0.22773279352226722
Purity: 0.869 , 70 :: 0.22773279352226722
Purity: 0.869 , 80 :: 0.20242914979757085
Purity: 0.869 , 90 :: 0.20242914979757085
Purity: 0.87 , 10 :: 0.305668016194332
Purity: 0.87 , 20 :: 0.2699055330634278
Purity: 0.87 , 30 :: 0.22908232118758434
Purity: 0.87 , 40 :: 0.22908232118758434
Purity: 0.87 , 50 :: 0.22874493927125505
Purity: 0.87 , 60 :: 0.22807017543859648
Purity: 0.87 , 70 :: 0.22773279352226722
Purity: 0.87 , 80 :: 0.20242914979757085
Purity: 0.87 , 90 :: 0.20242914979757085
Purity: 0.871 , 10 :: 0.30701754385964913
Purity: 0.871 , 20 :: 0.2699055330634278
Purity: 0.871 , 30 :: 0.22908232118758434
Purity: 0.871 , 40 :: 0.22908232118758434
Purity: 0.871 , 50 :: 0.22874493927125505
Purity: 0.871 , 60 :: 0.22807017543859648
Purity: 0.871 , 70 :: 0.22773279352226722
Purity: 0.871 , 80 :: 0.20242914979757085
Purity: 0.871 , 90 :: 0.20242914979757085
Purity: 0.872 , 10 :: 0.3076923076923077
Purity: 0.872 , 20 :: 0.27058029689608637
Purity: 0.872 , 30 :: 0.24595141700404857
Purity: 0.872 , 40 :: 0.22908232118758434
Purity: 0.872 , 50 :: 0.22874493927125505
Purity: 0.872 , 60 :: 0.22807017543859648
```

```
Purity: 0.872 , 70 :: 0.22773279352226722
Purity: 0.872 , 80 :: 0.20242914979757085
Purity: 0.872 , 90 :: 0.20242914979757085
Purity: 0.873 , 10 :: 0.3090418353576248
Purity: 0.873 , 20 :: 0.2715924426450742
Purity: 0.873 , 30 :: 0.24662618083670715
Purity: 0.873 , 40 :: 0.22941970310391363
Purity: 0.873 , 50 :: 0.22908232118758434
Purity: 0.873 , 60 :: 0.22807017543859648
Purity: 0.873 , 70 :: 0.22773279352226722
Purity: 0.873 , 80 :: 0.20242914979757085
Purity: 0.873 , 90 :: 0.20242914979757085
Purity: 0.874 , 10 :: 0.3164642375168691
Purity: 0.874 , 20 :: 0.2722672064777328
Purity: 0.874 , 30 :: 0.2543859649122807
Purity: 0.874 , 40 :: 0.22941970310391363
Purity: 0.874 , 50 :: 0.22908232118758434
Purity: 0.874 , 60 :: 0.22807017543859648
Purity: 0.874 , 70 :: 0.22773279352226722
Purity: 0.874 , 80 :: 0.20242914979757085
Purity: 0.874 , 90 :: 0.20242914979757085
Purity: 0.875 , 10 :: 0.31950067476383265
Purity: 0.875 , 20 :: 0.2726045883940621
Purity: 0.875 , 30 :: 0.2543859649122807
Purity: 0.875 , 40 :: 0.22941970310391363
Purity: 0.875 , 50 :: 0.22908232118758434
Purity: 0.875 , 60 :: 0.22807017543859648
Purity: 0.875 , 70 :: 0.22773279352226722
Purity: 0.875 , 80 :: 0.20242914979757085
Purity: 0.875 , 90 :: 0.20242914979757085
Purity: 0.876 , 10 :: 0.3225371120107962
Purity: 0.876 , 20 :: 0.2732793522267207
Purity: 0.876 , 30 :: 0.25472334682861
Purity: 0.876 , 40 :: 0.22941970310391363
Purity: 0.876 , 50 :: 0.22908232118758434
Purity: 0.876 , 60 :: 0.22807017543859648
Purity: 0.876 , 70 :: 0.22773279352226722
Purity: 0.876 , 80 :: 0.20242914979757085
Purity: 0.876 , 90 :: 0.20242914979757085
Purity: 0.877 , 10 :: 0.3252361673414305
Purity: 0.877 , 20 :: 0.27361673414304993
Purity: 0.877 , 30 :: 0.2550607287449393
Purity: 0.877 , 40 :: 0.22941970310391363
Purity: 0.877 , 50 :: 0.22908232118758434
Purity: 0.877 , 60 :: 0.22807017543859648
Purity: 0.877 , 70 :: 0.22773279352226722
Purity: 0.877 , 80 :: 0.20242914979757085
Purity: 0.877 , 90 :: 0.20242914979757085
Purity: 0.878 , 10 :: 0.32624831309041835
Purity: 0.878 , 20 :: 0.27968960863697706
Purity: 0.878 , 30 :: 0.2564102564102564
Purity: 0.878 , 40 :: 0.22975708502024292
Purity: 0.878 , 50 :: 0.22941970310391363
Purity: 0.878 , 60 :: 0.22908232118758434
Purity: 0.878 , 70 :: 0.22773279352226722
Purity: 0.878 , 80 :: 0.20242914979757085
Purity: 0.878 , 90 :: 0.20242914979757085
```

