Human Activity Recognition

This project is to build a model that predicts the human activities such as Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing or Laying.

This dataset is collected from 30 persons(referred as subjects in this dataset), performing different activities with a smartphone to their waists. The data is recorded with the help of sensors (accelerometer and Gyroscope) in that smartphone. This experiment was video recorded to label the data manually.

- Data source :-https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones (https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones)

How data was recorded

By using the sensors(Gyroscope and accelerometer) in a smartphone, they have captured '3-axial linear acceleration'(*tAcc-XYZ*) from accelerometer and '3-axial angular velocity' (*tGyro-XYZ*) from Gyroscope with several variations.

> prefix 't' in those metrics denotes time.
>
> suffix 'XYZ' represents 3-axial signals in X , Y, and Z directions.\

Train and test data were saperated

- The readings from ***70%*** of the volunteers were taken as ***trianing data*** and remaining ***30%*** subjects recordings were taken for ***test data***

In [1]:
```
# Importing Libraries
```

In [22]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import seaborn as sns
```

In [3]:
```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

# Utility function to print the confusion matrix
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

Data

In [4]:
```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

In [5]:
```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

In [6]:
```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subse
t}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signa
ls)
    return np.transpose(signals_data, (1, 2, 0))
```

In [7]:
```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummie
s.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

In [8]:
```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

In [9]:
```
import tensorflow as tf
from matplotlib import pyplot
np.random.seed(42)

tf.set_random_seed(42)
```

In [10]:
```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

In [11]:
```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

```
Using TensorFlow backend.
```

In [12]:
```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

In [13]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

In [14]:
```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

In [15]:
```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

In [16]:
```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print(timesteps)
