Classification and Sentiment analysis of Reddit's posts

October 2, 2022

0.1 Progetto sulla classificazione e il Sentiment analysis dei post di Reddit

Nome: Giuseppe

Cognome: Condorelli

Matricola: 1000001910

Corso: Social Media Management

Docente: Antonino Furnari

Anno Accademico: 2021-2022

1 Introduzione

Il progetto proposto si pone il problema di **analizzare** i post presenti nel **Social Media Reddit**.

In particolare lo script **estrarrà i testi dai post** del social media e dopodichè, attraverso **machine learning**, si occuperà del loro analizzamento fornendo all'utente il **sentiment analysis** ed inoltre la **categoria** a quale più appartiene.

Questo progetto risulta molto utile per gli utenti che vogliono avere una ampia e veloce idea dei post della piattaforma che, invece di doverli leggere per intero, possono subito avere, tramite una comoda dashboard, un'idea sull'argomento trattato ed un indicatore di negatività/positivà della discussione del post.

2 Dati

I dati usati per la realizzazione del progetto si dividono in 3 parti.

2.1 Topic classification Dataset

Per la classificazione del topic vengono valutati due dataset per valutare quale sia il più accurato tra i due e scegliere il migliore per lo script finale.

2.1.1 BBC Full Text Document Classification

Questo dataset è stato ottenuto documenti di BBC e classifica le seguenti categorie:

Business

Entertainment

Politcs

Sport

Tech

Contiene solo 2'225 documenti nelle cinque categorie sopra citate

Il dataset è stato ottenuto tramite la piattaforma kaggle al seguente link: BBC Full Text Document Classification

- D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006.All rights, including copyright, in the content of the original articles are owned by the BBC.

2.1.2 Yahoo! Answers Topic Classification

Questo dataset è stato creato usando le migliori risposte dalla piattaforma Yahoo! e contiene le seguenti categorie:

Society & Culture

Science & Mathematics

Health

Education & Reference

Computers & Internet

Sports

Business & Finance

Entertainment & Music

Family & Relationships

Politics & Government

Contiene 140'000 training samples e 6'000 testing samples per categoria per un totale di 1'400'000 training samples e 60'000 testing samples.

Il dataset è stato ottenuto tramite la piattaforma kaggle al seguente link: Yahoo! Answers Topic Classification

2.2 Sentiment Analysis Dataset

Per il sentiment analysis del testo ho optato nel fare il training di un altro modello di machine learning (piuttosto di usare VADER) usando un altro Dataset fornito, sempre dalla piattaforma kaggle, al seguente link: Sentiment140 dataset with 1.6 million tweets

Il dataset contiene 1'600'000 twetts estratti usando le api di Twitter

In particolare ad ogni sample assegna un valore col seguente criterio:

0 = negative

2 = neutral

4 = positive

-Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(2009), p.12.

2.3 Post di reddit

Infine gli ultimi dati vengono prevelevati direttamente da Reddit grazie all'utilizzo delle sue api.

3 Metodo proposto

3.0.1 Dichiarazione delle librerie utilizzate

Nella realizzazione dello script saranno utilizzate molte librerie, in particolare quelle offerte da sklearn che fornisce tantissimi modelli per affrontare problemi di machine learning nonchè dei metodi per valutarne la correttezza.

Inoltre sarà usato anche os, pandas e csv per trattatare con i dataset e matplotlib e seaborn per poter stampare la confusion matrix in un formato più leggibile

[1]:
```
import pandas as pd
import numpy as np
import os
import csv
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

3.1 Topics machine learning classifier

3.1.1 BBC Full Text Document Classification

Per affrontare il problema riguardo i topics di ogni post ho optato per l'utilizzo del KNN, ovvero un algoritmo di classificazione che funziona molto bene per elementi che si trovano vicini nello spazio.

[2]:
```
PATH_CLASSIFICATION = "bbc/"

TOPICS_BBC = "business entertainment politics sport tech".split()

TOPICS_YAHOO = ["Society & Culture", "Science & Mathematics", "Health",␣
 ↪"Education & Reference",
            "Computers & Internet", "Sports", "Business & Finance",␣
 ↪"Entertainment & Music",
```

```
            "Family & Relationships", "Politics & Governemnt"]
```

3.1.2 Dataset to csv

Qui di seguito ho implementato un piccolo algoritmo che legge i file txt contenuti nelle rispettive cartelle di "classe" e li scrive all'interno di un comodo file csv.

In questo modo il dataset risulta molto più organizzato (dato che avrà una colonna con il testo ed un'altra con il topic di appartenenza) e più facile da utilizzare per il training e il test.

[3]:
```
header = ['data', 'topic']
file = open("bbc/topics_dataset.csv", 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
count = 0
for g in TOPICS_BBC:
    for filename in os.listdir(PATH_CLASSIFICATION + f'{g}'):
        textname = PATH_CLASSIFICATION + f'{g}/{filename}'
        textname = os.fspath(textname)
        fp = open(textname, 'r')

        count += 1
        to_append = fp.read()
        file = open("bbc/topics_dataset.csv", 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow([to_append, f'{g}'])
```

3.1.3 TF - IDF

Nel codice seguente viene letto il csv creato precedentemente e, tramite TfidVectorizer importato dalla librearia sklearn vengono pre processati le features con la metodologia TF-IDF.

Infine i vettori features e labels vengono divisi con un split in training e test set.

[4]:
```
data = pd.read_csv("bbc/topics_dataset.csv", encoding= 'unicode_escape')
# Dropping unneccesary columns
data.head()

topic_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(topic_list)

