In [33]:
```
# Create a classification model using Jeopardy questions to predict categories for each
# Jeopardy question.
```

In [34]:
```
# Import required libraries.

import pandas as pd
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.svm import LinearSVC
```

In [35]:
```
# Create a dataframe. Read the Jeopardy questions from the jeopardy_data.csv file
# downloaded from https://www.kaggle.com/tunguz/200000-jeopardy-questions.

df = pd.read_csv("data/jeopardy_data.csv")
```

In [36]:
```
print(df.head(15))
```

```
   Category                                           Question  \
0   HISTORY  For the last 8 years of his life, Galileo was ...
1   HISTORY  Built in 312 B.C. to link Rome & the South of ...
2   HISTORY  In 1000 Rajaraja I of the Cholas battled to ta...
3   HISTORY  Karl led the first of these Marxist organizati...
4   HISTORY  This Asian political party was founded in 1885...
5   SCIENCE     99.95% of the mass of an atom is in this part
6   SCIENCE  During this plant process, carbon dioxide & wa...
7   SCIENCE  The wedge is an adaptation of the simple machi...
8   SCIENCE  Of the 6 noble gases on the periodic table, it...
9   SCIENCE  Lava & igneous rock are formed from this hot l...
10  HISTORY  After a 15-year stay in England, this propriet...
11  HISTORY  This young man put his savings into a small Cl...
12  HISTORY  First Lady Helen Taft led a fund-raising drive...
13  HISTORY  This Chiricahua Apache was a popular attractio...
14  HISTORY  In 1861 this onetime VP compiled "A Manual of ...

                Answer
0           Copernicus
1        the Appian Way
2   Ceylon (or Sri Lanka)
3       the International
4      the Congress Party
5              the nucleus
6          photosynthesis
7                   plane
8                  helium
9                   magma
10          William Penn
11    John D. Rockefeller
12  Sinking of the Titanic
13                Geronimo
14        Thomas Jefferson
```

In [37]:
```
# Use a support vector classifier (SVC) to vectorize the Jeopardy questions.

tfidf = TfidfVectorizer(max_features=20000, ngram_range=(1, 5), analyzer="char")
X = tfidf.fit_transform(df["Question"])
y = df["Category"]
```

In [38]:
```
# Train the data.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [39]:
```
print(X_train)
```

```
  (0, 13656)    0.08438716457823926
  (0, 10707)    0.07796285446760415
  (0, 17265)    0.07796285446760415
  (0, 10706)    0.07584506894980497
  (0, 19404)    0.08330287012787632
  (0, 8726)     0.08496869552426053
  (0, 13306)    0.08496869552426053
  (0, 8725)     0.07796285446760415
  (0, 17836)    0.07612552942072656
  (0, 7283)     0.07227838687983666
  (0, 8223)     0.07864412674941086
  (0, 15832)    0.08095827618251102
  (0, 2008)     0.08013594876247608
  (0, 8222)     0.07828862371550851
  (0, 15831)    0.07936671283366034
  (0, 8221)     0.07670706036674783
  (0, 13221)    0.08230904305734801
  (0, 9727)     0.06717474163727151
  (0, 12276)    0.07670706036674783
  (0, 12275)    0.07557105642065348
  (0, 14296)    0.08438716457823926
  (0, 6303)     0.08279560122938859
  (0, 12658)    0.08330287012787632
  (0, 14295)    0.08279560122938859
  (0, 12657)    0.07864412674941086
  :     :
  (7340, 16468) 0.037076095790076 71
  (7340, 17697) 0.026150056046636362
  (7340, 17479) 0.030154406934462117
  (7340, 7000)  0.038485690637463 54
  (7340, 9635)  0.021447481119101625
  (7340, 18141) 0.018499611911073636
  (7340, 2244)  0.018976978153749945
  (7340, 14527) 0.028788285587852 65
  (7340, 2793)  0.039970741956978 16
  (7340, 12222) 0.041293019757973 86
  (7340, 6456)  0.038701549560793 56
  (7340, 12709) 0.052673635718919312
  (7340, 18717) 0.029000094554025878
  (7340, 19370) 0.09572551557391816
  (7340, 10119) 0.034940177434709 18
  (7340, 16467) 0.052410266152063 77
  (7340, 4242)  0.034841330556490 45
  (7340, 11485) 0.055880892917237792
  (7340, 6999)  0.069304065422723632
  (7340, 9457)  0.017619214350595373
  (7340, 17696) 0.034833738667648 83
  (7340, 0)     0.12087315629595828
  (7340, 15348) 0.017580853940412394
  (7340, 13778) 0.18516375362925838
  (7340, 8775)  0.020893128808180866
```

In [40]:
```
print(y_train)
```

```
2784     SCIENCE
713      HISTORY
1235     SCIENCE
6088    LITERACY
5541    LITERACY
          ...
4373     HISTORY
7891     SCIENCE
4859     HISTORY
3264     SCIENCE
2732    LITERACY
Name: Category, Length: 7341, dtype: object
```

In [41]:
```
print(X_test)
```

```
  (0, 7285)     0.10509401645292553
  (0, 7284)     0.09719876874106466
  (0, 19374)    0.10231936133489786
  (0, 13234)    0.10509401645292553
  (0, 12714)    0.09177072185470411
  (0, 7283)     0.09228665159729793
  (0, 9727)     0.08577020386362265
  (0, 6698)     0.09756549660361982
  (0, 14063)    0.10703944476514358
  (0, 8862)     0.10509401645292553
  (0, 18600)    0.10231936133489786
  (0, 656)      0.10449713721167139
  (0, 15939)    0.09794128038424417
  (0, 5809)     0.09997342385909233
  (0, 15938)    0.08384459866229037
  (0, 4960)     0.09454537697273178
  (0, 10192)    0.10133718379575656
  (0, 19060)    0.08461798578802278
  (0, 13440)    0.08647002692378718
  (0, 1712)     0.09516662526622165
  (0, 10539)    0.09177072185470411
  (0, 13439)    0.08647002692378718
  (0, 1711)     0.09454537697273178
  (0, 12713)    0.08920891593871044
  (0, 13438)    0.08541209172613206
  :     :
  (1835, 5258)  0.042031561801053836
  (1835, 1469)  0.024715831577101617
  (1835, 9635)  0.027505726654290855
  (1835, 18141) 0.011862588056437372
  (1835, 2244)  0.024337383451724 78
  (1835, 12222) 0.026478509012076186
  (1835, 14929) 0.029695012793258 3
  (1835, 6456)  0.026041914959212907
  (1835, 12709) 0.12384587839988966
  (1835, 18717) 0.013401660170190481
  (1835, 19370) 0.046036872146372232
  (1835, 2890)  0.038439047230242 26
  (1835, 10119) 0.007214534043905099
  (1835, 19691) 0.030705454838270 17
  (1835, 3945)  0.029462889276209908
  (1835, 16467) 0.078416956383575 93
  (1835, 4242)  0.089365842635849 37
  (1835, 11485) 0.011928868919434317
  (1835, 6999)  0.099990343639136 86
  (1835, 9457)  0.033891448491793143
  (1835, 17696) 0.055841481197852094
  (1835, 0)     0.21037893649413158
  (1835, 15348) 0.056367244074635 95
  (1835, 13778) 0.056195516144216
  (1835, 8775)  0.040192180482420195
```

In [42]:
```
print(y_test)
```

```
6840    LITERACY
8527    LITERACY
3895     HISTORY
7628     HISTORY
3601     HISTORY
          ...
655      SCIENCE
3844     SCIENCE
445      HISTORY
5685    LITERACY
660      HISTORY
Name: Category, Length: 1836, dtype: object
```

In [43]:
```
# Fit the data.

clf = LinearSVC(C=20, class_weight="balanced", max_iter=20000)
clf.fit(X_train, y_train)
```

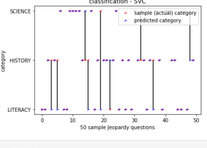
Out[43]:
```
LinearSVC(C=20, class_weight='balanced', max_iter=20000)
```

In [44]:
```
# Display a classification report of the model. Show the accuracy of being able to predict
# a category.

y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

     HISTORY       0.84      0.87      0.85       727
   LITERACY        0.87      0.85      0.86       635
     SCIENCE       0.83      0.81      0.82       474

    accuracy                           0.85      1836
   macro avg       0.85      0.84      0.84      1836
weighted avg       0.85      0.85      0.85      1836
```

In [45]:
```
# Plot the model. Show 50 samples.

# The red star is a sample category. It represents the ACTUAL category. The blue star
# is the predicted category. When a red and blue star overlap it means the prediction
# is correct. If they don't overlap it means the prediction is wrong and a line is drawn
# connecting the stars to visually indicate an incorrect prediction. There are thousands
# of questions, but only 50 are plotted because it is more visually pleasing and easy
# to read.

fig = plt.figure()
ax = plt.axes()

y_test = y_test.tolist()
y_pred = y_pred.tolist()

ax.plot(y_test[0:50], "3", color="red", label="sample (actual) category")
ax.plot(y_pred[0:50], "4", color="blue", label="predicted category")

for i in range(50):  # len(y_test)
    ax.vlines(x=i, ymin=y_test[i], ymax=y_pred[i], color="black")

ax.legend(numpoints=1)

ax.set_title("classification - SVC")
ax.set_xlabel("50 sample Jeopardy questions")
ax.set_ylabel("category")
```

Out[45]:
```
Text(0, 0.5, 'category')
```



In [31]:
```
# ANALYSIS
#
#
# The classification model was able predict the categories of Jeopardy questions that can
# be categorized as historical, literature, and scientific at an overall 85% accuracy.
# Science was the hardest category to predict by about 1%-4%. The reason could be because
# there could be literature-based and historicial questions ABOUT scientific events that
# may have occured. There are many overlaps with the language and word-patterns being used,
# but more of the science-based questions may fall within the literature and history
# categories as well.
```