In [1]:
```
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
import keras
import sys, time, os, warnings
import numpy as np
import pandas as pd
from collections import Counter
warnings.filterwarnings("ignore")
print("python {}".format(sys.version))
print("keras version {}".format(keras.__version__)); del keras
print("tensorflow version {}".format(tf.__version__))
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.95
config.gpu_options.visible_device_list = "0"
set_session(tf.Session(config=config))

def set_seed(sd=123):
    from numpy.random import seed
    from tensorflow import set_random_seed
    import random as rn
    ## numpy random seed
    seed(sd)
    ## core python's random number
    rn.seed(sd)
    ## tensor flow's random number
    set_random_seed(sd)
```

```
Using TensorFlow backend.
```

```
python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0]
keras version 2.3.1
tensorflow version 1.15.0
```

In [3]:
```
## The location of the Flickr8K_ photos
dir_Flickr_jpg = "./Flicker8k_Dataset/"
## The location of the caption file
dir_Flickr_text = "./Flickr8k.token.txt"

jpgs = os.listdir(dir_Flickr_jpg)
print("The number of jpg flies in Flicker8k: {}".format(len(jpgs)))
```

```
The number of jpg flies in Flicker8k: 8091
```

In [4]:
```
## read in the Flickr caption data
file = open(dir_Flickr_text,'r')
text = file.read()
file.close()


datatxt = []
for line in text.split('\n'):
    col = line.split('\t')
    if len(col) == 1:
        continue
    w = col[0].split("#")
    datatxt.append(w + [col[1].lower()])

df_txt = pd.DataFrame(datatxt,columns=["filename","index","caption"
])


uni_filenames = np.unique(df_txt.filename.values)
print("The number of unique file names : {}".format(len(uni_filenam
es)))
print("The distribution of the number of captions for each image:")
Counter(Counter(df_txt.filename.values).values())
```

```
The number of unique file names : 8092
The distribution of the number of captions for each image:
```

Out[4]:
```
Counter({5: 8092})
```

In [5]:
```
from keras.preprocessing.image import load_img, img_to_array

npic = 10
npix = 224
target_size = (npix,npix,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpgfnm in uni_filenames[:npic]:
    filename = dir_Flickr_jpg + '/' + jpgfnm
    captions = list(df_txt["caption"].loc[df_txt["filename"]==jpgfn
m].values)
    image_load = load_img(filename, target_size=target_size)

    ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
    ax.imshow(image_load)
    count += 1

    ax = fig.add_subplot(npic,2,count)
    plt.axis('off')
    ax.plot()
    ax.set_xlim(0,1)
    ax.set_ylim(0,len(captions))
    for i, caption in enumerate(captions):
        ax.text(0,i,caption,fontsize=20)
    count += 1
plt.show()
```

a little girl in a pink dress going into a wooden cabin .
a little girl climbing the stairs to her playhouse .
a little girl climbing into a wooden playhouse .
a girl going into a wooden building .
a child in a pink dress is climbing up a set of stairs in an entry way .

two dogs on pavement moving toward each other .
two dogs of different breeds looking at each other on the road .
a black dog and a white dog with brown spots are staring at each other in the street .
a black dog and a tri-colored dog playing with each other on the road .
a black dog and a spotted dog are fighting

young girl with pigtails painting outside in the grass .
there is a girl with pigtails sitting in front of a rainbow painting .
a small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
a little girl is sitting in front of a large painted rainbow .
a little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .

man laying on bench holding leash of dog sitting on ground
a shirtless man lies on a park bench with his dog .
a man sleeping on a bench outside with a white and black dog sitting next to him .
a man lays on the bench to which a white dog is also tied .
a man lays on a bench while his dog sits by him .

the man with pierced ears is wearing glasses and an orange hat .
a man with glasses is wearing a beer can crocheted hat .
a man with gauges and glasses is wearing a blitz hat .
a man wears an orange hat and glasses .
a man in an orange hat starring at something .

the small child climbs on a red ropes on a playground .
a small child grips onto the red ropes at the playground .
a little girl in pink climbs a rope bridge at the park .
a little girl climbing on red roping .
a child playing on a rope net .

a dog runs on the green grass near a wooden fence .
a boston terrier is running on lush green grass in front of a white fence .
a boston terrier is running in the grass .
a black and white dog is running through the grass .
a black and white dog is running in a grassy garden surrounded by a white fence .