print(input_dim)
print(len(X_train))
```

```
128
9
7352
```

(A) 2 layer with dropout rate 0.8

- Defining the Architecture of LSTM

In [51]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.8))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

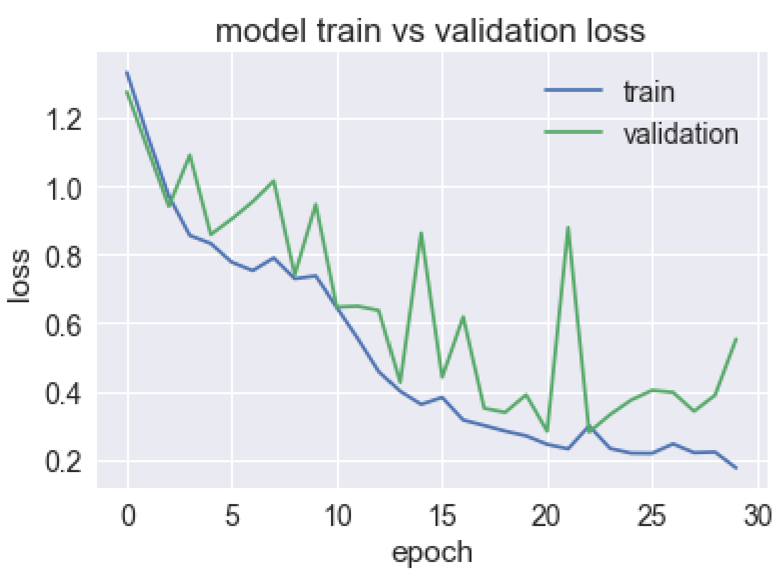
In [52]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [53]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 32s 4ms/step - loss: 1.3340 - ac
c: 0.4357 - val_loss: 1.2770 - val_acc: 0.4164
Epoch 2/30
7352/7352 [==============================] - 33s 4ms/step - loss: 1.1464 - ac
c: 0.5076 - val_loss: 1.1079 - val_acc: 0.5443
Epoch 3/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.9714 - ac
c: 0.5896 - val_loss: 0.9413 - val_acc: 0.5996
Epoch 4/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.8558 - ac
c: 0.6141 - val_loss: 1.0917 - val_acc: 0.5171
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.8323 - ac
c: 0.6224 - val_loss: 0.8586 - val_acc: 0.5962
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7773 - ac
c: 0.6428 - val_loss: 0.9042 - val_acc: 0.6037
Epoch 7/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.7528 - ac
c: 0.6491 - val_loss: 0.9550 - val_acc: 0.6030
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7899 - ac
c: 0.6387 - val_loss: 1.0157 - val_acc: 0.5921
Epoch 9/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7290 - ac
c: 0.6593 - val_loss: 0.7404 - val_acc: 0.6244
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7373 - ac
c: 0.6729 - val_loss: 0.9470 - val_acc: 0.6634
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.6461 - ac
c: 0.7360 - val_loss: 0.6471 - val_acc: 0.7411
Epoch 12/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.5552 - ac
c: 0.8156 - val_loss: 0.6499 - val_acc: 0.8185
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4579 - ac
c: 0.8667 - val_loss: 0.6373 - val_acc: 0.8514
Epoch 14/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.4004 - ac
c: 0.8927 - val_loss: 0.4263 - val_acc: 0.8700
Epoch 15/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.3620 - ac
c: 0.8974 - val_loss: 0.8630 - val_acc: 0.7822
Epoch 16/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3824 - ac
c: 0.8953 - val_loss: 0.4413 - val_acc: 0.8673
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3160 - ac
c: 0.9129 - val_loss: 0.6186 - val_acc: 0.8609
Epoch 18/30
7352/7352 [==============================] - 47s 6ms/step - loss: 0.3000 - ac
c: 0.9170 - val_loss: 0.3506 - val_acc: 0.8816
Epoch 19/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2835 - ac
```

