In [1]:
```
import os
import numpy as np
from string import punctuation
from os import listdir
from collections import Counter
from nltk.corpus import stopwords
import nltk
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer  # Adjusted impor
from tensorflow.keras.preprocessing.sequence import pad_sequences  # Adjust
from tensorflow.keras.models import Sequential  # Adjusted import for Keras
from tensorflow.keras.layers import Dense, Dropout  # Adjusted import for K
import pandas as pd
from matplotlib import pyplot as plt
nltk.download('stopwords')
tokenizer=Tokenizer()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sd616\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
def load_doc(filename):
    file = open(filename, 'r')
    text = file.read()
    file.close()
    return text
```

In [3]:
```
def clean_doc(doc, vocab):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()] #to remove tokens
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    return tokens

def add_doc_to_vocab(filename, vocab):
    doc = load_doc(filename)
    tokens = clean_doc(doc, vocab)
    vocab.update(tokens)

def doc_to_line(filename, vocab):
    doc = load_doc(filename)
    tokens = clean_doc(doc, vocab)
    tokens = [w for w in tokens if w in vocab]
    return ' '.join(tokens)

def process_docs(directory, vocab, is_train):
    lines = list()
    movie_reviews_path = nltk.data.find('corpora/movie_reviews').path
    directory_path = os.path.join(movie_reviews_path, directory)
    for filename in listdir(directory_path):
        if is_train and filename.startswith('cv9'):
            continue
        if not is_train and not filename.startswith('cv9'):
            continue
        path = os.path.join(directory_path, filename)
        line = doc_to_line(path, vocab)
        lines.append(line)
    return lines

def process_docsl(directory,vocab):
    movie_reviews_path = nltk.data.find('corpora/movie_reviews').path
    directory_path = os.path.join(movie_reviews_path, directory)
    for filename in listdir(directory_path):
        if filename.startswith('cv9'):
            continue
        path = os.path.join(directory_path, filename)
        add_doc_to_vocab(path, vocab)
```

In [4]:
```
from nltk.corpus import movie_reviews
nltk.download('movie_reviews')
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\sd616\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\movie_reviews.zip.
```

Out[4]: True

In [5]:
```
positive_ids = movie_reviews.fileids('pos')
negavtive_ids = movie_reviews.fileids('neg')
print(positive_ids)
print(negavtive_ids)
```

```
['pos/cv000_29590.txt', 'pos/cv001_18431.txt', 'pos/cv002_15918.txt',
'pos/cv003_11664.txt', 'pos/cv004_11636.txt', 'pos/cv005_29443.txt', 'p
os/cv006_15448.txt', 'pos/cv007_4968.txt', 'pos/cv008_29435.txt', 'pos/
cv009_29592.txt', 'pos/cv010_29198.txt', 'pos/cv011_12166.txt', 'pos/cv
012_29576.txt', 'pos/cv013_10159.txt', 'pos/cv014_13924.txt', 'pos/cv01
5_29439.txt', 'pos/cv016_4659.txt', 'pos/cv017_22464.txt', 'pos/cv018_2
0137.txt', 'pos/cv019_14482.txt', 'pos/cv020_8825.txt', 'pos/cv021_1583
8.txt', 'pos/cv022_12864.txt', 'pos/cv023_12672.txt', 'pos/cv024_6778.t
xt', 'pos/cv025_3108.txt', 'pos/cv026_29325.txt', 'pos/cv027_25219.tx
t', 'pos/cv028_26746.txt', 'pos/cv029_18643.txt', 'pos/cv030_21593.tx
t', 'pos/cv031_18452.txt', 'pos/cv032_22550.txt', 'pos/cv033_24444.tx
t', 'pos/cv034_29647.txt', 'pos/cv035_3954.txt', 'pos/cv036_16831.txt',
'pos/cv037_18510.txt', 'pos/cv038_9749.txt', 'pos/cv039_6170.txt', 'po
s/cv040_8276.txt', 'pos/cv041_21113.txt', 'pos/cv042_10982.txt', 'pos/c
v043_15013.txt', 'pos/cv044_16969.txt', 'pos/cv045_23923.txt', 'pos/cv0
46_10188.txt', 'pos/cv047_1754.txt', 'pos/cv048_16828.txt', 'pos/cv049_
20471.txt', 'pos/cv050_11175.txt', 'pos/cv051_10306.txt', 'pos/cv052_29
378.txt', 'pos/cv053_21822.txt', 'pos/cv054_4230.txt', 'pos/cv055_8338.
txt', 'pos/cv056_13133.txt', 'pos/cv057_7453.txt', 'pos/cv058_8025.tx
```

In [6]:
```
vocab = Counter()
process_docsl('neg', vocab)
process_docsl('pos', vocab)
print(len(vocab))
print(vocab.most_common(50))
```

```
44276
[('film', 7983), ('one', 4946), ('movie', 4826), ('like', 3201), ('even',
2262), ('good', 2080), ('time', 2041), ('story', 1907), ('films', 1873),
('would', 1844), ('much', 1824), ('also', 1757), ('characters', 1735), ('g
et', 1724), ('character', 1703), ('two', 1643), ('first', 1588), ('see', 1
557), ('way', 1515), ('well', 1511), ('make', 1418), ('really', 1407), ('l
ittle', 1351), ('life', 1334), ('plot', 1288), ('people', 1269), ('bad', 1
248), ('could', 1248), ('scene', 1241), ('movies', 1238), ('never', 1201),
('best', 1179), ('new', 1140), ('scenes', 1135), ('man', 1131), ('many', 1
130), ('doesnt', 1118), ('know', 1092), ('dont', 1086), ('hes', 1024), ('g
reat', 1014), ('another', 992), ('action', 985), ('love', 977), ('us', 96
7), ('go', 952), ('director', 948), ('end', 946), ('something', 945), ('st
ill', 936)]
```