```
Purity: 0.879 , 10 :: 0.32793522267206476
Purity: 0.879 , 20 :: 0.28205128205128205
Purity: 0.879 , 30 :: 0.25809716599190285
Purity: 0.879 , 40 :: 0.22975708502024292
Purity: 0.879 , 50 :: 0.22941970310391363
Purity: 0.879 , 60 :: 0.22941970310391363
Purity: 0.879 , 70 :: 0.22807017543859648
Purity: 0.879 , 80 :: 0.20242914979757085
Purity: 0.879 , 90 :: 0.20242914979757085
Purity: 0.88 , 10 :: 0.3309716599190283
Purity: 0.88 , 20 :: 0.2840755735492578
Purity: 0.88 , 30 :: 0.2601214574898785
Purity: 0.88 , 40 :: 0.24392712550607287
Purity: 0.88 , 50 :: 0.22941970310391363
Purity: 0.88 , 60 :: 0.22941970310391363
Purity: 0.88 , 70 :: 0.22807017543859648
Purity: 0.88 , 80 :: 0.20242914979757085
Purity: 0.88 , 90 :: 0.20242914979757085
```

Out[29]:
```
(array([2564.,   13.,  130.,   14.,   15.,   39.,   57.,   89.,   25.,
          18.]),
 array([-1. ,  0.4,  1.8,  3.2,  4.6,  6. ,  7.4,  8.8, 10.2, 11.6, 13. ]),
 <a list of 10 Patch objects>)
```

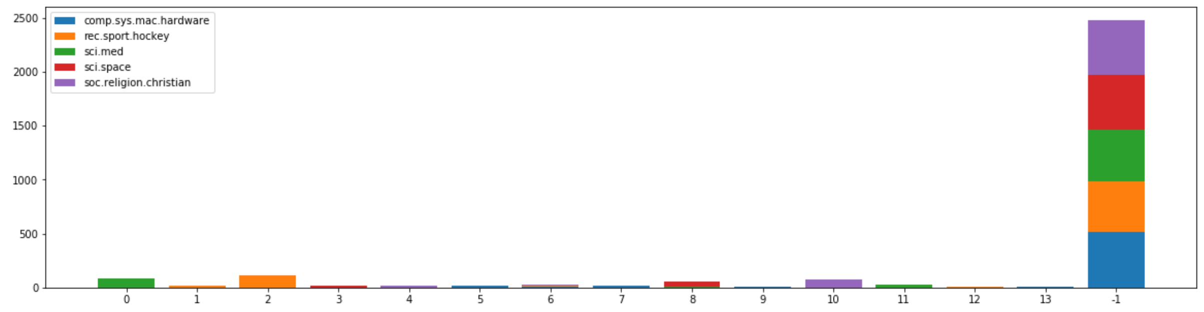


In [30]:
```
opt_eps, opt_min_sample
```

Out[30]:
```
(0.88, 10)
```

In [31]:
```
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

```
Purity: 0.3309716599190283
```