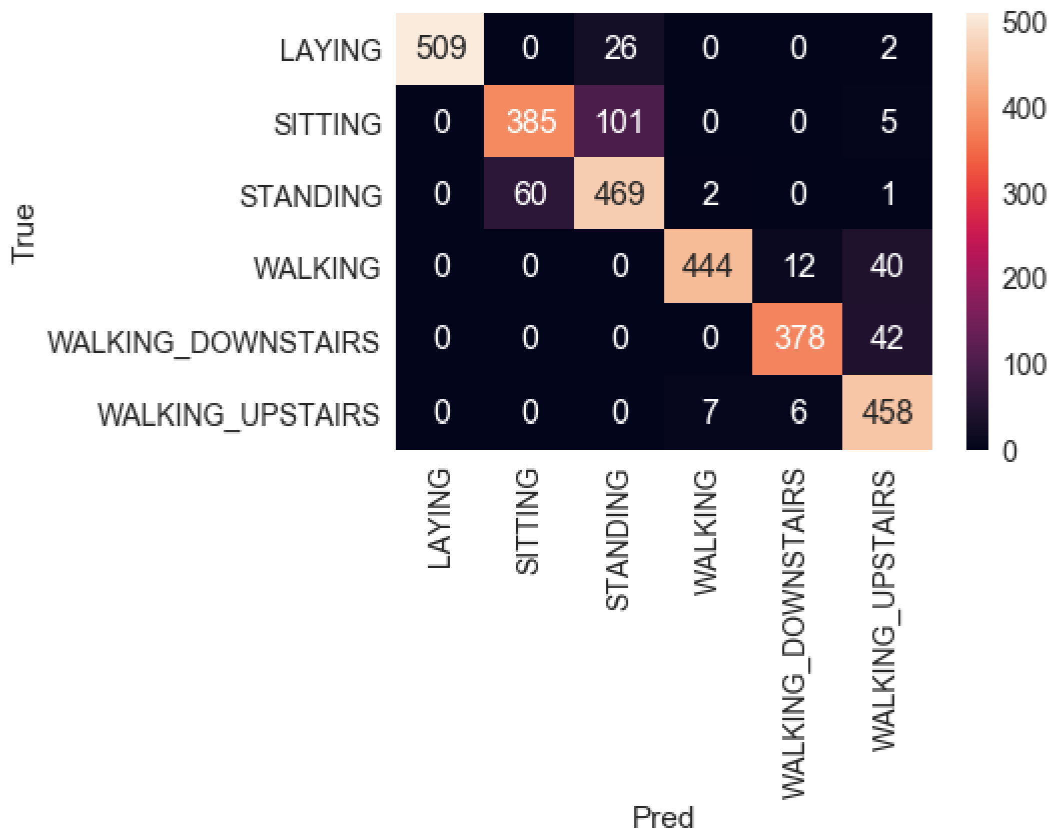
```
c: 0.9225 - val_loss: 0.3384 - val_acc: 0.9013
Epoch 20/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2696 - ac
c: 0.9234 - val_loss: 0.3900 - val_acc: 0.8911
Epoch 21/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2447 - ac
c: 0.9293 - val_loss: 0.2829 - val_acc: 0.9108
Epoch 22/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2314 - ac
c: 0.9237 - val_loss: 0.8795 - val_acc: 0.8293
Epoch 23/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2988 - ac
c: 0.9215 - val_loss: 0.2804 - val_acc: 0.9152
Epoch 24/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.2326 - ac
c: 0.9302 - val_loss: 0.3320 - val_acc: 0.9067
Epoch 25/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2189 - ac
c: 0.9329 - val_loss: 0.3745 - val_acc: 0.8914
Epoch 26/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.2184 - ac
c: 0.9357 - val_loss: 0.4032 - val_acc: 0.9043
Epoch 27/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.2460 - ac
c: 0.9306 - val_loss: 0.3972 - val_acc: 0.8985
Epoch 28/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.2202 - ac
c: 0.9358 - val_loss: 0.3417 - val_acc: 0.9104
Epoch 29/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.2221 - ac
c: 0.9308 - val_loss: 0.3888 - val_acc: 0.9060
Epoch 30/30
7352/7352 [==============================] - 66s 9ms/step - loss: 0.1755 - ac
c: 0.9387 - val_loss: 0.5532 - val_acc: 0.8968
```