white dog with brown ears standing near water with head turned to one side .
white dog playing with a red ball on the shore near the water .
dog with orange ball at feet , stands on shore shaking off water
a white dog shakes on the edge of a beach with an orange ball .
a dog shakes its head near the shore , a red ball next to it .

smiling boy in white shirt and blue jeans in front of rock wall with man in overalls behind him .
a young child is walking on a stone paved street with a metal pole and a man behind him .
a young boy runs aross the street .
a little boy is standing on the street while a man in overalls is working on a stone wall .
a boy smiles in front of a stony wall in a city .

the black dog jumped the tree stump .
a mottled black and grey dog in a blue collar jumping over a fallen tree .
a large black dog leaps a fallen log .
a grey dog is leaping over a fallen tree .
a black dog leaps over a log .

In [7]:
```
def df_word(df_txt):
    vocabulary = []
    for i in range(len(df_txt)):
        temp=df_txt.iloc[i,2]
        vocabulary.extend(temp.split())
    print('Vocabulary Size: %d' % len(set(vocabulary)))
    ct = Counter(vocabulary)
    dfword = pd.DataFrame({"word":list(ct.keys()),"count":list(ct.v
alues())})
    dfword = dfword.sort_values("count",ascending=False)
    dfword = dfword.reset_index()[["word","count"]]
    return(dfword)
dfword = df_word(df_txt)
dfword.head(3)
```

```
Vocabulary Size: 8918
```

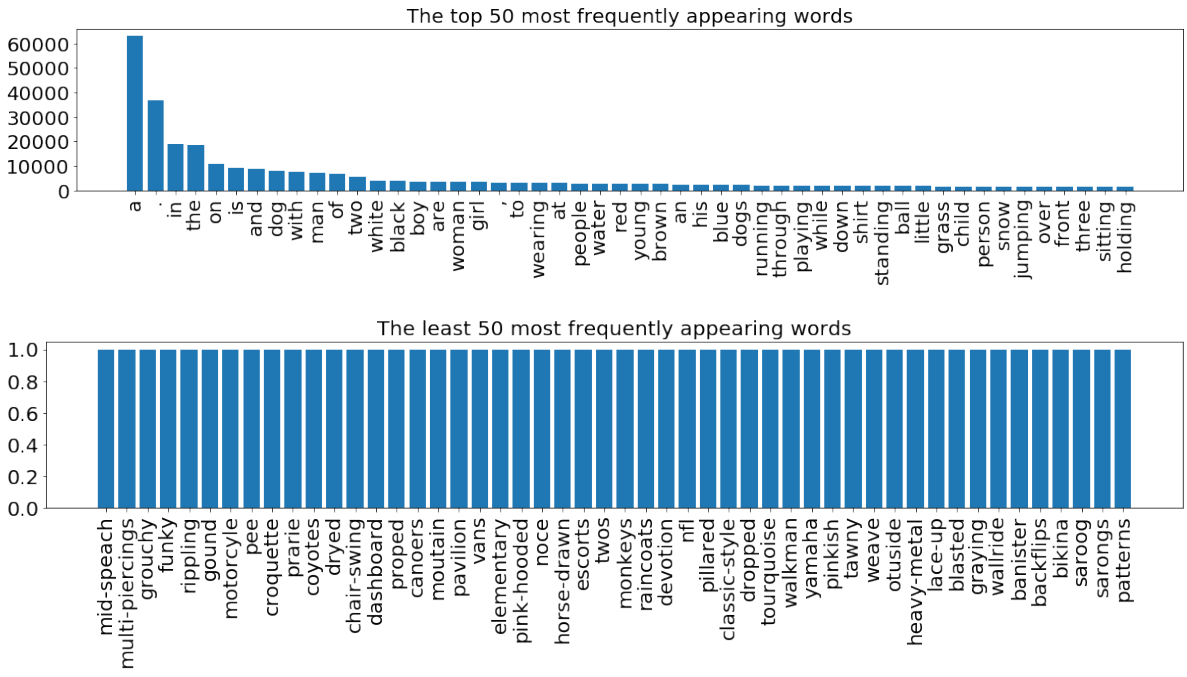
Out[7]:

	word	count
0	a	62989
1	.	36581
2	in	18975

In [8]:
```
topn = 50

def plthist(dfsub, title="The top 50 most frequently appearing word
s"):
    plt.figure(figsize=(20,3))
    plt.bar(dfsub.index,dfsub["count"])
    plt.yticks(fontsize=20)
    plt.xticks(dfsub.index,dfsub["word"],rotation=90,fontsize=20)
    plt.title(title,fontsize=20)
    plt.show()

plthist(dfword.iloc[:topn,:],
        title="The top 50 most frequently appearing words")
plthist(dfword.iloc[-topn:,:],
        title="The least 50 most frequently appearing words")
```