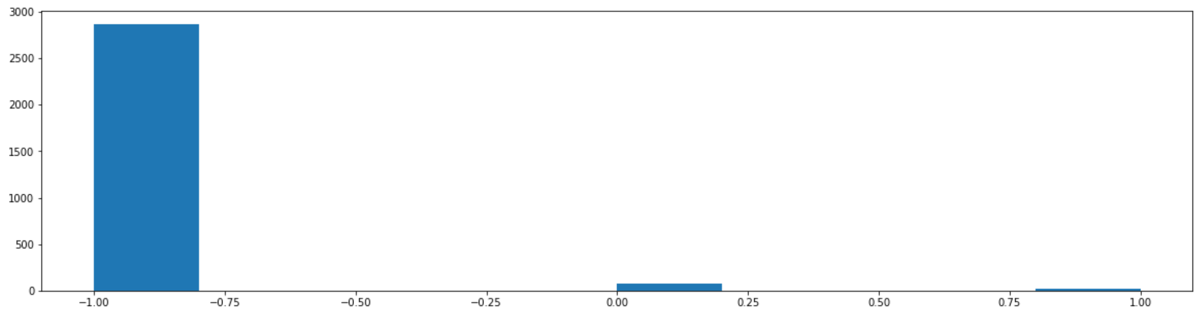
In [37]:
```
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True, max_features=2000, max_df=0.5)
train.vecs = features.fit_transform(train.data)
eps_range = np.arange(0.5,0.9,0.05)
max_purity = 0
for i in range(len(eps_range)):
    for j in range(2, 20, 2):
        clusterer = sklearn.cluster.DBSCAN(eps=eps_range[i], min_samples=j, me
tric='cosine')
        clusts = clusterer.fit_predict(train.vecs)
        score = purity_score(clusts, train.target)
        print('Purity:', eps_range[i], ",", j, "::",  score)
        if( score > max_purity):
            opt_eps = eps_range[i]
            opt_min_sample = j
            max_purity = score
```

```
Purity: 0.5 , 2 :: 0.40350877192982454
Purity: 0.5 , 4 :: 0.30195681511470984
Purity: 0.5 , 6 :: 0.26214574898785425
Purity: 0.5 , 8 :: 0.24932523616734142
Purity: 0.5 , 10 :: 0.2344804318488529
Purity: 0.5 , 12 :: 0.2341430499325236
Purity: 0.5 , 14 :: 0.2321187584345479
Purity: 0.5 , 16 :: 0.22705802968960864
Purity: 0.5 , 18 :: 0.22705802968960864
Purity: 0.55 , 2 :: 0.46390013495276655
Purity: 0.55 , 4 :: 0.354251012145749
Purity: 0.55 , 6 :: 0.29790823211875844
Purity: 0.55 , 8 :: 0.26855600539811064
Purity: 0.55 , 10 :: 0.2506747638326586
Purity: 0.55 , 12 :: 0.24527665317139002
Purity: 0.55 , 14 :: 0.2408906882591093
Purity: 0.55 , 16 :: 0.23515519568151147
Purity: 0.55 , 18 :: 0.23515519568151147
Purity: 0.6000000000000001 , 2 :: 0.5293522267206477
Purity: 0.6000000000000001 , 4 :: 0.4247638326585695
Purity: 0.6000000000000001 , 6 :: 0.3690958164642375
Purity: 0.6000000000000001 , 8 :: 0.3181511470985155
Purity: 0.6000000000000001 , 10 :: 0.28036437246963564
Purity: 0.6000000000000001 , 12 :: 0.25877192982456143
Purity: 0.6000000000000001 , 14 :: 0.2516869095816464
Purity: 0.6000000000000001 , 16 :: 0.2516869095816464
Purity: 0.6000000000000001 , 18 :: 0.24696356275303644
Purity: 0.6500000000000001 , 2 :: 0.4183535762483131
Purity: 0.6500000000000001 , 4 :: 0.4898785425101215
Purity: 0.6500000000000001 , 6 :: 0.446693657219973
Purity: 0.6500000000000001 , 8 :: 0.4139676113360324
Purity: 0.6500000000000001 , 10 :: 0.3562753036437247
Purity: 0.6500000000000001 , 12 :: 0.3083670715249663
Purity: 0.6500000000000001 , 14 :: 0.28711201079622134
Purity: 0.6500000000000001 , 16 :: 0.2759784075573549
Purity: 0.6500000000000001 , 18 :: 0.26956815114709853
Purity: 0.7000000000000002 , 2 :: 0.2921727395411606
Purity: 0.7000000000000002 , 4 :: 0.26619433198380565
Purity: 0.7000000000000002 , 6 :: 0.2941970310391363
Purity: 0.7000000000000002 , 8 :: 0.3282726045883941
Purity: 0.7000000000000002 , 10 :: 0.3815789473684211
Purity: 0.7000000000000002 , 12 :: 0.45883940620782726
Purity: 0.7000000000000002 , 14 :: 0.4001349527665317
Purity: 0.7000000000000002 , 16 :: 0.3714574898785425
Purity: 0.7000000000000002 , 18 :: 0.354251012145749
Purity: 0.7500000000000002 , 2 :: 0.22941970310391363
Purity: 0.7500000000000002 , 4 :: 0.2257085020242915
Purity: 0.7500000000000002 , 6 :: 0.22941970310391363
Purity: 0.7500000000000002 , 8 :: 0.2412280701754386
Purity: 0.7500000000000002 , 10 :: 0.26855600539811064
Purity: 0.7500000000000002 , 12 :: 0.2726045883940621
Purity: 0.7500000000000002 , 14 :: 0.3282726045883941
Purity: 0.7500000000000002 , 16 :: 0.3184885290148448
Purity: 0.7500000000000002 , 18 :: 0.4483805668016194
Purity: 0.8000000000000003 , 2 :: 0.2054655870445344
Purity: 0.8000000000000003 , 4 :: 0.2054655870445344
Purity: 0.8000000000000003 , 6 :: 0.20715249662618085
```

```
Purity: 0.8000000000000003 , 8 :: 0.2101889338731444
Purity: 0.8000000000000003 , 10 :: 0.21524966261808368
Purity: 0.8000000000000003 , 12 :: 0.2216599190283401
Purity: 0.8000000000000003 , 14 :: 0.22435897435897437
Purity: 0.8000000000000003 , 16 :: 0.23110661268556004
Purity: 0.8000000000000003 , 18 :: 0.23144399460188933
Purity: 0.8500000000000003 , 2 :: 0.20647773279352227
Purity: 0.8500000000000003 , 4 :: 0.20647773279352227
Purity: 0.8500000000000003 , 6 :: 0.20647773279352227
Purity: 0.8500000000000003 , 8 :: 0.20647773279352227
Purity: 0.8500000000000003 , 10 :: 0.20647773279352227
Purity: 0.8500000000000003 , 12 :: 0.20647773279352227
Purity: 0.8500000000000003 , 14 :: 0.20647773279352227
Purity: 0.8500000000000003 , 16 :: 0.20647773279352227
Purity: 0.8500000000000003 , 18 :: 0.20647773279352227
```

