Assignment 10

10.1

10.1.a

In [1]:

```
def tokenize(sentence):

    # Remove all unwanted characters
    remove = '''!@#$%^&*()~`-=_+{}[]:";'<>?,./\|'"'''

    for char in remove:
        sentence = sentence.replace(char,'')

    normalized_sentence = sentence.lower()

    # split the sentence into words
    words = normalized_sentence.split()

    return words

# test the function
sentence = 'Making this code work is not that difficult.'
tokens = tokenize(sentence)
print(tokens)
```

```
['making', 'this', 'code', 'work', 'is', 'not', 'that', 'difficult']
```

10.1.b

In [2]:

```
def ngram(tokens, n):
    ngrams = []
    # Create ngrams
    return ngrams
```

In [3]:

```
def ngram(tokens, n):
    ngrams = zip(*[tokens[i:] for i in range(n)])
    return [" ".join(ngram) for ngram in ngrams]

n = 4
sentence = 'Making this code work is not that difficult.'
ngram(tokens, n)
```

Out[3]:

```
['making this code work',
 'this code work is',
 'code work is not',
 'work is not that',
 'is not that difficult']
```

10.1.c

In [4]:

```
# Import library
import numpy as np
```

```
def one_hot_encode(tokens, num_words):
    token_index = {}
    for sample in samples:
        for word in sample.split():
            if word not in token_index:
                token_index[word] = len(token_index) + 1
    results = np.zeros(shape = (len(samples), num_words, max(token_index.values()) + 1))
    for i, sample in enumerate(samples):
        for j, word in list(enumerate(sample.split()))[ :num_words]:
            index = token_index.get(word)
            results[i, j, index] = 1
    return results


num_words = 20
samples = ['This is a fun exercise. I think I\'ll use it in the future.']
one_hot_encode(tokens, num_words)
```

Out[4]:

```
array([[[0., 1., 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., 0., 0., 1., 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., 0., 0., 1., 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., 0., 0., 1., 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., 0., 0., 1., 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., 0., 0., 1., 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., 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., 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., 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., 0., 0., 0., 0., 0., 0.]]])
```

10.2

In [6]:

```
# import libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import os
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt

current_dir = Path(os.getcwd()).absolute()
imdb_dir = Path('C:\\Users\\Nick\\Documents\\School\\Big Data\\dsc650\\data\\external\\imdb\\aclImdb')
test_dir = os.path.join(imdb_dir, 'test')
train_dir = os.path.join(imdb_dir, 'train')
```

In [9]:

```
# 6.8
labels = []
texts = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
```

```
    for fname in os.listdir(dir_name):
        try:
            if fname[-4:] == '.txt':
                f = open(os.path.join(dir_name, fname))
                texts.append(f.read())
                f.close()
                if label_type == 'neg':
                    labels.append(0)
                else:
                    labels.append(1)
        except:
            pass
```

In [11]:

```
1 in labels
```

Out[11]:

```
True
```

In [12]:

```
# 6.9 Tokenizing
maxlen = 100
training_samples = 200
validation_samples = 10000
max_words = 10000

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

data = pad_sequences(sequences, maxlen=maxlen)

labels = np.asarray(labels)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)

indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]

x_train = data[:training_samples]
y_train = labels[:training_samples]
x_val = data[training_samples: training_samples + validation_samples]
y_val = labels[training_samples: training_samples + validation_samples]
```

```
Found 88413 unique tokens.
Shape of data tensor: (24984, 100)
Shape of label tensor: (24984,)
```

In [16]:

```
# 6.10
embeddings_index = {}
f = open('C:\\Users\\Nick\\Documents\\School\\Big Data\\dsc650\\data\\glove.6B.100d.txt'
, encoding="UTF-8")
for line in f:
    #print(line)
    try:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
    except:
        pass
f.close()
```

```
print('Found %s word vectors.' % len(embeddings_index))
```

```
Found 400000 word vectors.
```

In [17]:

```
# 6.11
embedding_dim = 100

embedding_matrix = np.zeros((max_words, embedding_dim))
for word, i in word_index.items():
    if i < max_words:
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
```

In [18]:

```
# 6.12
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 100)          1000000
_________________________________________________________________
flatten (Flatten)            (None, 10000)             0
_________________________________________________________________
dense (Dense)                (None, 32)                320032
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,320,065
Trainable params: 1,320,065
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
# 6.13
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = False
```

In [20]:

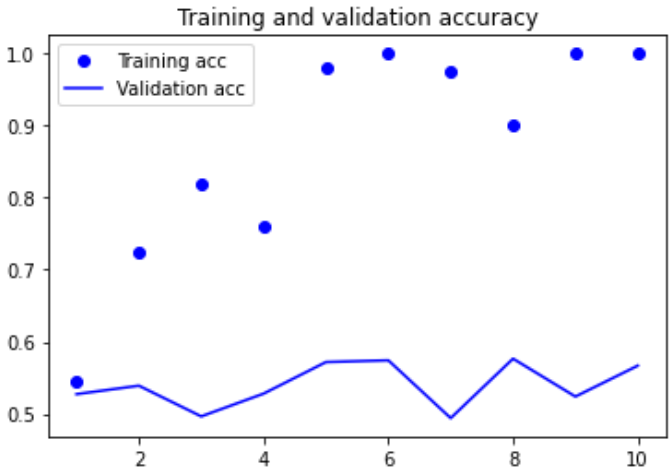
```
# 6.14
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_data=(x_val, y_val))
model.save_weights('pre_trained_glove_model.h5')
```

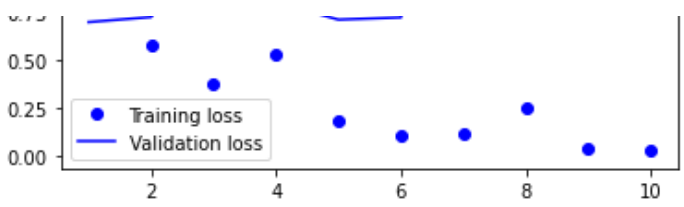
```
Train on 200 samples, validate on 10000 samples
Epoch 1/10
200/200 [==============================] - 3s 14ms/sample - loss: 1.8473 - acc: 0.5450 -
val_loss: 0.6967 - val_acc: 0.5273
Epoch 2/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.5788 - acc: 0.7250 - v
al_loss: 0.7231 - val_acc: 0.5390
Epoch 3/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.3761 - acc: 0.8200 - v
al loss: 1.1298 - val acc: 0.4964
```

```
Epoch 4/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.5306 - acc: 0.7600 - v
al_loss: 0.8000 - val_acc: 0.5280
Epoch 5/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.1862 - acc: 0.9800 - v
al_loss: 0.7096 - val_acc: 0.5718
Epoch 6/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.1113 - acc: 1.0000 - v
al_loss: 0.7212 - val_acc: 0.5742
Epoch 7/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.1207 - acc: 0.9750 - v
al_loss: 1.9236 - val_acc: 0.4940
Epoch 8/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.2540 - acc: 0.9000 - v
al_loss: 0.7382 - val_acc: 0.5765
Epoch 9/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.0460 - acc: 1.0000 - v
al_loss: 1.0958 - val_acc: 0.5239
Epoch 10/10
200/200 [==============================] - 1s 6ms/sample - loss: 0.0329 - acc: 1.0000 - v
al_loss: 0.8100 - val_acc: 0.5668
```

In [21]:

```
# 6.15
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [22]:

```
# 6.16
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_data=(x_val, y_val))
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 100)          1000000
_________________________________________________________________
flatten_1 (Flatten)          (None, 10000)             0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                320032
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,320,065
Trainable params: 1,320,065
Non-trainable params: 0
_________________________________________________________________
Train on 200 samples, validate on 10000 samples
Epoch 1/10
200/200 [==============================] - 2s 12ms/sample - loss: 0.7005 - acc: 0.5100 -
val_loss: 0.6928 - val_acc: 0.5108
Epoch 2/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.4928 - acc: 0.9900 - v
al_loss: 0.7168 - val_acc: 0.5083
Epoch 3/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.2754 - acc: 0.9900 - v
al_loss: 0.7017 - val_acc: 0.5183
Epoch 4/10
200/200 [==============================] - 2s 8ms/sample - loss: 0.1195 - acc: 1.0000 - v
al_loss: 0.7348 - val_acc: 0.5135
Epoch 5/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.0589 - acc: 1.0000 - v
al_loss: 0.7289 - val_acc: 0.5220
Epoch 6/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.0298 - acc: 1.0000 - v
al_loss: 0.7455 - val_acc: 0.5198
Epoch 7/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.0163 - acc: 1.0000 - v
al_loss: 0.7260 - val_acc: 0.5306
Epoch 8/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.0094 - acc: 1.0000 - v
al_loss: 0.7458 - val_acc: 0.5277
Epoch 9/10
200/200 [==============================] - 1s 7ms/sample - loss: 0.0056 - acc: 1.0000 - v
al_loss: 0.7756 - val_acc: 0.5200
Epoch 10/10
200/200 [==============================] - 2s 8ms/sample - loss: 0.0034 - acc: 1.0000 - v
al_loss: 0.7964 - val_acc: 0.5205
```

In [24]:

```
# 6.17
labels = []
texts = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(test_dir, label_type)
    for fname in sorted(os.listdir(dir_name)):
        try:
            if fname[-4:] == '.txt':
                f = open(os.path.join(dir_name, fname))
                texts.append(f.read())
                f.close()
                if label_type == 'neg':
                    labels.append(0)
                else:
                    labels.append(1)
        except:
            pass

sequences = tokenizer.texts_to_sequences(texts)
x_test = pad_sequences(sequences, maxlen=maxlen)
y_test = np.asarray(labels)
```

In [25]:

```
1 in labels
```

Out[25]:

```
True
```

In [34]:

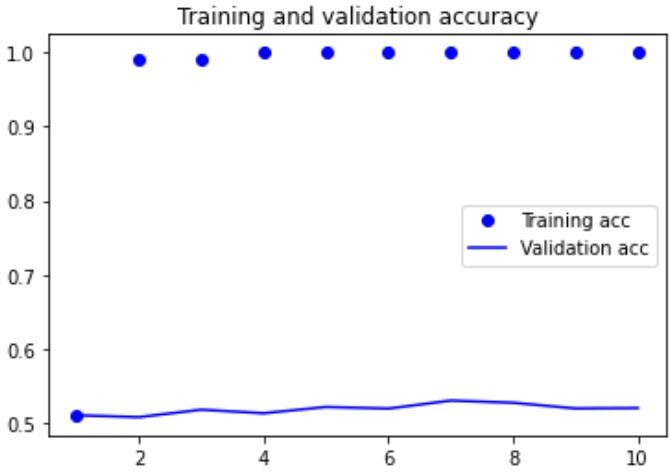
```
# 6.18
# !!! Doesn't work with my version of Keras (tensorflow.keras)
model.load_weights('./pre_trained_glove_model.h5')
model.evaluate(x_test, y_test)
```

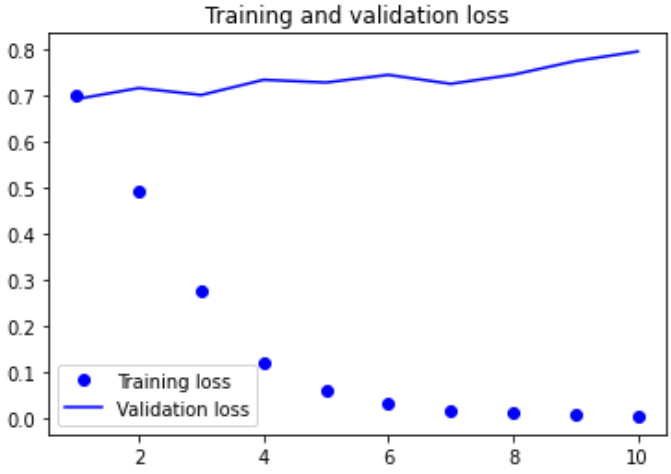
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-34-a482c7ba16a4> in <module>
      1 # 6.18
----> 2 model.load_weights('C:\\Users\\Nick\\Documents\\School\\Big Data\\dsc650-bigdata-main\\assignment10\\pre_trained_glove_model.h5')
      3 model.evaluate(x_test, y_test)

~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training.py in load_weights(self, filepath, by_name)
    179             raise ValueError('Load weights is not yet supported with TPUStrategy '
    180                              'with steps_per_run greater than 1.')
--> 181       return super(Model, self).load_weights(filepath, by_name)
    182
    183     @trackable.no_automatic_dependency_tracking

~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\network.py in load_weights(self, filepath, by_name)
   1175           saving.load_weights_from_hdf5_group_by_name(f, self.layers)
   1176         else:
-> 1177           saving.load_weights_from_hdf5_group(f, self.layers)
   1178
   1179     def _updated_config(self):

~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\saving\hdf5_format.py in load_weights_from_hdf5_group(f, layers)
    649     """
    650     if 'keras_version' in f.attrs:
--> 651       original_keras_version = f.attrs['keras_version'].decode('utf8')
    652     else:
    653       original_keras_version = '1'
```

```
AttributeError: 'str' object has no attribute 'decode'
```

In [35]:

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





10.3

In [36]:

```
# 6.27
# import libraries
from tensorflow.keras.layers import LSTM
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
```

In [37]:

```
# Listing 6.22
max_features = 10000
maxlen = 500
batch_size = 32

print('Loading data...')
(input_train, y_train), (input_test, y_test) = imdb.load_data(
     num_words=max_features)
print(len(input_train), 'train sequences')
print(len(input_test), 'test sequences')

print('Pad sequences (samples x time)')
input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
input_test = sequence.pad_sequences(input_test, maxlen=maxlen)
print('input_train shape:', input_train.shape)
print('input_test shape:', input_test.shape)
```

```
Loading data...
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.np
z
17465344/17464789 [==============================] - 2s 0us/step
```

```
C:\Users\Nick\Anaconda3\lib\site-packages\tensorflow_core\python\keras\datasets\imdb.py:1
29: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is
a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is depre
cated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\Nick\Anaconda3\lib\site-packages\tensorflow_core\python\keras\datasets\imdb.py:1
30: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is
a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is depre
cated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
```

```
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
input_train shape: (25000, 500)
input_test shape: (25000, 500)
```

In [38]:

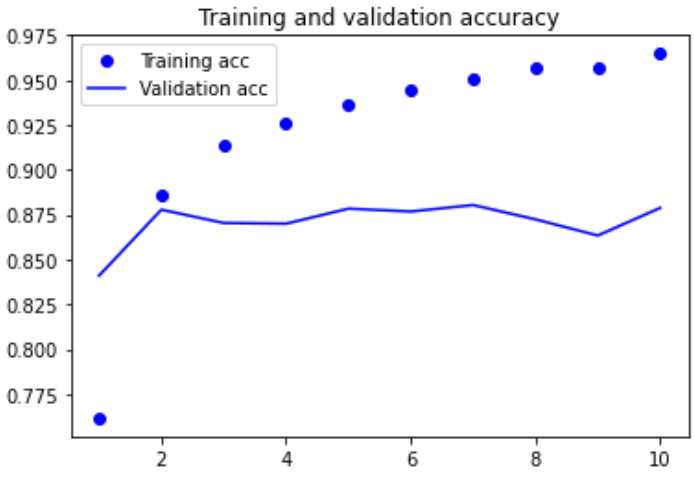
```
# 6.27
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

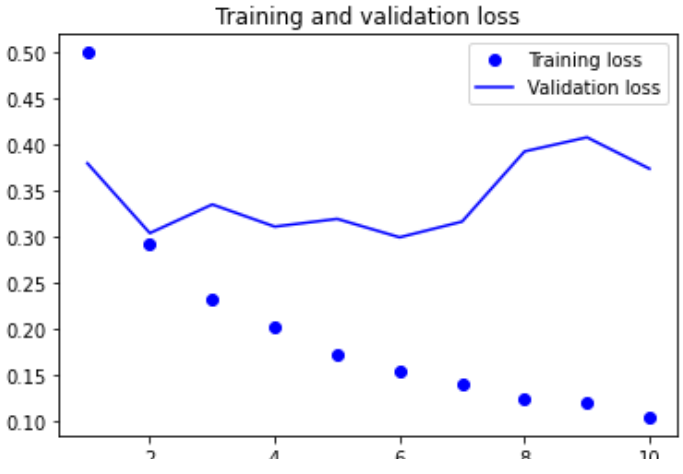
```
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 72s 4ms/sample - loss: 0.4999 - acc: 0.761
6 - val_loss: 0.3799 - val_acc: 0.8410
Epoch 2/10
20000/20000 [==============================] - 70s 4ms/sample - loss: 0.2934 - acc: 0.886
0 - val_loss: 0.3043 - val_acc: 0.8778
Epoch 3/10
20000/20000 [==============================] - 70s 4ms/sample - loss: 0.2320 - acc: 0.913
9 - val_loss: 0.3353 - val_acc: 0.8704
Epoch 4/10
20000/20000 [==============================] - 71s 4ms/sample - loss: 0.2028 - acc: 0.926
1 - val_loss: 0.3113 - val_acc: 0.8700
Epoch 5/10
20000/20000 [==============================] - 71s 4ms/sample - loss: 0.1725 - acc: 0.936
```