In [11]:
```
from copy import copy
def add_start_end_seq_token(captions):
    caps = []
    for txt in captions:
        txt = 'startseq ' + txt + ' endseq'
        caps.append(txt)
    return(caps)
df_txt0 = copy(df_txt)
df_txt0["caption"] = add_start_end_seq_token(df_txt["caption"])
df_txt0.head(5)
del df_txt
```

In [13]:
```
from keras.applications import VGG16

modelvgg = VGG16(include_top=True,weights=None)
modelvgg.summary()
```

```
WARNING:tensorflow:From /home/veda18/.conda/envs/keras/lib/python3
.7/site-packages/tensorflow_core/python/ops/resource_variable_ops.
py:1630: calling BaseResourceVariable.__init__ (from tensorflow.py
thon.ops.resource_variable_ops) with constraint is deprecated and
will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /home/veda18/.conda/envs/keras/lib/python3
.7/site-packages/keras/backend/tensorflow_backend.py:4070: The nam
e tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instea
d.

Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
```

```
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

In [14]:
```
from keras import models
modelvgg.layers.pop()
modelvgg = models.Model(inputs=modelvgg.inputs, outputs=modelvgg.la
yers[-1].output)
## show the deep learning model
modelvgg.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import preprocess_input
from collections import OrderedDict

images = OrderedDict()
npix = 224
target_size = (npix,npix,3)
data = np.zeros((len(jpgs),npix,npix,3))
for i,name in enumerate(jpgs):
    # load an image from file
    filename = dir_Flickr_jpg + '/' + name
    image = load_img(filename, target_size=target_size)
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    nimage = preprocess_input(image)

    y_pred = modelvgg.predict(nimage.reshape( (1,) + nimage.shape[:
3]))
    images[name] = y_pred.flatten()
```

```
WARNING:tensorflow:From /home/veda18/.conda/envs/keras/lib/python3
.7/site-packages/keras/backend/tensorflow_backend.py:422: The name
tf.global_variables is deprecated. Please use tf.compat.v1.global_
variables instead.
```

In [16]:
```
dimages, keepindex = [],[]
df_txt0 = df_txt0.loc[df_txt0["index"].values == "0",: ]
for i, fnm in enumerate(df_txt0.filename):
    if fnm in images.keys():
        dimages.append(images[fnm])
        keepindex.append(i)

fnames = df_txt0["filename"].iloc[keepindex].values
dcaptions = df_txt0["caption"].iloc[keepindex].values
dimages = np.array(dimages)
```

In [17]:
```
from keras.preprocessing.text import Tokenizer
## the maximum number of words in dictionary
nb_words = 8000
tokenizer = Tokenizer(nb_words=nb_words)
tokenizer.fit_on_texts(dcaptions)
vocab_size = len(tokenizer.word_index) + 1
print("vocabulary size : {}".format(vocab_size))
dtexts = tokenizer.texts_to_sequences(dcaptions)
print(dtexts[:5])
```

```
vocabulary size : 4423
[[2, 1, 39, 4, 1, 67, 145, 8, 125, 51, 1, 413, 10, 370, 4, 25, 235
3, 485, 3], [2, 1, 13, 9, 6, 1, 761, 9, 18, 371, 3], [2, 1, 49, 16
, 152, 4, 558, 102, 4, 43, 10, 1, 559, 1207, 12, 56, 219, 4, 1, 10
88, 3], [2, 1, 11, 630, 7, 1, 153, 28, 24, 9, 102, 47, 113, 3], [2
, 1, 11, 4, 25, 83, 97, 1208, 20, 167, 3]]
```

In [18]:
```
prop_test, prop_val = 0.2, 0.2