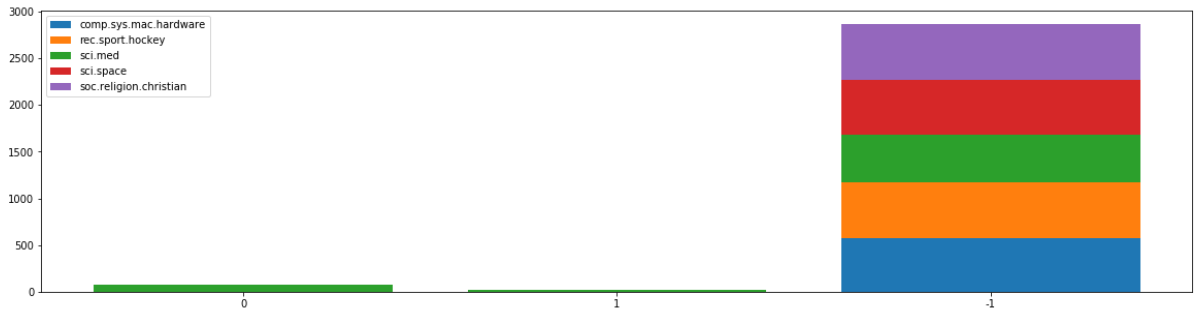
In [38]:
```
opt_min_sample = 10
opt_eps = 0.5

clusterer = sklearn.cluster.DBSCAN(eps=opt_eps, min_samples=opt_min_sample, me
tric='cosine')
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

Out[38]:
```
(array([2869.,    0.,    0.,    0.,    0.,   73.,    0.,    0.,    0.,
          22.]),
 array([-1. , -0.8, -0.6, -0.4, -0.2,  0. ,  0.2,  0.4,  0.6,  0.8,  1. ]),
 <a list of 10 Patch objects>)
```



In [39]:
```
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

```
Purity: 0.2344804318488529
```



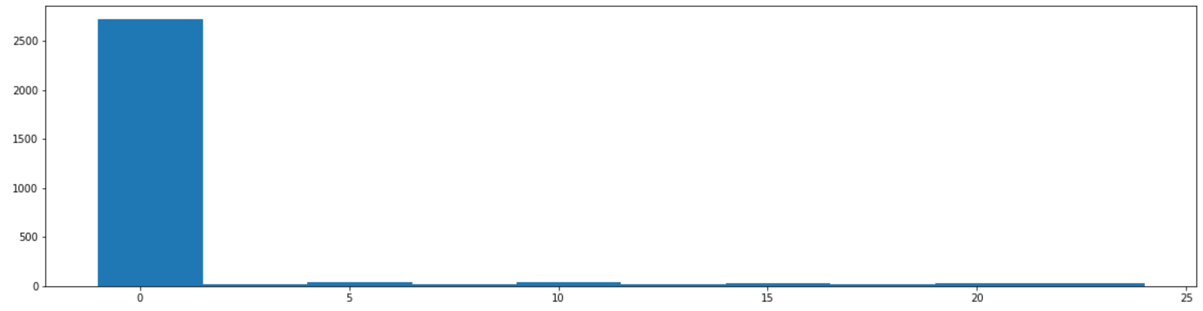
In [52]:
```
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True, max_features=300)
train.vecs = features.fit_transform(train.data)
eps_range = np.arange(0.5,0.9,0.05)
max_purity = 0
for i in range(len(eps_range)):
    for j in range(2, 20, 2):
        clusterer = sklearn.cluster.DBSCAN(eps=eps_range[i], min_samples=j, me
tric='cosine')
        clusts = clusterer.fit_predict(train.vecs)
        score = purity_score(clusts, train.target)
        print('Purity:', eps_range[i], ",", j, "::",  score)
        if( score > max_purity):
            opt_eps = eps_range[i]
            opt_min_sample = j
            max_purity = score
```

```
Purity: 0.5 , 2 :: 0.24865047233468285
Purity: 0.5 , 4 :: 0.23515519568151147
Purity: 0.5 , 6 :: 0.2479757085020243
Purity: 0.5 , 8 :: 0.2540485829959514
Purity: 0.5 , 10 :: 0.2567476383265857
Purity: 0.5 , 12 :: 0.37685560053981104
Purity: 0.5 , 14 :: 0.36403508771929827
Purity: 0.5 , 16 :: 0.3893387314439946
Purity: 0.5 , 18 :: 0.3630229419703104
Purity: 0.55 , 2 :: 0.2344804318488529
Purity: 0.55 , 4 :: 0.2257085020242915
Purity: 0.55 , 6 :: 0.22435897435897437
Purity: 0.55 , 8 :: 0.23751686909581646
Purity: 0.55 , 10 :: 0.23954116059379219
Purity: 0.55 , 12 :: 0.23987854251012145
Purity: 0.55 , 14 :: 0.2388663967611336
Purity: 0.55 , 16 :: 0.24662618083670715
Purity: 0.55 , 18 :: 0.33535762483130904
Purity: 0.6000000000000001 , 2 :: 0.21086369770580296
Purity: 0.6000000000000001 , 4 :: 0.2101889338731444
Purity: 0.6000000000000001 , 6 :: 0.21120107962213225
Purity: 0.6000000000000001 , 8 :: 0.2125506072874494
Purity: 0.6000000000000001 , 10 :: 0.2165991902834008
Purity: 0.6000000000000001 , 12 :: 0.22098515519568152
Purity: 0.6000000000000001 , 14 :: 0.22064777327935223
Purity: 0.6000000000000001 , 16 :: 0.22267206477732793
Purity: 0.6000000000000001 , 18 :: 0.2300944669365722
Purity: 0.6500000000000001 , 2 :: 0.20681511470985156
Purity: 0.6500000000000001 , 4 :: 0.2058029689608637
Purity: 0.6500000000000001 , 6 :: 0.20614035087719298
Purity: 0.6500000000000001 , 8 :: 0.20614035087719298
Purity: 0.6500000000000001 , 10 :: 0.20647773279352227
Purity: 0.6500000000000001 , 12 :: 0.20715249662618085
Purity: 0.6500000000000001 , 14 :: 0.20850202429149797
Purity: 0.6500000000000001 , 16 :: 0.20850202429149797
Purity: 0.6500000000000001 , 18 :: 0.21086369770580296
Purity: 0.7000000000000002 , 2 :: 0.20816464237516868
Purity: 0.7000000000000002 , 4 :: 0.20816464237516868
Purity: 0.7000000000000002 , 6 :: 0.20816464237516868
Purity: 0.7000000000000002 , 8 :: 0.20816464237516868
Purity: 0.7000000000000002 , 10 :: 0.20816464237516868
Purity: 0.7000000000000002 , 12 :: 0.20816464237516868
Purity: 0.7000000000000002 , 14 :: 0.20816464237516868
Purity: 0.7000000000000002 , 16 :: 0.20816464237516868
Purity: 0.7000000000000002 , 18 :: 0.20816464237516868
Purity: 0.7500000000000002 , 2 :: 0.2078272604588394
Purity: 0.7500000000000002 , 4 :: 0.2078272604588394
Purity: 0.7500000000000002 , 6 :: 0.2078272604588394
Purity: 0.7500000000000002 , 8 :: 0.2078272604588394
Purity: 0.7500000000000002 , 10 :: 0.2078272604588394
Purity: 0.7500000000000002 , 12 :: 0.2078272604588394
Purity: 0.7500000000000002 , 14 :: 0.2078272604588394
Purity: 0.7500000000000002 , 16 :: 0.2078272604588394
Purity: 0.7500000000000002 , 18 :: 0.2078272604588394
Purity: 0.8000000000000003 , 2 :: 0.2078272604588394
Purity: 0.8000000000000003 , 4 :: 0.2078272604588394
Purity: 0.8000000000000003 , 6 :: 0.2078272604588394
```

```
Purity: 0.8000000000000003 , 8 :: 0.2078272604588394
Purity: 0.8000000000000003 , 10 :: 0.2078272604588394
Purity: 0.8000000000000003 , 12 :: 0.2078272604588394
Purity: 0.8000000000000003 , 14 :: 0.2078272604588394
Purity: 0.8000000000000003 , 16 :: 0.2078272604588394
Purity: 0.8000000000000003 , 18 :: 0.2078272604588394
Purity: 0.8500000000000003 , 2 :: 0.2078272604588394
Purity: 0.8500000000000003 , 4 :: 0.2078272604588394
Purity: 0.8500000000000003 , 6 :: 0.2078272604588394
Purity: 0.8500000000000003 , 8 :: 0.2078272604588394
Purity: 0.8500000000000003 , 10 :: 0.2078272604588394
Purity: 0.8500000000000003 , 12 :: 0.2078272604588394
Purity: 0.8500000000000003 , 14 :: 0.2078272604588394
Purity: 0.8500000000000003 , 16 :: 0.2078272604588394
Purity: 0.8500000000000003 , 18 :: 0.2078272604588394
```