In [54]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4b7bd7710>



In [55]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 3s 1ms/step
```

In [56]:
```
score
```

Out[56]: [0.5532151480927816, 0.8968442483881914]

- With a simple 2 layer architecture we got 89.68% accuracy and a loss of 0.55
- We can further imporve the performace with Hyperparameter tuning

(B) 2 layer with dropout rate 0.7

In [17]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 64)                18944
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

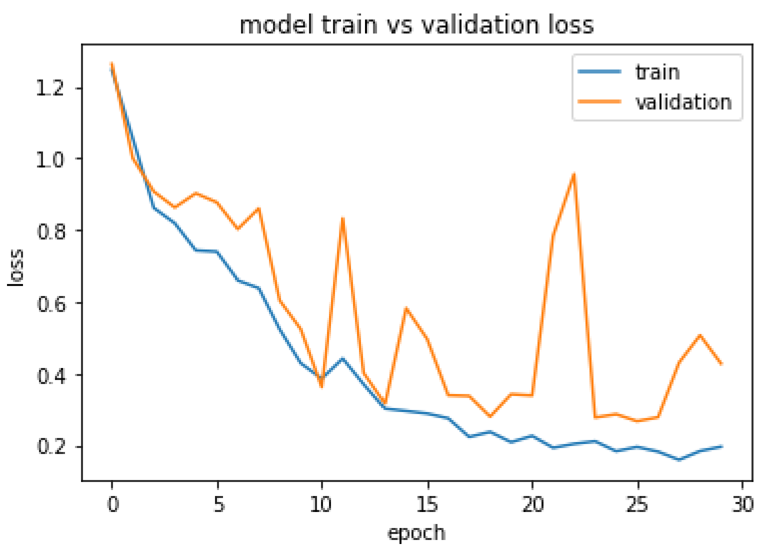
In [18]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [20]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 32s 4ms/step - loss: 1.2472 - ac
c: 0.4629 - val_loss: 1.2637 - val_acc: 0.4445
Epoch 2/30
7352/7352 [==============================] - 33s 4ms/step - loss: 1.0554 - ac
c: 0.5482 - val_loss: 1.0001 - val_acc: 0.5606
Epoch 3/30
7352/7352 [==============================] - 35s 5ms/step - loss: 0.8615 - ac
c: 0.6224 - val_loss: 0.9072 - val_acc: 0.6200
Epoch 4/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.8186 - ac
c: 0.6579 - val_loss: 0.8627 - val_acc: 0.6756
Epoch 5/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7427 - ac
c: 0.7031 - val_loss: 0.9022 - val_acc: 0.6773
Epoch 6/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.7393 - ac
c: 0.7078 - val_loss: 0.8770 - val_acc: 0.6576
Epoch 7/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.6600 - ac
c: 0.7352 - val_loss: 0.8023 - val_acc: 0.7078
Epoch 8/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.6385 - ac
c: 0.7731 - val_loss: 0.8602 - val_acc: 0.7503
Epoch 9/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.5232 - ac
c: 0.8194 - val_loss: 0.6040 - val_acc: 0.8086
Epoch 10/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4289 - ac
c: 0.8659 - val_loss: 0.5240 - val_acc: 0.8100
Epoch 11/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3856 - ac
c: 0.8848 - val_loss: 0.3620 - val_acc: 0.8772
Epoch 12/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.4418 - ac
c: 0.8813 - val_loss: 0.8325 - val_acc: 0.7543
Epoch 13/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.3696 - ac
c: 0.8897 - val_loss: 0.4011 - val_acc: 0.8785
Epoch 14/30
7352/7352 [==============================] - 28s 4ms/step - loss: 0.3019 - ac
c: 0.9089 - val_loss: 0.3165 - val_acc: 0.8955
Epoch 15/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2952 - ac
c: 0.9108 - val_loss: 0.5825 - val_acc: 0.8554
Epoch 16/30
7352/7352 [==============================] - 31s 4ms/step - loss: 0.2882 - ac
c: 0.9136 - val_loss: 0.4963 - val_acc: 0.8470
Epoch 17/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2749 - ac
c: 0.9168 - val_loss: 0.3393 - val_acc: 0.8785
Epoch 18/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2227 - ac
c: 0.9275 - val_loss: 0.3372 - val_acc: 0.9070
Epoch 19/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2368 - ac
```