N = len(dtexts)
Ntest, Nval = int(N*prop_test), int(N*prop_val)

def split_test_val_train(dtexts,Ntest,Nval):
    return(dtexts[:Ntest],
           dtexts[Ntest:Ntest+Nval],
           dtexts[Ntest+Nval:])

dt_test,  dt_val, dt_train   = split_test_val_train(dtexts,Ntest,Nv
al)
di_test,  di_val, di_train   = split_test_val_train(dimages,Ntest,N
val)
fnm_test,fnm_val, fnm_train  = split_test_val_train(fnames,Ntest,Nv
al)
```

In [19]:
```
maxlen = np.max([len(text) for text in dtexts])
```

In [20]:
```
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

def preprocessing(dtexts,dimages):
    N = len(dtexts)
    print("# captions/images = {}".format(N))

    assert(N==len(dimages))
    Xtext, Ximage, ytext = [],[],[]
    for text,image in zip(dtexts,dimages):

        for i in range(1,len(text)):
            in_text, out_text = text[:i], text[i]
            in_text = pad_sequences([in_text],maxlen=maxlen).flatte
n()
            out_text = to_categorical(out_text,num_classes = vocab_
size)

            Xtext.append(in_text)
            Ximage.append(image)
            ytext.append(out_text)

    Xtext  = np.array(Xtext)
    Ximage = np.array(Ximage)
    ytext  = np.array(ytext)
    print(" {} {} {}".format(Xtext.shape,Ximage.shape,ytext.shape))
    return(Xtext,Ximage,ytext)


Xtext_train, Ximage_train, ytext_train = preprocessing(dt_train,di_
train)
Xtext_val,   Ximage_val,   ytext_val   = preprocessing(dt_val,di_va
l)
# pre-processing is not necessary for testing data
#Xtext_test,  Ximage_test,  ytext_test  = preprocessing(dt_test,di_
test)
```

```
# captions/images = 4855
 (59087, 35) (59087, 4096) (59087, 4423)
# captions/images = 1618
 (19489, 35) (19489, 4096) (19489, 4423)
```

In [21]:
```
from keras import layers
print(vocab_size)
## image feature

dim_embedding = 64

input_image = layers.Input(shape=(Ximage_train.shape[1],))
fimage = layers.Dense(256,activation='relu',name="ImageFeature")(in
put_image)
## sequence model
input_txt = layers.Input(shape=(maxlen,))
ftxt = layers.Embedding(vocab_size,dim_embedding, mask_zero=True)(i
nput_txt)
ftxt = layers.LSTM(256,name="CaptionFeature")(ftxt)
## combined model for decoder
decoder = layers.add([ftxt,fimage])
decoder = layers.Dense(256,activation='relu')(decoder)
output = layers.Dense(vocab_size,activation='softmax')(decoder)
model = models.Model(inputs=[input_image, input_txt],outputs=output
)

model.compile(loss='categorical_crossentropy', optimizer='adam')