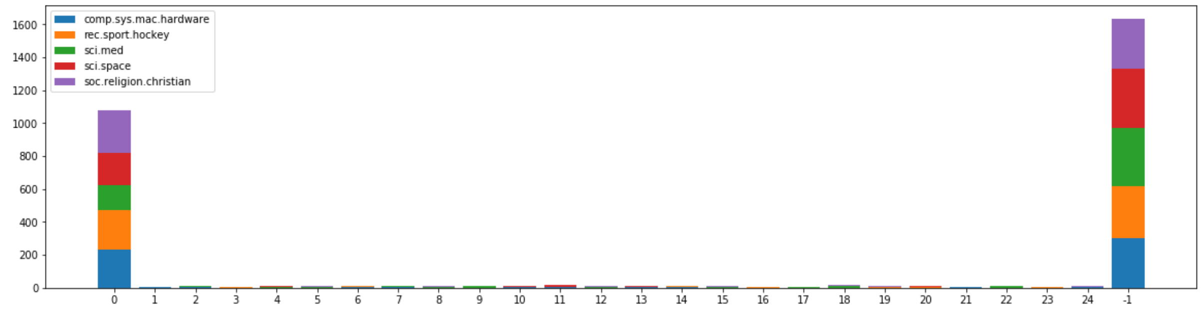
In [53]:
```python
opt_min_sample = 10
opt_eps = 0.5

clusterer = sklearn.cluster.DBSCAN(eps=opt_eps, min_samples=opt_min_sample, me
tric='cosine')
clusts = clusterer.fit_predict(train.vecs)
plt.hist(clusts)
```

Out[53]:
```
(array([2722.,   17.,   36.,   23.,   38.,   21.,   28.,   21.,   28.,
          30.]),
 array([-1. ,  1.5,  4. ,  6.5,  9. , 11.5, 14. , 16.5, 19. , 21.5, 24. ]),
 <a list of 10 Patch objects>)
```



In [54]:
```python
cluster_purity(clusts, train.target)
print('Purity:', purity_score(clusts, train.target))
```

```
Purity: 0.2567476383265857
```



```
In [ ]: # interpretation:
        I changed features max_features to 300, and the purity score increased from 0.
        234 to 0.257
```