In [7]:
```
best_model = None
history=None
def evaluate_mode(Xtrain, ytrain, Xtest, ytest):
    scores = list()
    n_repeats = 1
    n_words = Xtest.shape[1]
    best_acc = 0

    for i in range(n_repeats):
        model = Sequential()
        model.add(Dense(50, input_shape=(n_words,), activation='relu'))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(loss='binary_crossentropy', optimizer='adam', metrics

        history_temp=model.fit(Xtrain, ytrain, epochs=50, verbose=2)

        loss, acc = model.evaluate(Xtest, ytest, verbose=0)
        scores.append(acc)
        global history
        global best_model
        if acc > best_acc:
            best_acc = acc
            best_model = model  # Keep track of the best model
            history=history_temp

    return scores


def prepare_data(train_docs, test_docs, mode):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(train_docs)  # Build the word index on the train

    # Convert the texts to matrix representation based on the specified mod
    Xtrain = tokenizer.texts_to_matrix(train_docs, mode=mode)
    Xtest = tokenizer.texts_to_matrix(test_docs, mode=mode)

    # Convert to NumPy arrays (Keras works directly with NumPy)
    Xtrain = np.array(Xtrain, dtype=np.float32)
    Xtest = np.array(Xtest, dtype=np.float32)

    return Xtrain, Xtest
```

In [8]:
```
min_occurance=2
tokens=[k for k,c in vocab.items() if c >= min_occurance]
print(len(tokens))
```

```
25767
```

In [9]:
```
def save_list(lines, filename):
    data = '\n'.join(lines)
    file = open(filename, 'w')
    file.write(data)
    file.close()

save_list(tokens, 'vocab.txt')
```

In [10]:
```
vocab_filename = 'vocab.txt'
vocab = load_doc(vocab_filename)
vocab = vocab.split()
vocab = set(vocab)
print(len(vocab))
print(vocab)
```

```
25767
{'ripoff', 'miniature', 'humbly', 'dalmatian', 'observed', 'queasy', 's
evere', 'skg', 'impregnating', 'rohmer', 'sugary', 'kumble', 'review',
'plotless', 'cracked', 'celebrities', 'wheelchairbound', 'jamileh', 'ov
erbearing', 'hank', 'mimieux', 'obvious', 'navaz', 'mant', 'capitalis
t', 'angular', 'sausages', 'ferocity', 'pimple', 'panama', 'unravelin
g', 'continue', 'northern', 'iceberg', 'hesitation', 'robust', 'travel
s', 'soak', 'protagonists', 'mountain', 'commonly', 'opulent', 'anxious
ly', 'girl', 'tricks', 'momentum', 'hamming', 'pitch', 'tooth', 'gave',
'silverstone', 'vincent', 'mutual', 'guard', 'brody', 'react', 'sells',
'loudly', 'illegal', 'collectible', 'morsels', 'plants', 'theft', 'ger
e', 'excitement', 'dignity', 'ndorff', 'sequencesthey', 'earthly', 'rif
fraff', 'divinity', 'getting', 'rigg', 'freddy', 'shoddy', 'disappointe
d', 'chester', 'deltas', 'extracurricular', 'pompous', 'creed', 'haye
k', 'daleks', 'divisions', 'reviewers', 'outdid', 'shepards', 'tricky',
'dougnac', 'psychedelic', 'exaggeration', 'crooked', 'martin', 'mosco
w', 'amaze', 'sphere', 'understudy', 'richardsons', 'beautiful', 'astro
naut', 'spectacularly', 'intrigued', 'pixars', 'mechanical', 'homemad
e', 'atlantic', 'ovitz', 'mano', 'pictures', 'footprints', 'untamed',
```

In [11]:
```
positive_lines = process_docs('pos', vocab, True)
negative_lines = process_docs('neg', vocab, True)
test_positive_lines = process_docs('pos', vocab, False)
test_negative_lines = process_docs('neg', vocab, False)
```

In [12]:
```
train_docs = negative_lines + positive_lines
test_docs = test_negative_lines + test_positive_lines
```

In [13]:
```
ytrain=np.array([0 for _ in range(900)] + [1 for _ in range(900)])
ytest=np.array([0 for _ in range(100)] + [1 for _ in range(100)])
result=pd.DataFrame()
```

In [14]:
```
modes = ['binary']
# modes = ['binary', 'count', 'tfidf', 'freq']


for mode in modes:
    Xtrain, Xtest = prepare_data(train_docs, test_docs, mode)
    score = evaluate_mode(Xtrain, ytrain, Xtest, ytest)
    result[mode] = score

    print('Mode:', mode)
    print('Accuracy: %.3f (%.3f)' % (np.mean(score), np.std(score)))
    print()

print(result)
result.boxplot()
plt.show()
```

```
C:\Users\sd616\anaconda\lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Epoch 1/50
57/57 - 2s - 37ms/step - accuracy: 0.7539 - loss: 0.4789
Epoch 2/50
57/57 - 1s - 19ms/step - accuracy: 0.9928 - loss: 0.0580
Epoch 3/50
57/57 - 1s - 19ms/step - accuracy: 1.0000 - loss: 0.0151
Epoch 4/50
57/57 - 1s - 19ms/step - accuracy: 1.0000 - loss: 0.0066
Epoch 5/50
57/57 - 1s - 18ms/step - accuracy: 1.0000 - loss: 0.0039
Epoch 6/50
57/57 - 1s - 18ms/step - accuracy: 1.0000 - loss: 0.0026
Epoch 7/50
57/57 - 1s - 18ms/step - accuracy: 1.0000 - loss: 0.0017
```

In [ ]:
```
```