print(model.summary())
```

```
4423
WARNING:tensorflow:From /home/veda18/.conda/envs/keras/lib/python3
.7/site-packages/tensorflow_core/python/keras/backend.py:3994: whe
re (from tensorflow.python.ops.array_ops) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "model_2"
__________________________________________________________________
_______________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
===============
input_3 (InputLayer)         (None, 35)                0
__________________________________________________________________
_______________
embedding_1 (Embedding)      (None, 35, 64)            283072      i
nput_3[0][0]
__________________________________________________________________
_______________
input_2 (InputLayer)         (None, 4096)              0
__________________________________________________________________
_______________
CaptionFeature (LSTM)        (None, 256)               328704      e
mbedding_1[0][0]
__________________________________________________________________
_______________
ImageFeature (Dense)         (None, 256)               1048832     i
nput_2[0][0]
__________________________________________________________________
_______________
add_1 (Add)                  (None, 256)               0           C
aptionFeature[0][0]
                                                                   I
mageFeature[0][0]
__________________________________________________________________
_______________
dense_1 (Dense)              (None, 256)               65792       a
dd_1[0][0]
__________________________________________________________________
_______________
dense_2 (Dense)              (None, 4423)              1136711     d
ense_1[0][0]
==================================================================
===============
Total params: 2,863,111
Trainable params: 2,863,111
Non-trainable params: 0
__________________________________________________________________
_______________
None
```

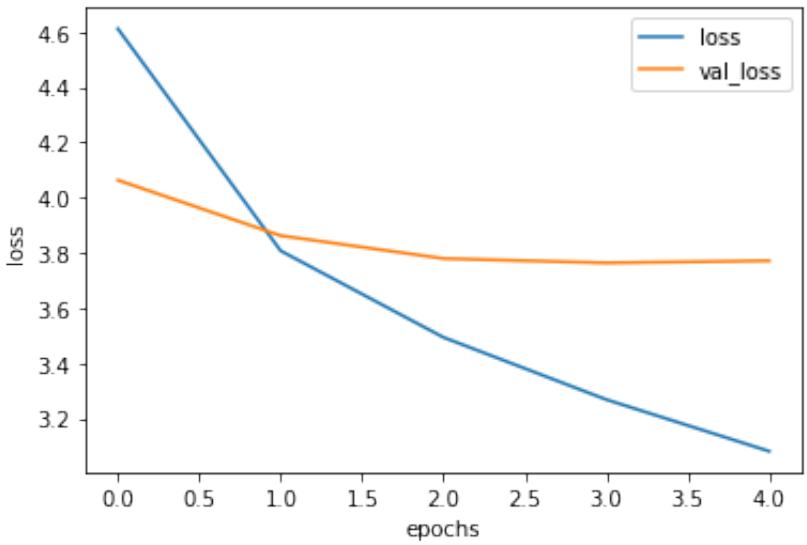
In [22]:
```
# fit model
start = time.time()
hist = model.fit([Ximage_train, Xtext_train], ytext_train,
                 epochs=5, verbose=2,
                 batch_size=64,
                 validation_data=([Ximage_val, Xtext_val], ytext_v
al))
end = time.time()
print("TIME TOOK {:3.2f}MIN".format((end - start )/60))
```

```
Train on 59087 samples, validate on 19489 samples
Epoch 1/5
 - 52s - loss: 4.6115 - val_loss: 4.0629
Epoch 2/5
 - 52s - loss: 3.8071 - val_loss: 3.8619
Epoch 3/5
 - 52s - loss: 3.4935 - val_loss: 3.7791
Epoch 4/5
 - 52s - loss: 3.2682 - val_loss: 3.7634
Epoch 5/5
 - 52s - loss: 3.0812 - val_loss: 3.7711
TIME TOOK 4.33MIN
```

In [23]:
```
print(Ximage_train.shape,Xtext_train.shape,ytext_train.shape)
```

```
(59087, 4096) (59087, 35) (59087, 4423)
```

In [24]:
```
for label in ["loss","val_loss"]:
    plt.plot(hist.history[label],label=label)
plt.legend()
plt.xlabel("epochs")
plt.ylabel("loss")
plt.show()
```



In [25]:
```
index_word = dict([(index,word) for word, index in tokenizer.word_i
ndex.items()])
def predict_caption(image):
    '''
    image.shape = (1,4462)
    '''

    in_text = 'startseq'

    for iword in range(maxlen):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence],maxlen)
        yhat = model.predict([image,sequence],verbose=0)
        yhat = np.argmax(yhat)
        newword = index_word[yhat]
        in_text += " " + newword
        if newword == "endseq":
            break
    return(in_text)



npic = 10
npix = 224
target_size = (npix,npix,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpgfnm, image_feature in zip(fnm_test[:npic],di_test[:npic]):
    ## images
    filename = dir_Flickr_jpg + '/' + jpgfnm
    image_load = load_img(filename, target_size=target_size)
    ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
    ax.imshow(image_load)
    count += 1

    ## captions
    caption = predict_caption(image_feature.reshape(1,len(image_fea
ture)))
    ax = fig.add_subplot(npic,2,count)
    plt.axis('off')
    ax.plot()
    ax.set_xlim(0,1)
    ax.set_ylim(0,1)
    ax.text(0,0.5,caption,fontsize=20)
    count += 1