```
7 - val_loss: 0.3197 - val_acc: 0.8784
Epoch 6/10
20000/20000 [==============================] - 69s 3ms/sample - loss: 0.1549 - acc: 0.944
5 - val_loss: 0.2997 - val_acc: 0.8768
Epoch 7/10
20000/20000 [==============================] - 70s 3ms/sample - loss: 0.1402 - acc: 0.950
3 - val_loss: 0.3168 - val_acc: 0.8804
Epoch 8/10
20000/20000 [==============================] - 69s 3ms/sample - loss: 0.1256 - acc: 0.956
5 - val_loss: 0.3927 - val_acc: 0.8724
Epoch 9/10
20000/20000 [==============================] - 69s 3ms/sample - loss: 0.1210 - acc: 0.957
2 - val_loss: 0.4080 - val_acc: 0.8634
Epoch 10/10
20000/20000 [==============================] - 69s 3ms/sample - loss: 0.1050 - acc: 0.964
9 - val_loss: 0.3740 - val_acc: 0.8788
```

In [39]:

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





2 4 6 8 10

10.4

In [40]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
```

In [41]:

```
# 6.45
max_features = 10000
max_len = 500

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

```
Loading data...
C:\Users\Nick\Anaconda3\lib\site-packages\tensorflow_core\python\keras\datasets\imdb.py:1
29: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is
a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is depre
cated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\Nick\Anaconda3\lib\site-packages\tensorflow_core\python\keras\datasets\imdb.py:1
30: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is
a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is depre
cated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 500)
x_test shape: (25000, 500)
```

In [42]:

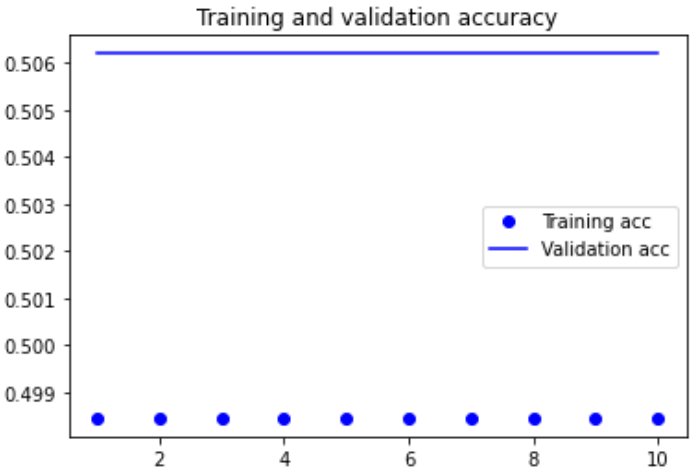
```
# 6.46
model = Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(lr=1e-4),
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

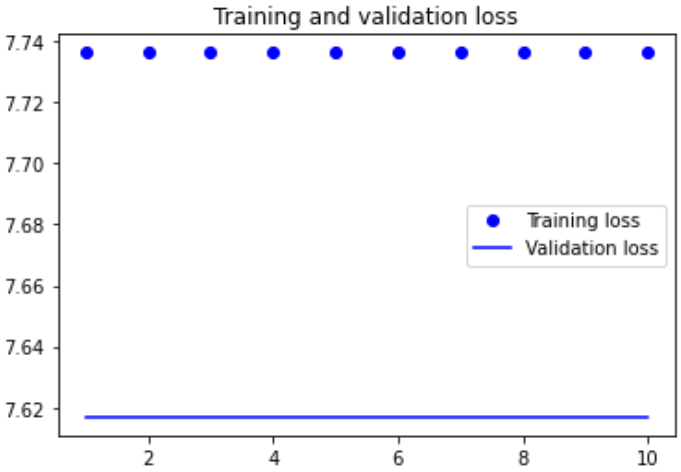
```
Model: "sequential_3"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 500, 128)          1280000
_________________________________________________________________
conv1d (Conv1D)              (None, 494, 32)           28704
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 98, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 92, 32)            7200
_________________________________________________________________
global_max_pooling1d (Global (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 48s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 2/10
20000/20000 [==============================] - 46s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 3/10
20000/20000 [==============================] - 46s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 4/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 5/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 6/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 7/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 8/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 9/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
Epoch 10/10
20000/20000 [==============================] - 47s 2ms/sample - loss: 7.7364 - acc: 0.498
5 - val_loss: 7.6168 - val_acc: 0.5062
```

In [43]:

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```


Training and validation accuracy
0.506
0.505
0.504
0.503
0.502
0.501
0.500
0.499
2
4
6
8
10
Training acc
Validation acc


Training and validation loss
7.74
7.72
7.70
7.68
7.66
7.64
7.62
2
4
6
8
10
Training loss
Validation loss

In [ ]: