[Sequence to Sequence]
음성파일 위치 추적기

주제 : 회의 시에 녹음하는 파일에서 원하는 정보가 있는 부분의 시간정보를 알아낸다.

방법 :
1, [그림1]과 같이 wave file이 클라우드 서비스로 업로드되면, wave file을 분할하여 각각 lstm learning을 수행한다.
2. 각각의 lstm learning에 사용되는 network는 [그림2]와 같다. input의 sampling rate가 44100hz로 하게되면 너무 데이터량이 많아지기 때문에 decimation을 하여, sampling rate를 4000hz로 줄였고, 총 input length는 73만 sample이다.
 이를 8000개의 sequence로 LSTM의 input으로 넣고 input마다 그 해당 위치를 labeling해서 (처음과 끝을 256 level로 나누고 그 위치를 표시) 예상 output으로 사용하였따.
3. learning이 다 끝난 network에 회의내용 중 기억이 나는 키워드를 음성으로 입력하면, 그 해당 위치가 출력이 되고, 각 위치마다 원본 음성과 입력 음성의 correlation을 비교하고 이 값들을 softmax에 넣어 최종 위치를 알아내는 방식이다.

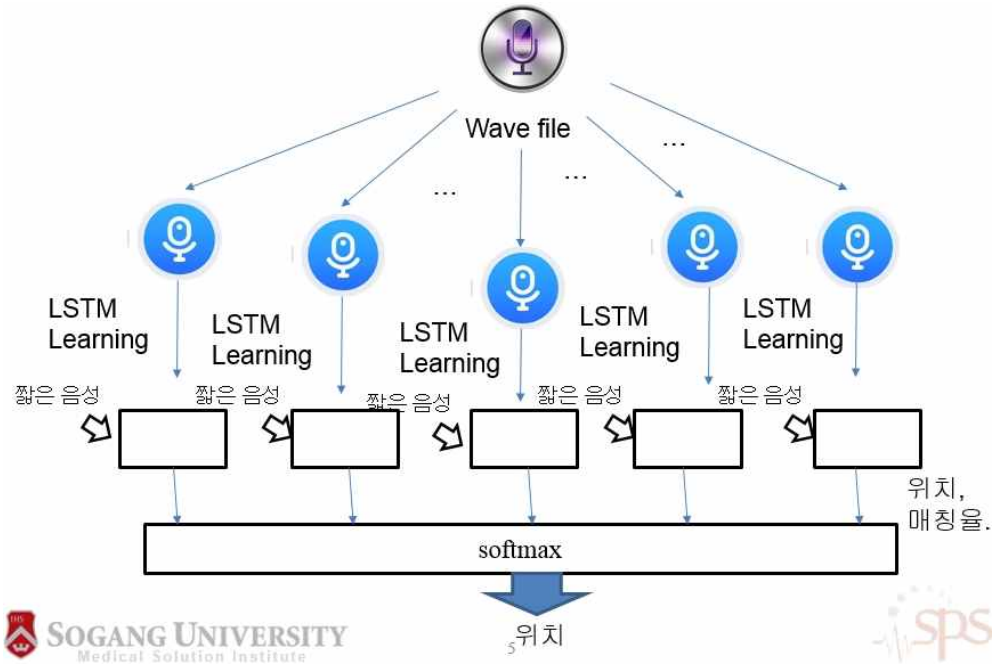
Wave file
LSTM Learning
LSTM Learning
LSTM Learning
LSTM Learning
LSTM Learning
짧은 음성
짧은 음성
짧은 음성
짧은 음성
짧은 음성
위치,
매칭율.
softmax
위치
SOGANG UNIVERSITY
Medical Solution Institute
SPS

그림 1

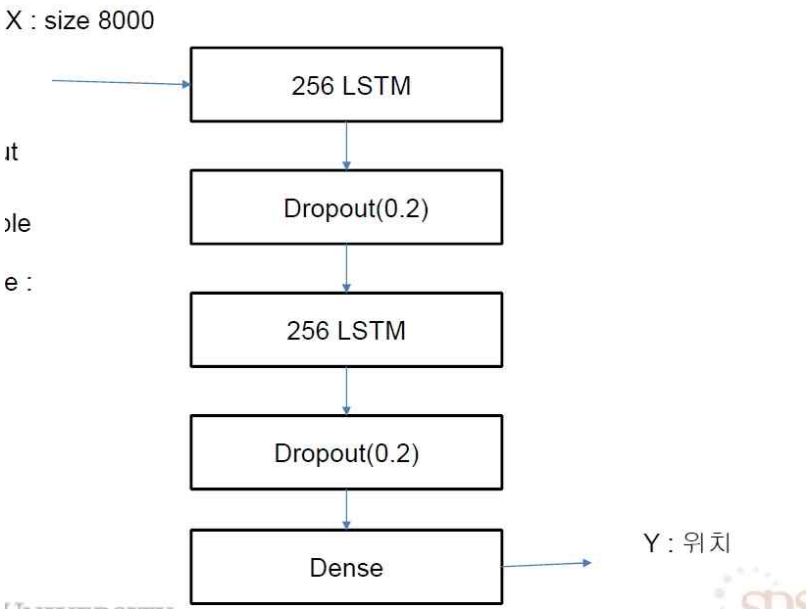
X : size 8000
256 LSTM
Dropout(0.2)
256 LSTM
Dropout(0.2)
Dense
Y : 위치

그림 2

결과 :

```
model.add(Dropout(0.2))
model.add(LSTM(256))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
# define the checkpoint
filepath="weights-improvement-{epoch:02d}-{loss:.4f}-bigger.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True,
mode='min')
callbacks_list = [checkpoint]
# fit the model
model.fit(X, y, epochs=50, batch_size=4, callbacks=callbacks_list)
```

```
Epoch 1/50
208/366 [================>.............] - ETA: 1237s - loss: 5.5783
```

```
[ ]: wavefile=wave.open('seed.wav','r')
nchannel=wavefile.getnchannels()
slength=wavefile.getnframes()
print(nchannel)
print(slength)
print(wavefile.getframerate())
```

그림 3

현재 컴퓨터 세팅으로 50 epoch를 돌리는데 2일이 걸려 시간에 맞추질 못하여 결과를 내진 못했지만, 발표때와는 달리 돌아가도록 코드를 잘 구성하였다.

코드 :

In [1]:

```
import wave
import struct

wavefile=wave.open('3mm_decimate4.wav','r')
nchannel=wavefile.getnchannels()
length=wavefile.getnframes()
print(nchannel)
print(length)
print(wavefile.getframerate())
print(wavefile.getsampwidth())

sampleRate = wavefile.getframerate() # hertz
duration = 1.0       # seconds
frequency = 440.0    # hertz

nread=length
if nread<=length:
    nbytes=nchannel*nread
    format="<" + str(nbytes) + "h"
    wavedata=wavefile.readframes(nread)
    data_a=struct.unpack(format,wavedata)
wavefile.close()

if nchannel==1:
    wave_channel1=list(data_a)
elif nchannel==2:
    wave_channel1=list(data_a[0::2])
    wave_channel2=list(data_a[1::2])
```

In [2]:
```
import sys
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils


seq_length = 8000
dataX = []
dataY = []
print('1phase done')
for i in range(0, (length - seq_length),2000):
    seq_in = wave_channel1[i:i + seq_length]
    dataX.append( seq_in)
    dataY.append(i)
print('2phase done')
n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)
# reshape X to be [samples, time steps, features]
X = numpy.reshape(dataX, (n_patterns, seq_length, 1))
# normalize
X = X / 32768.0
```

In [ ]:
```
# one hot encode the output variable
samplepos=length/256;
datY=[(int)((dat)/samplepos) for dat in dataY]
y = np_utils.to_categorical(datY)

# define the LSTM model
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(256))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
# define the checkpoint
filepath="weights-improvement-{epoch:02d}-{loss:.4f}-bigger.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save_best_only=True,
mode='min')
callbacks_list = [checkpoint]
# fit the model
model.fit(X, y, epochs=50, batch_size=4, callbacks=callbacks_list)
```

```
Epoch 1/50

216/366 [================>.............] - ETA: 1175s - loss: 5.5769
```

In [ ]:
```
wavefile=wave.open('seed.wav','r')
nchannel=wavefile.getnchannels()
slength=wavefile.getnframes()
print(nchannel)
print(slength)
print(wavefile.getframerate())
print(wavefile.getsampwidth())

sampleRate = wavefile.getframerate() # hertz
duration = 1.0       # seconds
frequency = 440.0    # hertz

sread=slength
if sread<=length:
    nbytes=nchannel*sread
    format="<" + str(nbytes) + "h"
    wavedata=wavefile.readframes(sread)
    data_a=struct.unpack(format,wavedata)
wavefile.close()

if nchannel==1:
    seed1=list(data_a)
elif nchannel==2:
    seed1=list(data_a[0::2])
    seed2=list(data_a[1::2])
```

In [ ]:
```
# define the LSTM model
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(256))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
# load the network weights
filename = "weights-improvement-30-1.7248-bigger.hdf5"
model.load_weights(filename)
model.compile(loss='categorical_crossentropy', optimizer='adam')

# pick a random seed
pattern = seed1[0:8000]
print("Seed:")400-
# generate characters
x = numpy.reshape(pattern, (1, len(pattern), 1))
x = x / float(n_vocab)
prediction = model.predict(x, verbose=0)
index = numpy.argmax(prediction)
print("위치 : "+(int)(index*samplepos/4000)+"초")
print("\nDone.")
```