#Applying TF_IDF
corpus = data.iloc[:, 0]
vectorizer = TfidfVectorizer(stop_words='english', use_idf=True)
topic_fitted_vectorizer_bbc = vectorizer.fit(corpus)
X = topic_fitted_vectorizer_bbc.transform(corpus)
```

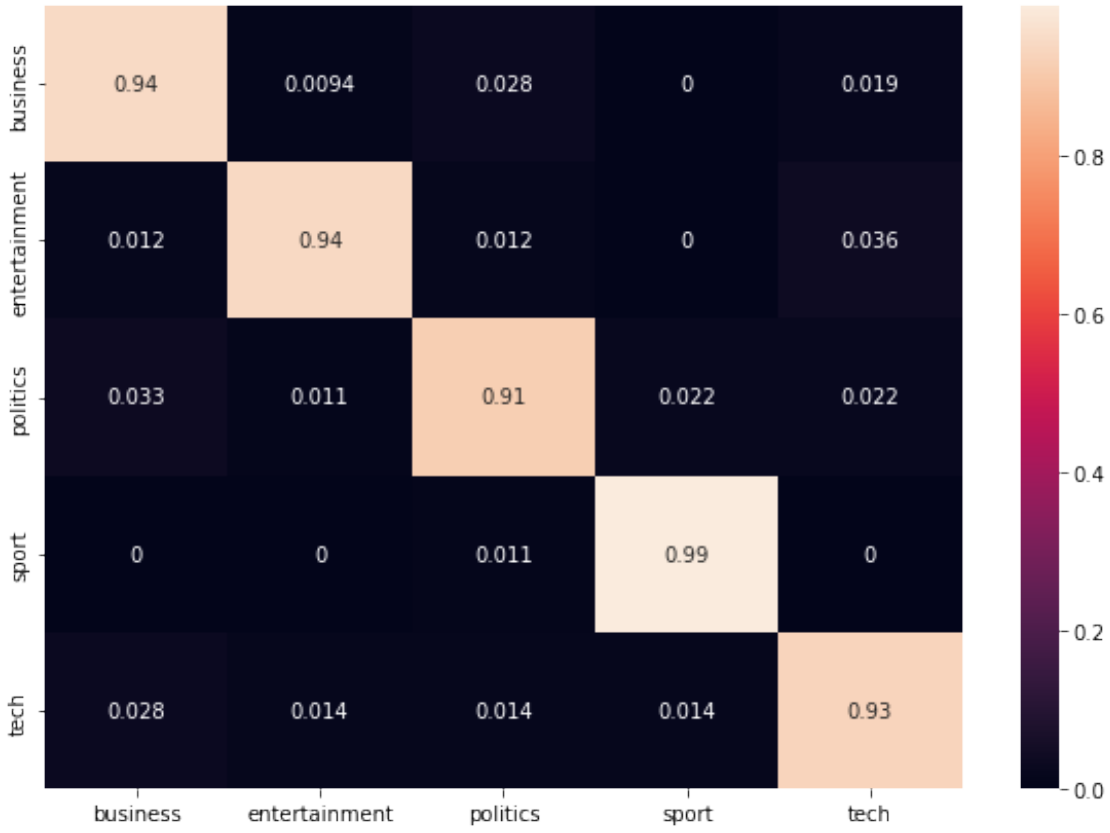
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2)
```

3.1.4 KNN model training and evaluation

Qui viene fatto il training del modello KNN ed in seguito, grazie al classification report e confusion matrix di sklearn, ne stampo la metrica per valutare la correttezza del modello

[5]:
```
KNN_model = KNeighborsClassifier(n_neighbors=4)

KNN_model.fit(X_train, y_train)

KNN_prediction = KNN_model.predict(X_test)

print("KNN")
print(classification_report(KNN_prediction, y_test, target_names= TOPICS_BBC))
df_cm = pd.DataFrame(confusion_matrix(KNN_prediction, y_test,␣
 ↪normalize='true'), index = [i for i in TOPICS_BBC],
                columns = [i for i in TOPICS_BBC])
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True)
plt.show()
```

```
KNN
              precision    recall  f1-score   support

     business       0.94      0.94      0.94       106
entertainment       0.96      0.94      0.95        84
     politics       0.93      0.91      0.92        91
        sport       0.97      0.99      0.98        93
         tech       0.90      0.93      0.92        71

     accuracy                           0.94       445
    macro avg       0.94      0.94      0.94       445
 weighted avg       0.94      0.94      0.94       445
```



3.1.5 Yahoo! Answers Topic Classification

3.1.6 TF-IDF

Qui data che la grandezza del Dataset ho deciso di diminuire il numero di features a 7000, il che permette di avere un traning più rapido del modello non compromettendo troppo la sua accuracy

[6]:
```
data = pd.read_csv("YahooDataset/train.csv", encoding= 'unicode_escape')
#data.head()# Dropping unneccesary columns

topic_list = data.iloc[:, 0]

topic_list = [x - 1 for x in topic_list]

#Applying TF_IDF
corpus = data.iloc[:, 3].values.astype('U')

vectorizer = TfidfVectorizer(stop_words='english', use_idf=True, max_features =␣
↪7000)
topic_fitted_vectorizer_yahoo = vectorizer.fit(corpus)
```