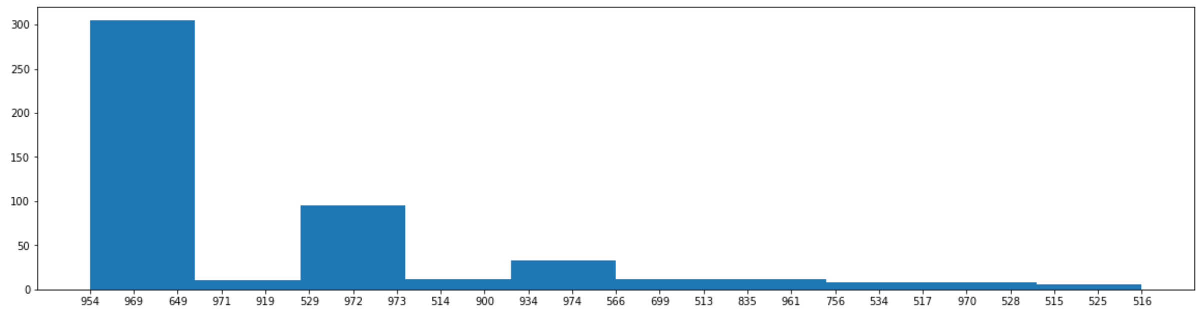
Agglomerative Clustering

```
In [40]: num_points = 500
         data = train.data[:num_points]
         target= train.target[:num_points]
         features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
         =True)
         vecs = features.fit_transform(data)
         clusterer = sklearn.cluster.AgglomerativeClustering()
         clusts = np.array(clusterer.fit_predict(vecs.toarray()))
```

```
In [41]: def clustering_from_tree(clusterer, i):
           n_samples = clusterer.n_leaves_
           nodes = clusterer.children_
           pclusts = np.arange(n_samples)
           def label_clust(nid, label):
             if nid < n_samples:
               pclusts[nid] = label
             else:
               lchild = nodes[nid-n_samples][0]
               label_clust(lchild, label)
               rchild = nodes[nid-n_samples][1]
               label_clust(rchild, label)

           #simulate the clustering
           for j in range(i):
             lchild = nodes[j][0]
             label_clust(lchild, n_samples+j)
             rchild = nodes[j][1]
             label_clust(rchild, n_samples+j)
           return pclusts
```

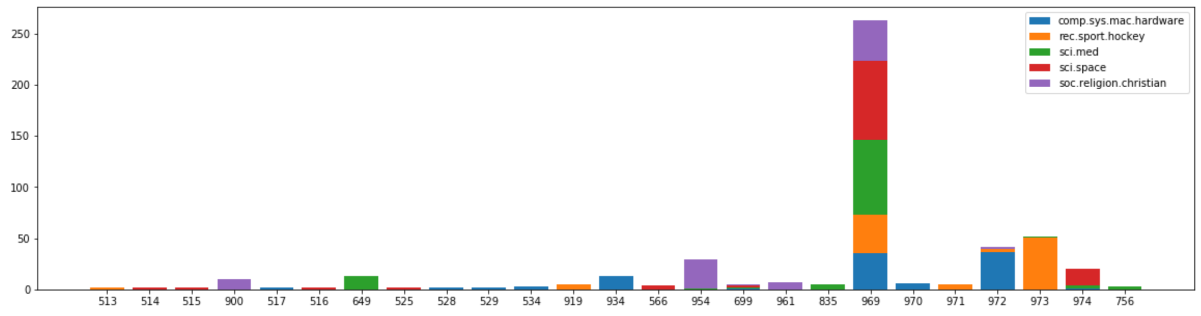
```
In [42]: clusts = clustering_from_tree(clusterer, 475)
         plt.hist([str(c) for c in clusts])
         print("Number of clusters", len(list(set(clusts))))
```

```
Number of clusters 25
```



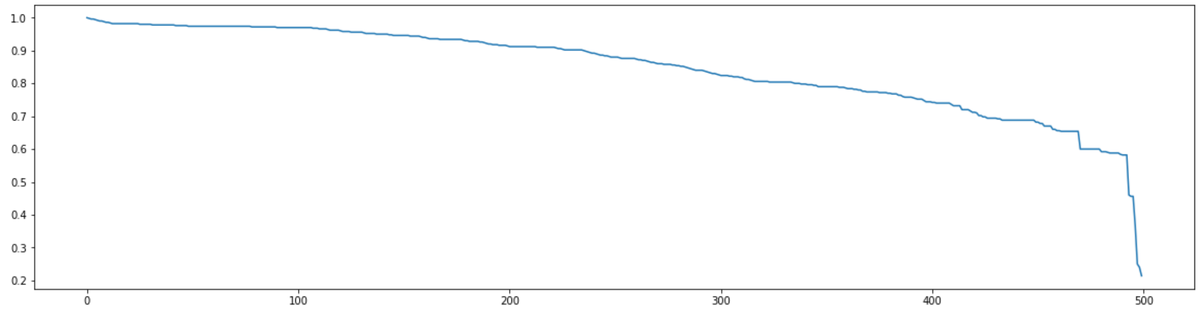
In [43]:
```
cluster_purity(clusts, target)
print('Purity:', purity_score(clusts, target))
```

```
Purity: 0.6
```



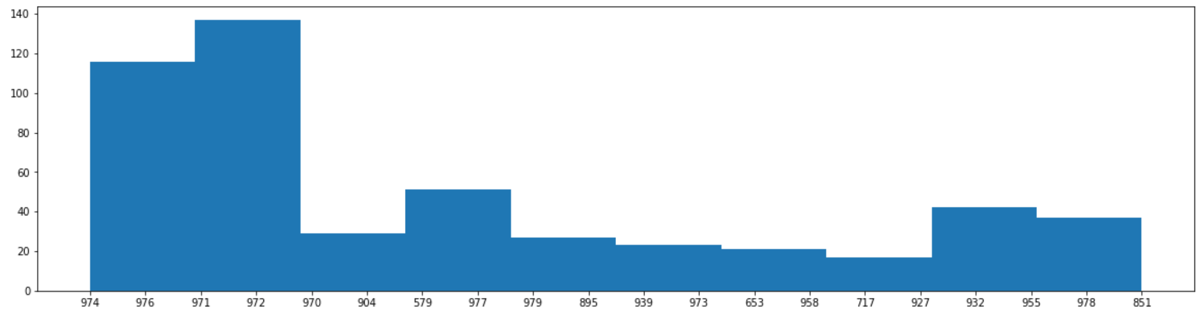
In [44]:
```
numcs = []
purities = []
for i in range(num_points):
  clusts = clustering_from_tree(clusterer, i)
  numc = len(list(set(clusts)))
  numcs.append(numc)
  purities.append(purity_score(clusts, target))
plt.plot(range(num_points), purities)
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x15623ef6d48>]
```



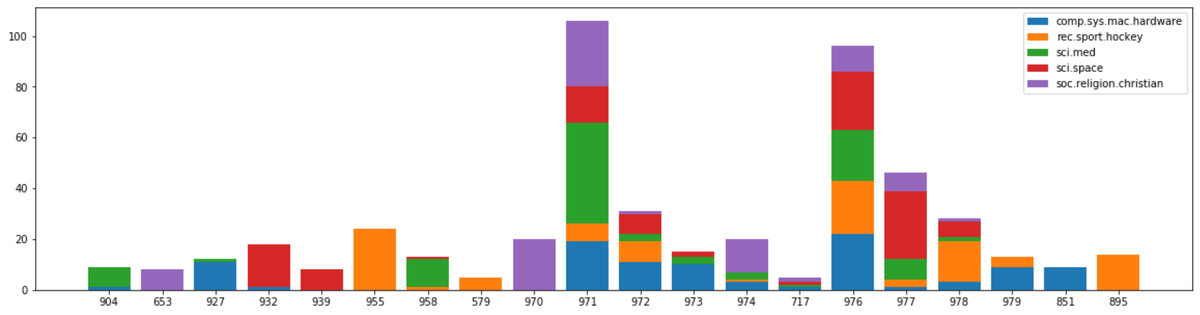
In [49]:
```
clusts = clustering_from_tree(clusterer, 480)
plt.hist([str(c) for c in clusts])
print("Number of clusters", len(list(set(clusts))))
```

```
Number of clusters 20
```



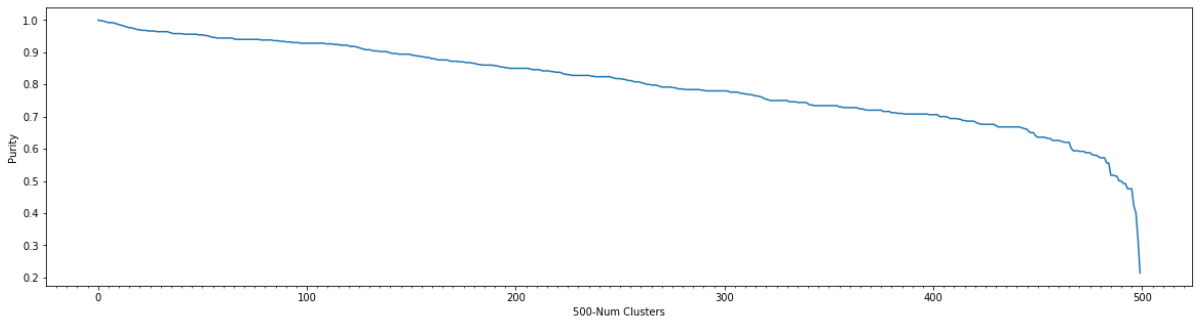
In [50]:
```
cluster_purity(clusts, target)
print('Purity:', purity_score(clusts, target))
```

```
Purity: 0.572
```



In [51]:
```
from matplotlib.ticker import (MultipleLocator, AutoMinorLocator)

numcs = []
purities = []
for i in range(num_points):
  clusts = clustering_from_tree(clusterer, i)
  numc = len(list(set(clusts)))
  numcs.append(numc)
  purities.append(purity_score(clusts, target))

fig, ax = plt.subplots()
ax.plot(range(num_points), purities)

# For the minor ticks, use no labels; default NullFormatter.
ax.xaxis.set_minor_locator(MultipleLocator(5))
ax.yaxis.set_minor_locator(MultipleLocator(5))
ax.set_xlabel('500-Num Clusters')
ax.set_ylabel('Purity')
```