plt.show()
```


startseq a man in a red shirt and a woman in a red shirt and a woman in a white shirt and a woman in a white shirt and a woman in a white shirt and


startseq a black and white dog is running through the water endseq


startseq a man in a red shirt and a white shirt is jumping in the air endseq


startseq a man in a red shirt and a woman in a red shirt and a woman in a white shirt and a woman in a white shirt and a woman in a blue shirt and


startseq a man in a red shirt and a white shirt is jumping in the air endseq


startseq a man in a red shirt and a woman in a red shirt and a woman in a white shirt and a woman in a white shirt and a woman in a white shirt and


startseq a black and white dog is running through the water endseq


startseq a black and white dog is running through the water endseq


startseq a black and white dog is running through the grass endseq


startseq a black and white dog is running through the water endseq

In [28]:
```
from nltk.translate.bleu_score import sentence_bleu
index_word = dict([(index,word) for word, index in tokenizer.word_i
ndex.items()])


nkeep = 5
pred_good, pred_bad, bleus = [], [], []
count = 0
for jpgfnm, image_feature, tokenized_text in zip(fnm_test,di_test,d
t_test):
    count += 1
    if count % 200 == 0:
        print("  {:4.2f}% is done..".format(100*count/float(len(fnm
_test))))

    caption_true = [ index_word[i] for i in tokenized_text ]
    caption_true = caption_true[1:-1] ## remove startreg, and endre
g
    ## captions
    caption = predict_caption(image_feature.reshape(1,len(image_fea
ture)))
    caption = caption.split()
    caption = caption[1:-1]## remove startreg, and endreg

    bleu = sentence_bleu([caption_true],caption)
    bleus.append(bleu)
    if bleu > 0.7 and len(pred_good) < nkeep:
        pred_good.append((bleu,jpgfnm,caption_true,caption))
    elif bleu < 0.3 and len(pred_bad) < nkeep:
        pred_bad.append((bleu,jpgfnm,caption_true,caption))
```

```
  12.36% is done..
  24.72% is done..
  37.08% is done..
  49.44% is done..
  61.80% is done..
  74.17% is done..
  86.53% is done..
  98.89% is done..
```

In [29]:
```
print("Mean BLEU {:4.3f}".format(np.mean(bleus)))
```

```
Mean BLEU 0.015
```

In [30]:
```
def plot_images(pred_bad):
    def create_str(caption_true):
        strue = ""
        for s in caption_true:
            strue += " " + s
        return(strue)
    npix = 224
    target_size = (npix,npix,3)
    count = 1
    fig = plt.figure(figsize=(10,20))
    npic = len(pred_bad)
    for pb in pred_bad:
        bleu,jpgfnm,caption_true,caption = pb
        ## images
        filename = dir_Flickr_jpg + '/' + jpgfnm
        image_load = load_img(filename, target_size=target_size)
        ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
        ax.imshow(image_load)
        count += 1

        caption_true = create_str(caption_true)
        caption = create_str(caption)

        ax = fig.add_subplot(npic,2,count)
        plt.axis('off')
        ax.plot()
        ax.set_xlim(0,1)
        ax.set_ylim(0,1)
        ax.text(0,0.7,"true:" + caption_true,fontsize=20)
        ax.text(0,0.4,"pred:" + caption,fontsize=20)
        ax.text(0,0.1,"BLEU: {}".format(bleu),fontsize=20)
        count += 1
    plt.show()

print("Bad Caption")
plot_images(pred_bad)
print("Good Caption")
plot_images(pred_good)
```

```
Bad Caption
```

true: a child in a pink dress is climbing up a set of stairs in an entry way

pred: a man in a red shirt and a woman in a red shirt and a woman in a white shirt and a woman in a white shirt and a woman in a white shirt

BLEU: 3.8540425273546335e-155

true: a black dog and a spotted dog are fighting

pred: a black and white dog is running through the water

BLEU: 6.8489908526642754e-155

true: a little girl covered in paint sits in front of a painted rainbow with her hands in a bowl

pred: a man in a red shirt and a white shirt is jumping in the air

BLEU: 4.487950221566228e-155

true: a man lays on a bench while his dog sits by him

pred: a man in a red shirt and a woman in a red shirt and a woman in a white shirt and a woman in a white shirt and a woman in a blue shirt

BLEU: 3.391988062425597e-155

true: a man in an orange hat starring at something

pred: a man in a red shirt and a white shirt is jumping in the air

BLEU: 2.6444372049983823e-78

Good Caption

true: a black and white dog is running through the field

pred: a black and white dog is running through the grass

BLEU: 0.8801117367933934

true: a black and white dog is running in the grass

pred: a black and white dog is running through the grass

BLEU: 0.7071067811865475