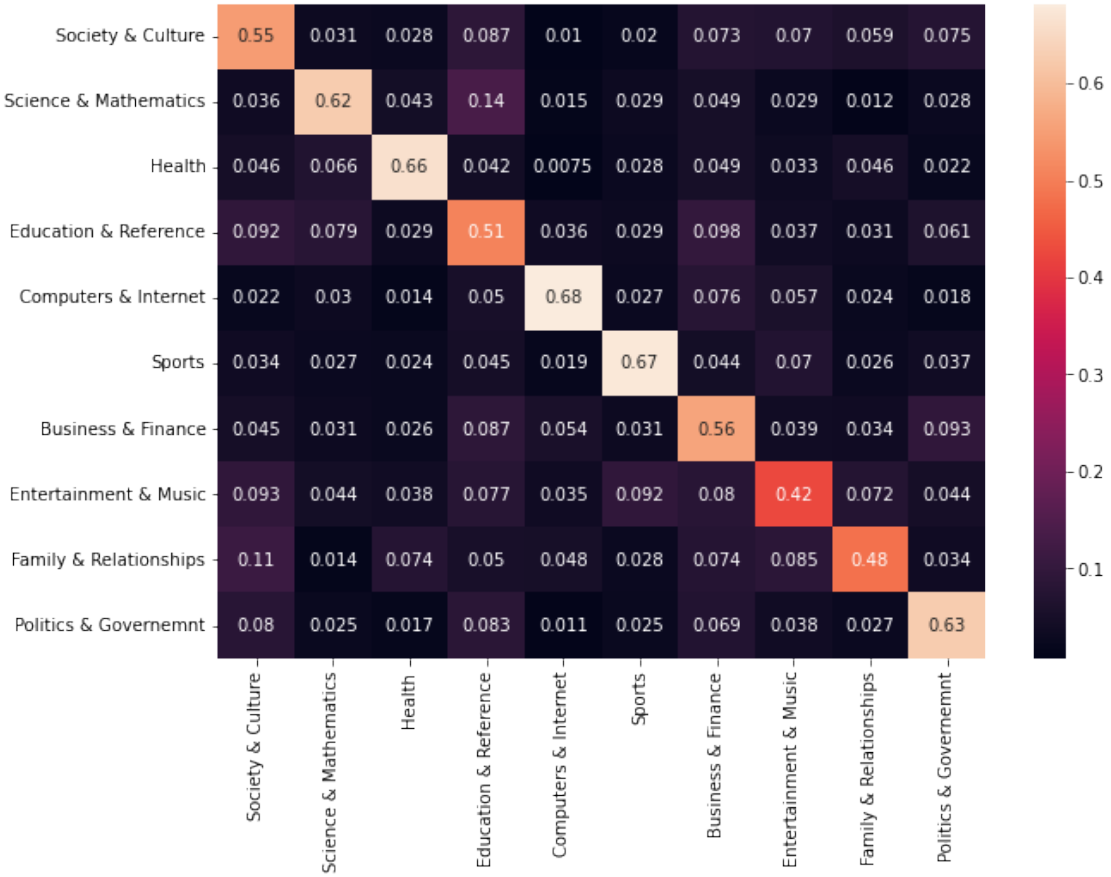
```
X_train = topic_fitted_vectorizer_yahoo.transform(corpus)
y_train = topic_list

data = pd.read_csv("YahooDataset/test.csv", encoding= 'unicode_escape')
#data.head()# Dropping unneccesary columns

topic_list = data.iloc[:, 0]

topic_list = [x - 1 for x in topic_list]

#Applying TF_IDF
corpus = data.iloc[:, 3].values.astype('U')

X_test = topic_fitted_vectorizer_yahoo.transform(corpus)
y_test = topic_list
```

3.1.7 Linear Support Vector Classifier model training and evaluation

[7]:
```
from sklearn.svm import LinearSVC
svc_model = LinearSVC()

svc_model.fit(X_train, y_train)

svc_prediction = svc_model.predict(X_test)

print("SVC")
print(classification_report(svc_prediction, y_test, target_names= TOPICS_YAHOO))
df_cm = pd.DataFrame(confusion_matrix(svc_prediction, y_test,␣
 ↪normalize='true'), index = [i for i in TOPICS_YAHOO],
                columns = [i for i in TOPICS_YAHOO])
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True)
plt.show()
```

```
SVC
                      precision    recall  f1-score   support

     Society & Culture       0.40      0.55      0.47      4443
 Science & Mathematics       0.67      0.62      0.65      6490
                Health       0.69      0.66      0.67      6230
 Education & Reference       0.33      0.51      0.40      3969
  Computers & Internet       0.77      0.68      0.72      6734
                Sports       0.67      0.67      0.67      5979
    Business & Finance       0.37      0.56      0.45      3985
 Entertainment & Music       0.54      0.42      0.48      7629
Family & Relationships       0.68      0.48      0.56      8541
 Politics & Governemnt       0.63      0.63      0.63      5999
```

```
              accuracy                           0.58     59999
             macro avg       0.58      0.58      0.57     59999
          weighted avg       0.60      0.58      0.58     59999
```



3.1.8 TopicClassify

Qui dichiaro ed implemento un semplice metodo che transforma un vettore di stringhe in **vettori di features** trattabili dal modello allenato in precedenza che successivamente saranno classificate nel topic più opportuno

[8]:
```
def topicClassifyBbc(text):
    X = topic_fitted_vectorizer_bbc.transform([text])
    return encoder.inverse_transform(KNN_model.predict(X))[0]
```

[9]:
```
def topicClassifyYahoo(text):
    X = topic_fitted_vectorizer_yahoo.transform([text])
    return TOPICS_YAHOO[svc_model.predict(X)[0]]
```

3.2 Sentiment Analysis

Il problema di sentiment analysis del testo viene affrontato come un problema di regressione, dunque si cerca di fare una stima quanto più precisa dello score di negatività o positività (che va da 0 a 4) per fornire all'utente un feedback quanto più preciso riguardo i post di reddit visualizzati.