Out[51]:
```
Text(0, 0.5, 'Purity')
```

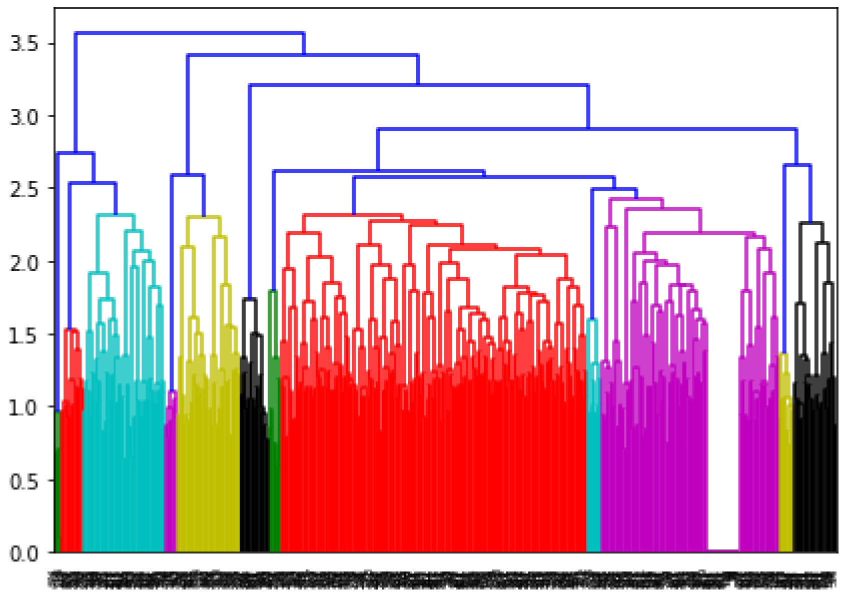


In [52]:
```
import scipy.cluster.hierarchy as sch
```

In [53]:
```
print(clusterer.fit_predict(vecs.toarray()))
```

```
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0
 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0
 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
```

In [54]:
```
plt.figure(figsize=(7,5))
dend = sch.dendrogram(sch.linkage(vecs.toarray(), method='ward'))
plt.show()
```

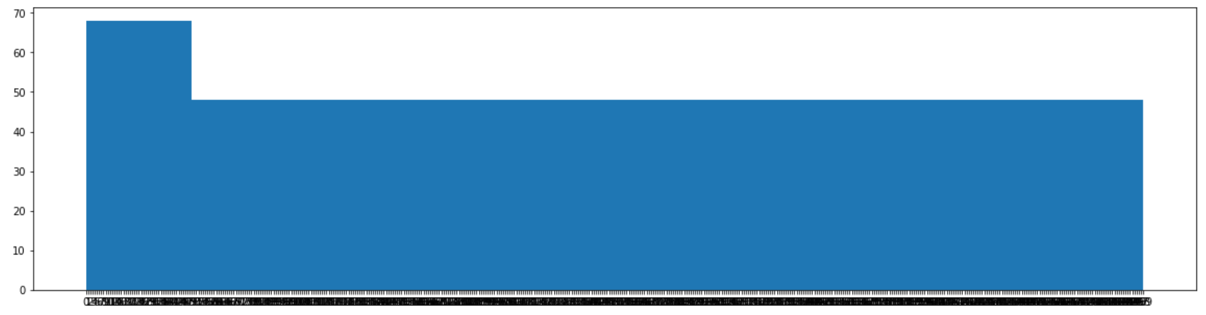


In [45]:
```
num_points = 500
data = train.data[:num_points]
target= train.target[:num_points]
features = TfidfVectorizer(ngram_range=(1,2), stop_words= 'english', lowercase
=True, max_features=300)
vecs = features.fit_transform(data)
clusterer = sklearn.cluster.AgglomerativeClustering(affinity='euclidean', link
age="ward")
clusts = np.array(clusterer.fit_predict(vecs.toarray()))
```

In [46]:
```
def clustering_from_tree(clusterer, i):
  n_samples = clusterer.n_leaves_
  nodes = clusterer.children_
  pclusts = np.arange(n_samples)
  def label_clust(nid, label):
    if nid < n_samples:
      pclusts[nid] = label
    else:
      lchild = nodes[nid-n_samples][0]
      label_clust(lchild, label)
      rchild = nodes[nid-n_samples][1]
      label_clust(rchild, label)

  #simulate the clustering
  for j in range(i):
    lchild = nodes[j][0]
    label_clust(lchild, n_samples+j)
    rchild = nodes[j][1]
    label_clust(rchild, n_samples+j)
  return pclusts
```

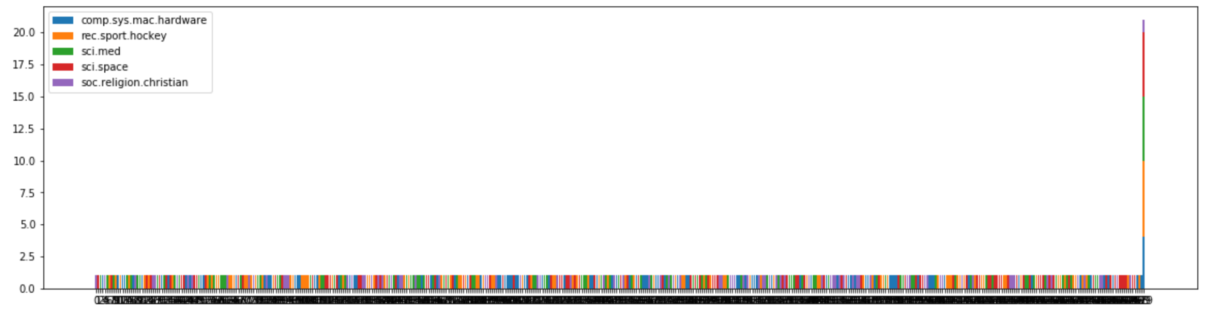
In [47]:
```
clusts = clustering_from_tree(clusterer, 20)
plt.hist([str(c) for c in clusts])
print("Number of clusters", len(list(set(clusts))))
```

```
Number of clusters 480
```



In [48]:
```
cluster_purity(clusts, target)
print('Purity:', purity_score(clusts, target))
```

```
Purity: 0.97
```



In [ ]:
```
# interpretation:
I changed features max_features to 300, clusters to 20, and the purity score i
s 0.97.
```