Di seguito viene implementata anche una banale funzione **index** che ha lo scopo di arrotondare il valore ottenuto dal regressore in uno vicino alle labels del dataset utilizzato (0 : negative, 2 : neutral, 4 : positive)

[10]:
```
from sklearn.metrics import r2_score

from sklearn.metrics import explained_variance_score

SENTIMENT = {0 : 'negative', 2: 'neutral',  4:'positive'}

def index(x):
    if x < 1.5 : return 0
    elif x >= 1.5 and x < 2.5 : return 2
    else : return 4
```

3.3 TF - IDF

Così com'è stato fatto per il classificatore di topics, anche per il regressore vengono create un set di features utilizzando la metodologia TF - IDF

[11]:
```
data = pd.read_csv("sentiment_dataset.csv")

y = data.iloc[:, 0]

#Applying TF_IDF
corpus = data.iloc[:, -1]
vectorizer = TfidfVectorizer(stop_words='english', use_idf=False, max_features␣
 ↪= 15000)
sentiment_fitted_vectorizer = vectorizer.fit(corpus)
X = sentiment_fitted_vectorizer.transform(corpus)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2)
```

3.4 Regression model training

[12]:
```
reg_model = LinearRegression()

reg_model = reg_model.fit(X_train, y_train)
```

3.5 Regression model evaluation

Qui viene analizzato lo score r2 per valutare quanto il modello utilizzato per affrontare il problema di regressione sia preciso.

In particolare per valori vicini all'1 avremo un modello perfetto, mentre avvicinandoci allo 0 il contrario.

[13]:
```
reg_prediction = reg_model.predict(X_test)

print("LinearRegression")

print("R2 score: " + str(r2_score(y_test, reg_prediction)))
```

```
LinearRegression
R2 score: 0.3575692226357622
```

[14]:
```
def sentimentAnalysis(text):
    X = sentiment_fitted_vectorizer.transform([text])
    return [SENTIMENT[index(reg_model.predict(X))], reg_model.predict(X)]
```

3.6 Reddit posts retriever

[15]:
```
import praw
from pretty_html_table import build_table
from IPython.display import display, HTML
from praw.models import MoreComments

# Reddit api key
KEY:str = "4Eok9mDfRpJQG82rqUdc-g"

# Reddit api secret
SECRET:str = "bugY82JB2QcfvjJpJU72ePM9Net7oQ"

reddit = praw.Reddit(client_id = KEY, client_secret = SECRET,  user_agent =␣
 ↪"SocialMediaTutorial")
```

3.7 Reddit's topic classifier and sentiment analyzer

3.8 1st Attempt

Nello script di seguito utilizzo sia il modello allenato con il dataset BBC che quello con il dataset di Yahoo.

[16]:
```
subreddit = reddit.subreddit("news")

data = pd.DataFrame(columns=['Titolo', 'Link', 'Topic 1', 'Topic 2',␣
 ↪'SentimentScore', 'Sentiment', 'Up'])

for submission in subreddit.new(limit = 10):
    text = submission.title + submission.selftext
    submission.comments.replace_more(limit=10)
    for comment in submission.comments.list():
        text += comment.body
```

```
    sentiment = sentimentAnalysis(text)
    data.loc[len(data)] = [submission.title, submission.shortlink,␣
 ↪topicClassifyBbc(text),topicClassifyYahoo(text), sentiment[1][0],␣
 ↪sentiment[0], submission.ups]

display(HTML("<h1>SUBREDDIT " + "news" +" </h1>" + build_table(data,␣
 ↪'blue_light')))
```

```
<IPython.core.display.HTML object>
```

3.8.1 Valutazione dei due modelli di Topic Classification

Confrontando i risultati dei due modelli di topic possiamo arrivare alla conclusione che il modello allenato con il dataset BBC fornisce spesso risultati discordanti con il modello allenato con il dataset di Yahoo.

Questo era già abbastanza prevedibile data la scarsità di samples nonchè il numero ridotto di Topics nel primo.

3.9 Final script

In questo script finale non verrà utilizzato più il modello di classificazione allenato con bbc ed in più viene fatto un piccolo refactoring per dividere le responsabilità dello script

[17]:
```
def RedditAnalyzer(subredditName, lim = 10):

    subreddit = reddit.subreddit(subredditName)

    data = pd.DataFrame(columns=['Titolo', 'Link', 'Topic', 'SentimentScore',␣
 ↪'Sentiment', 'Up'])

    for submission in subreddit.new(limit = lim):
        text = submission.title + submission.selftext
        submission.comments.replace_more(limit=10)
        for comment in submission.comments.list():
            text += comment.body
        sentiment = sentimentAnalysis(text)
        data.loc[len(data)] = [submission.title, submission.shortlink,␣
 ↪topicClassifyYahoo(text), sentiment[1][0], sentiment[0], submission.ups]
    return data

def displayTable(data, subreddit):
    display(HTML("<h1>SUBREDDIT " + subreddit +" </h1>" + build_table(data,␣
 ↪'blue_light')))
```

4 Risultati

L'algoritmo proposto riesce a dare una stima descrittiva abbastanza veritiera dei post estratti da Reddit.

Inoltre tramite le informazioni che lo script ci fornisce è possibile anche fare una analisi molto interessante del subreddit preso in osservazione.

Di seguito proporrò un suo possibile utilizzo di analisi dati.

4.1 Analisi dati approfondita subreddit News

[18]:
```
newsAnalysis = RedditAnalyzer("news", 100)
```

[19]:
```
displayTable(newsAnalysis, "news")
```

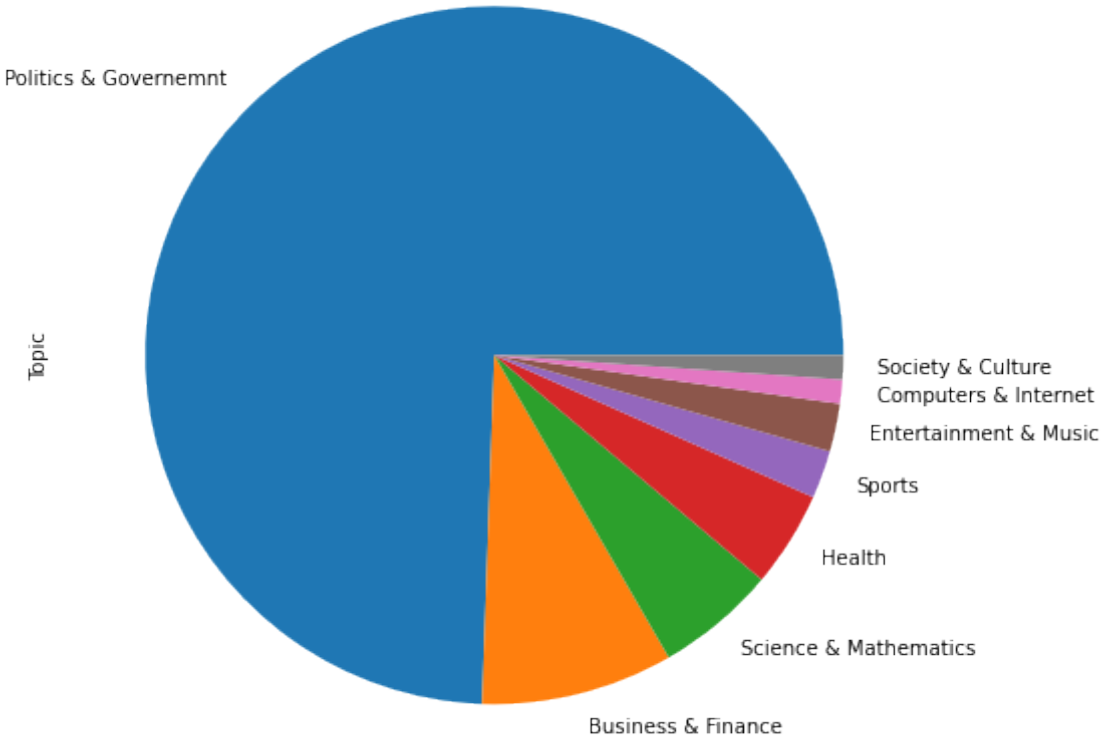
```
<IPython.core.display.HTML object>
```

4.2 Analisi dei topic

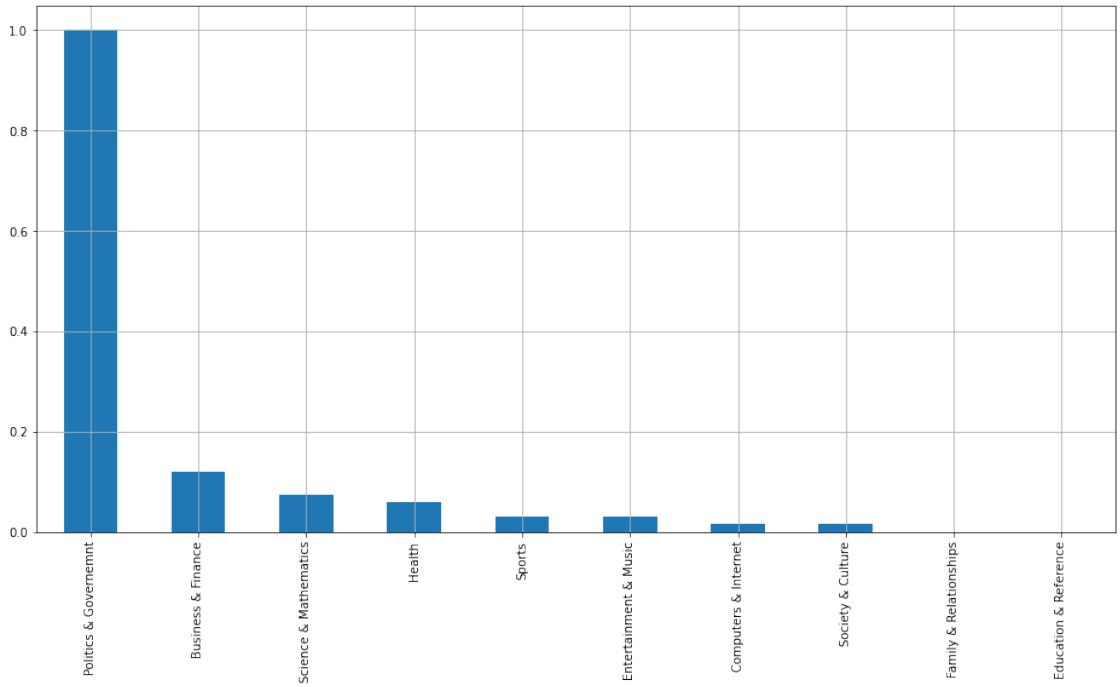
Di seguito analizzerò quali sono i principali topic discussi nel subreddit

[20]:
```
r = newsAnalysis['Topic'].value_counts();
r_norm = (r-r.min())/(r.max()-r.min())
```

[21]:
```
plt.figure(figsize=(8,8))

r_norm.plot.pie()

plt.show()
```



[22]:
```
plt.figure(figsize=(16,8))
r_norm.plot.bar()
plt.grid()
plt.show()
```



4.2.1 Osservazioni sui dati

Da questa analisi riguardo i topic è evidente che nel subreddit **"News"** viene principalmente parlato di **Politica**, il che pensandoci era molto prevedibile.

A seguire, anche se molto più piccole a paragone, viene discusso prevelentemente di **economia** e **scienza**.

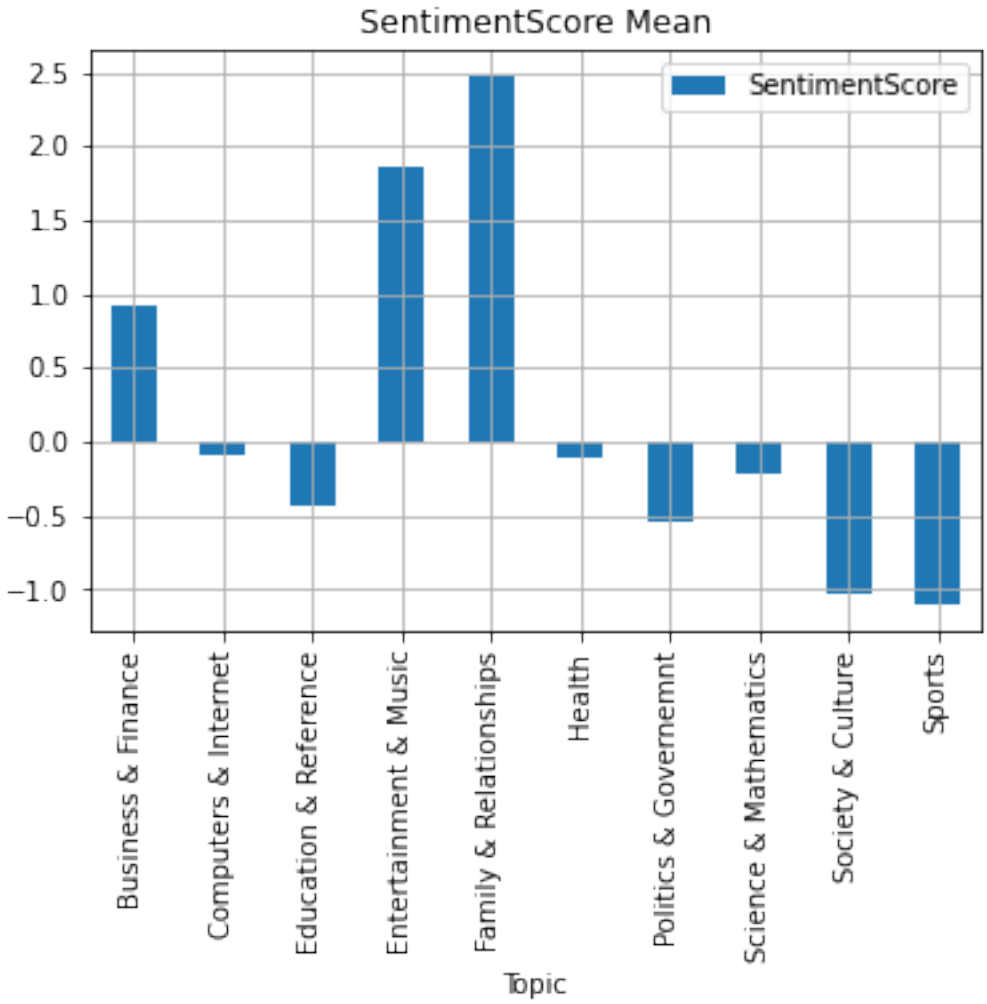
4.3 Analisi sentiment dei singoli topic

[23]:
```
r_sentiment = newsAnalysis.groupby("Topic").sum().drop('Up', axis=1)

for i in range(len(r)):
    r_sentiment.loc[r.index[i], "SentimentScore"] /= r[i]

print(r_sentiment)
```

```
                        SentimentScore
Topic
Business & Finance            0.923920
Computers & Internet         -0.090138
Education & Reference        -0.432292
Entertainment & Music         1.858706
Family & Relationships        2.473049
Health                       -0.109155
Politics & Governemnt        -0.545532
```

```
Science & Mathematics        -0.209003
Society & Culture            -1.023874
Sports                       -1.094345
```

[24]:
```
r_sentiment.plot.bar()
plt.title('SentimentScore Mean')
plt.grid()
plt.show()
```



4.3.1 Analisi sui dati

Osservando i dati ottenuti è facile convincerci che argomenti come **"Intrattenimento e musica"** e **"Famiglia e relazioni"** abbiano un sentiment più neutrale/positivo, mentre il resto tende ad essere molto negativo.

Per molti argomenti non possiamo fare una osservazione generale precisa per via della "scarsa" quantità di numeri (questo si traduce nel fatto che magari solo gli ultimi post siano magari negativi,

ma non vale per la totalità).

Diversamente si può invece dire per la **"Politica e governo"** dato che il numero di campioni è **molto più grande**, il che ci può fornire un valore un po' più accurato di **Sentiment score mean**, che, osservando il plot, è chiaramente **negativo**.

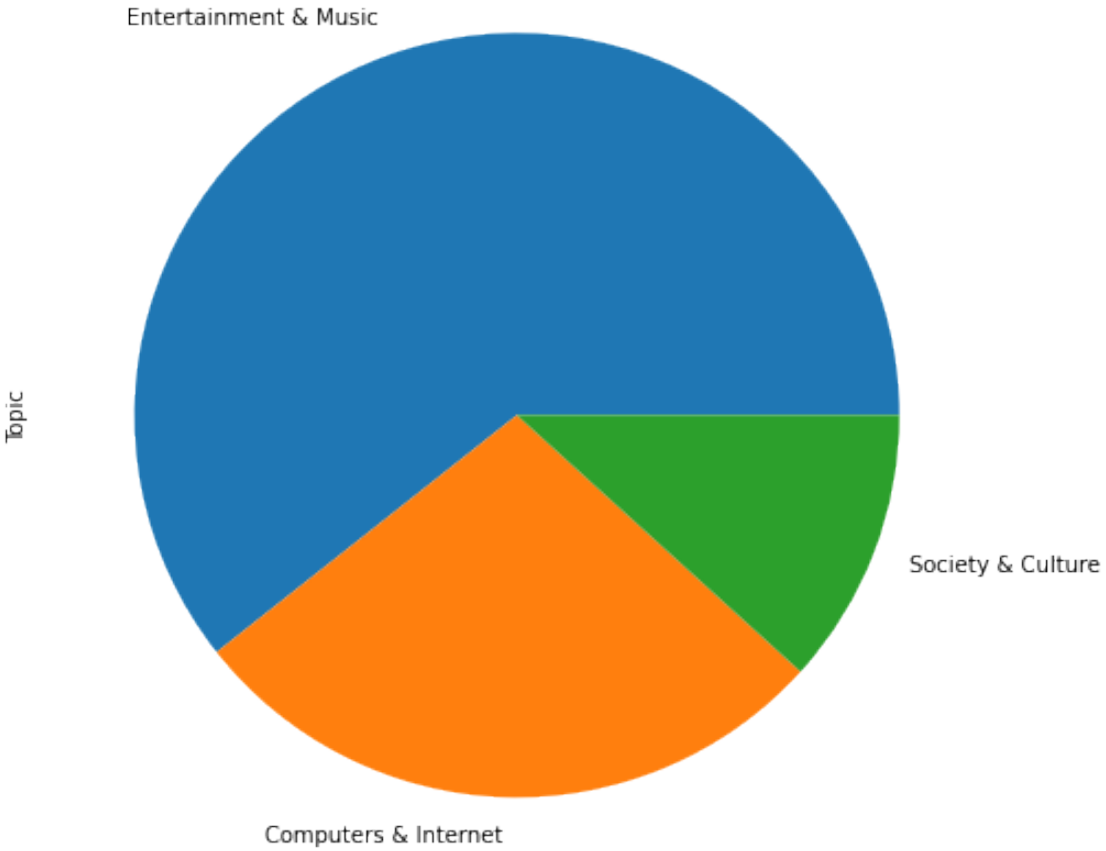
Questo è un dato che potevamo anch'esso aspettarci dato che di solito vengono discussi argomenti come "guerra, virus, corruzioni" e quant'altro nel topic di "Politiva e Governo".

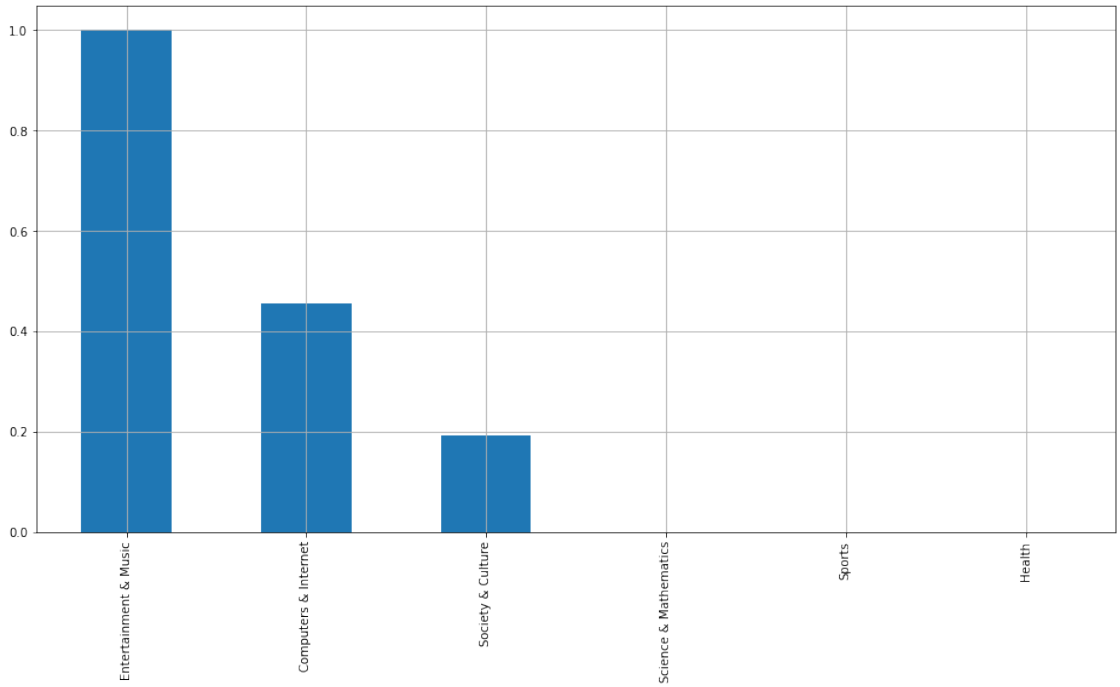
4.4 Definizione funzione per il deep analysis

[25]:
```
def RedditDeepAnalysis(subreddit):
    newsAnalysis = RedditAnalyzer(subreddit, 100)

    r = newsAnalysis['Topic'].value_counts();
    r_norm = (r-r.min())/(r.max()-r.min())

    plt.figure(figsize=(8,8))

    r_norm.plot.pie()

    plt.show()

    plt.figure(figsize=(16,8))
    r_norm.plot.bar()
    plt.grid()
    plt.show()

    r_sentiment = newsAnalysis.groupby("Topic").sum().drop('Up', axis=1)

    for i in range(len(r)):
        r_sentiment.loc[r.index[i], "SentimentScore"] /= r[i]

    print(r_sentiment)

    r_sentiment.plot.bar()
    plt.title('SentimentScore Mean')
    plt.grid()
    plt.show()
```

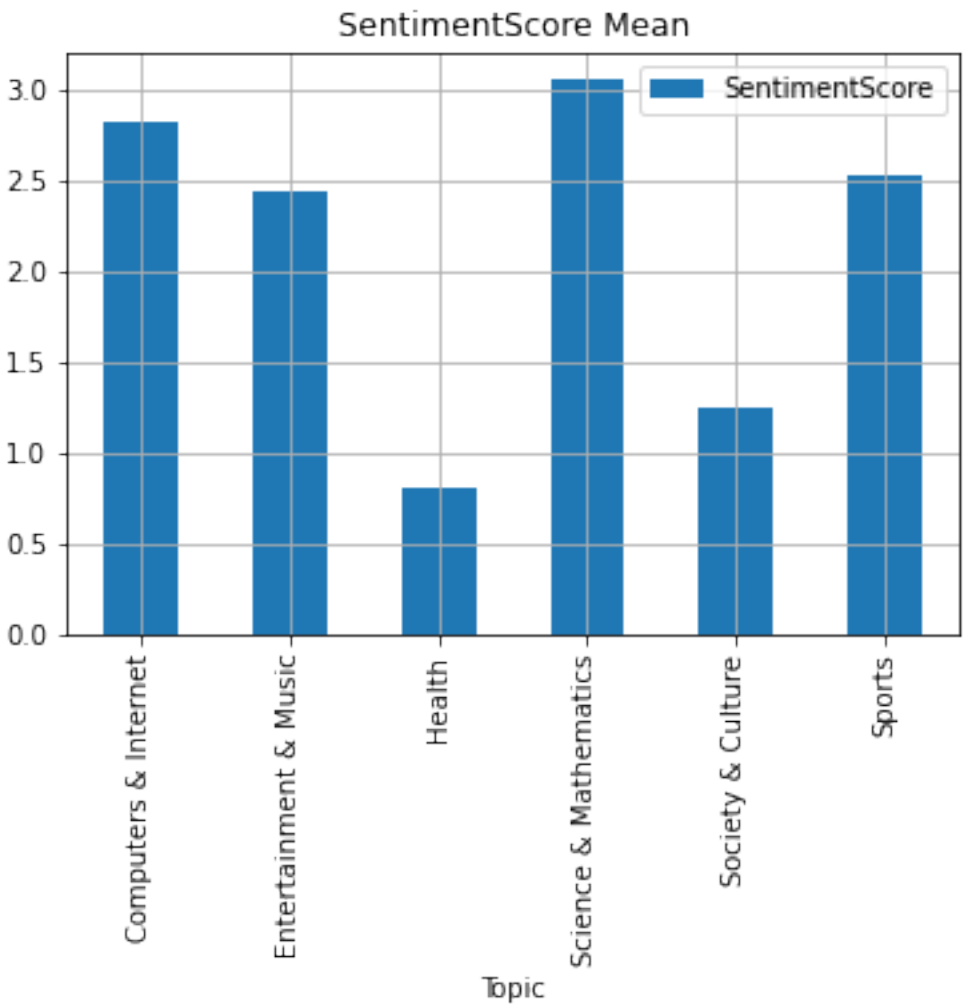
4.5 Altro esempio di analisi approfondita

[26]:
```
RedditDeepAnalysis("funny")
```


Entertainment & Music
Topic
Society & Culture
Computers & Internet



Topic	SentimentScore
Computers & Internet	2.830656
Entertainment & Music	2.443191
Health	0.807631
Science & Mathematics	3.056891
Society & Culture	1.246377
Sports	2.529071



4.5.1 Analisi dei dati

Come possiamo vedere dai dati ottenuti sopra il subreddit **"Funny"** tende essenzialmente ad essere positivo e discute principalmente di **"intrattenimento e musica"** (come potevamo immaginare).

Da un subreddit del genere potremmo anche pensare di ottenere dei sentiment score solo positivi, tuttavia la presenza di score tendenti al neutro o addirittura negativo (per esempio riguardo "Health") può essere dovuto al fatto che nel subreddit ci sia anche una tendenza verso il **"Black humor"** che potrebbe essere valutato negativamente dal regressore allenato per il **sentiment analysis.**

5 Conclusioni

Nonostante i modelli allenati non abbiano ottenuto degli score molto alti riescono a fornire delle stime abbastanza precise.

Inoltre i dati che si riescono ad ottenere tramite quest'ultime riescono a dare delle panoramiche abbastanza interessanti riguardo **Reddit** che permettono un'ampia analisi di quest'ultimo, sia come negli esempi forniti sia in altri possibili modi.