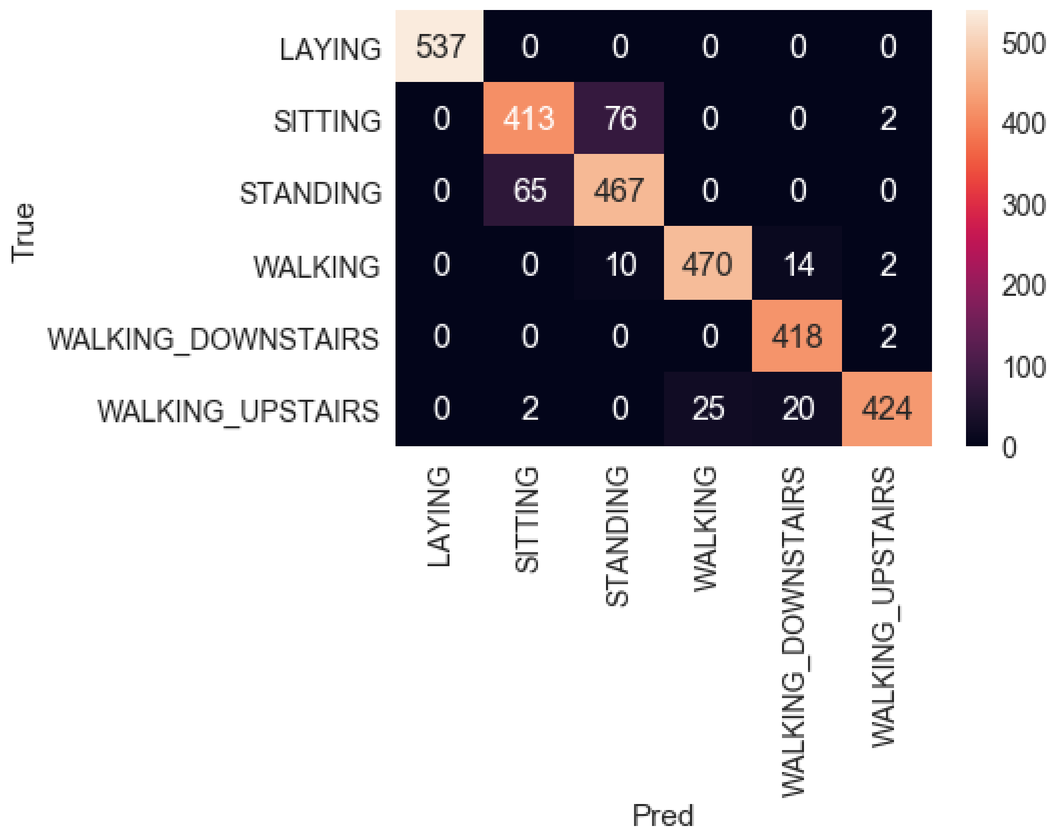
```
c: 0.9252 - val_loss: 0.2795 - val_acc: 0.9226
Epoch 20/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2080 - ac
c: 0.9329 - val_loss: 0.3421 - val_acc: 0.9006
Epoch 21/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.2257 - ac
c: 0.9343 - val_loss: 0.3384 - val_acc: 0.9057
Epoch 22/30
7352/7352 [==============================] - 30s 4ms/step - loss: 0.1923 - ac
c: 0.9353 - val_loss: 0.7833 - val_acc: 0.8551
Epoch 23/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2033 - ac
c: 0.9395 - val_loss: 0.9561 - val_acc: 0.8371
Epoch 24/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.2106 - ac
c: 0.9357 - val_loss: 0.2769 - val_acc: 0.9128
Epoch 25/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1828 - ac
c: 0.9385 - val_loss: 0.2863 - val_acc: 0.9291
Epoch 26/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1948 - ac
c: 0.9418 - val_loss: 0.2667 - val_acc: 0.9223
Epoch 27/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1816 - ac
c: 0.9423 - val_loss: 0.2777 - val_acc: 0.9186
Epoch 28/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1586 - ac
c: 0.9416 - val_loss: 0.4304 - val_acc: 0.9087
Epoch 29/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1833 - ac
c: 0.9441 - val_loss: 0.5070 - val_acc: 0.9080
Epoch 30/30
7352/7352 [==============================] - 29s 4ms/step - loss: 0.1953 - ac
c: 0.9421 - val_loss: 0.4276 - val_acc: 0.9260
```



In [23]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4a8095128>



In [24]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 487us/step
```

In [25]:
```
score
```

Out[25]: [0.42758658551398804, 0.9260264675941635]

(C) 2 layer with dropout rate 0.5

- Defining the Architecture of LSTM

In [26]:

```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 32)                5376
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

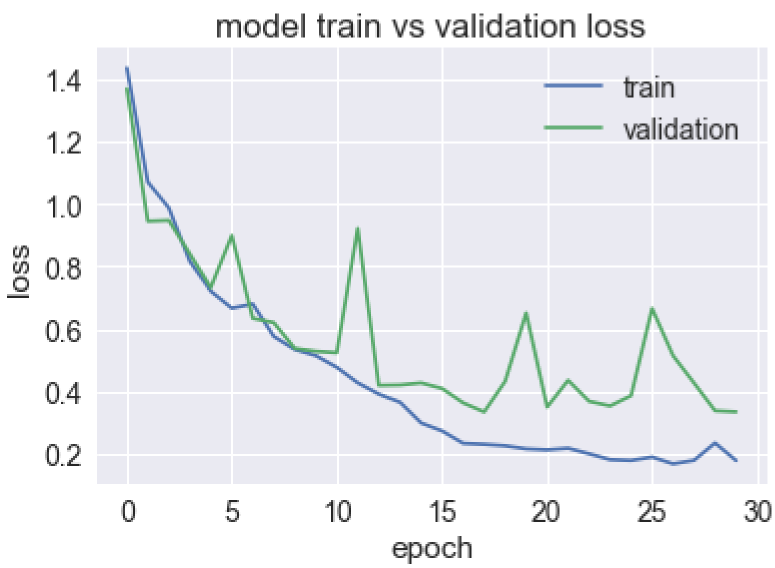
In [27]:

```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [28]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 26s 4ms/step - loss: 1.4364 - ac
c: 0.3526 - val_loss: 1.3697 - val_acc: 0.3797
Epoch 2/30
7352/7352 [==============================] - 25s 3ms/step - loss: 1.0705 - ac
c: 0.5405 - val_loss: 0.9452 - val_acc: 0.5816
Epoch 3/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.9877 - ac
c: 0.5724 - val_loss: 0.9476 - val_acc: 0.5643
Epoch 4/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.8164 - ac
c: 0.6187 - val_loss: 0.8402 - val_acc: 0.6006
Epoch 5/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.7219 - ac
c: 0.6498 - val_loss: 0.7325 - val_acc: 0.6186
Epoch 6/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.6677 - ac
c: 0.6910 - val_loss: 0.8990 - val_acc: 0.6138
Epoch 7/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.6811 - ac
c: 0.7193 - val_loss: 0.6352 - val_acc: 0.7319
Epoch 8/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.5771 - ac
c: 0.7692 - val_loss: 0.6214 - val_acc: 0.7299
Epoch 9/30
7352/7352 [==============================] - 26s 4ms/step - loss: 0.5352 - ac
c: 0.7790 - val_loss: 0.5385 - val_acc: 0.7387
Epoch 10/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.5155 - ac
c: 0.7886 - val_loss: 0.5292 - val_acc: 0.7631
Epoch 11/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4781 - ac
c: 0.8109 - val_loss: 0.5247 - val_acc: 0.8035
Epoch 12/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.4282 - ac
c: 0.8411 - val_loss: 0.9218 - val_acc: 0.7353
Epoch 13/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3922 - ac
c: 0.8690 - val_loss: 0.4202 - val_acc: 0.8629
Epoch 14/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.3656 - ac
c: 0.8870 - val_loss: 0.4213 - val_acc: 0.8683
Epoch 15/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2994 - ac
c: 0.9121 - val_loss: 0.4280 - val_acc: 0.8666
Epoch 16/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.2738 - ac
c: 0.9184 - val_loss: 0.4098 - val_acc: 0.8721
Epoch 17/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2342 - ac
c: 0.9248 - val_loss: 0.3639 - val_acc: 0.8782
Epoch 18/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2311 - ac
c: 0.9275 - val_loss: 0.3348 - val_acc: 0.8992
Epoch 19/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2264 - ac
```

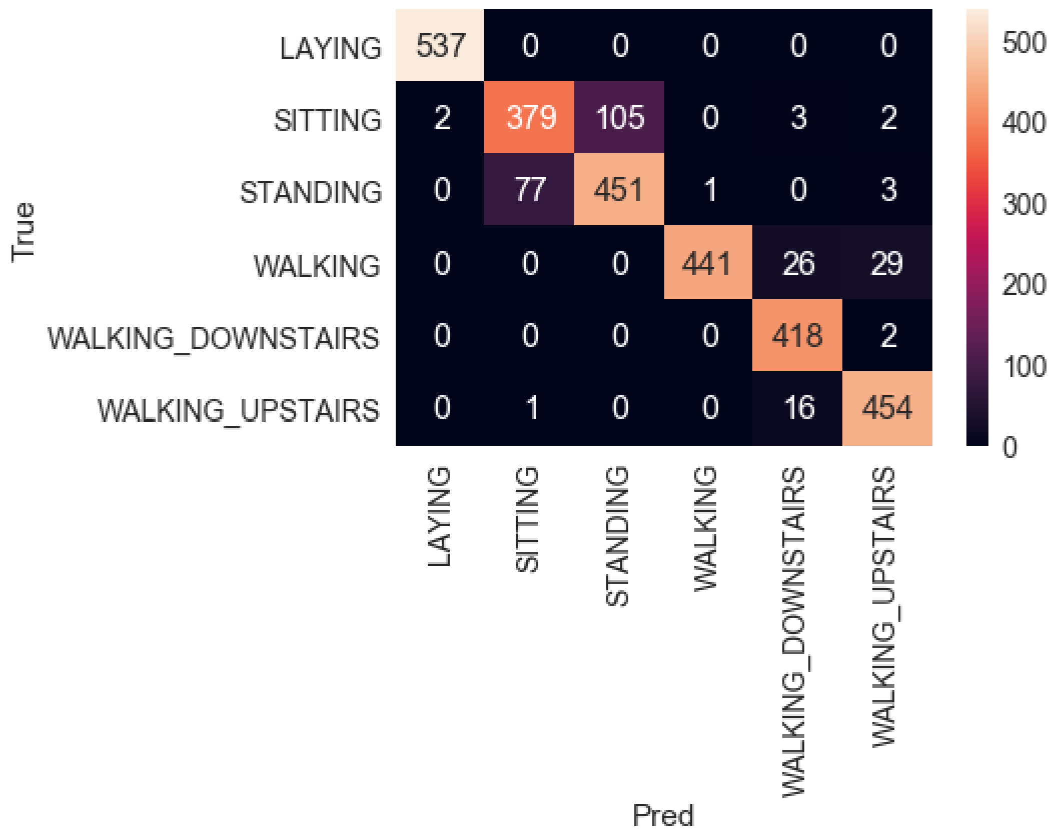
```
c: 0.9290 - val_loss: 0.4330 - val_acc: 0.9006
Epoch 20/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2164 - ac
c: 0.9324 - val_loss: 0.6524 - val_acc: 0.8507
Epoch 21/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2132 - ac
c: 0.9358 - val_loss: 0.3505 - val_acc: 0.9087
Epoch 22/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2185 - ac
c: 0.9334 - val_loss: 0.4364 - val_acc: 0.8843
Epoch 23/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2006 - ac
c: 0.9363 - val_loss: 0.3689 - val_acc: 0.9016
Epoch 24/30
7352/7352 [==============================] - 23s 3ms/step - loss: 0.1817 - ac
c: 0.9437 - val_loss: 0.3541 - val_acc: 0.8945
Epoch 25/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1792 - ac
c: 0.9416 - val_loss: 0.3870 - val_acc: 0.8935
Epoch 26/30
7352/7352 [==============================] - 25s 3ms/step - loss: 0.1901 - ac
c: 0.9387 - val_loss: 0.6668 - val_acc: 0.8537
Epoch 27/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1684 - ac
c: 0.9426 - val_loss: 0.5152 - val_acc: 0.8955
Epoch 28/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1797 - ac
c: 0.9402 - val_loss: 0.4276 - val_acc: 0.9050
Epoch 29/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.2353 - ac
c: 0.9353 - val_loss: 0.3386 - val_acc: 0.9155
Epoch 30/30
7352/7352 [==============================] - 24s 3ms/step - loss: 0.1789 - ac
c: 0.9440 - val_loss: 0.3352 - val_acc: 0.9094
```



In [29]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4a8d6de10>



In [30]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 1s 331us/step
```

In [31]:
```
score
```

Out[31]: [0.3352397249300588, 0.9093993892093655]

(A) 3 layer with dropout rate 0.8.

In [32]:
```
# Initializing parameters
epochs = 30
batch_size = 16
n_hidden = 64
```

- Defining the Architecture of LSTM

In [33]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),return_sequences=True))
model.add(LSTM(32))
# Adding a dropout layer
model.add(Dropout(0.8))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

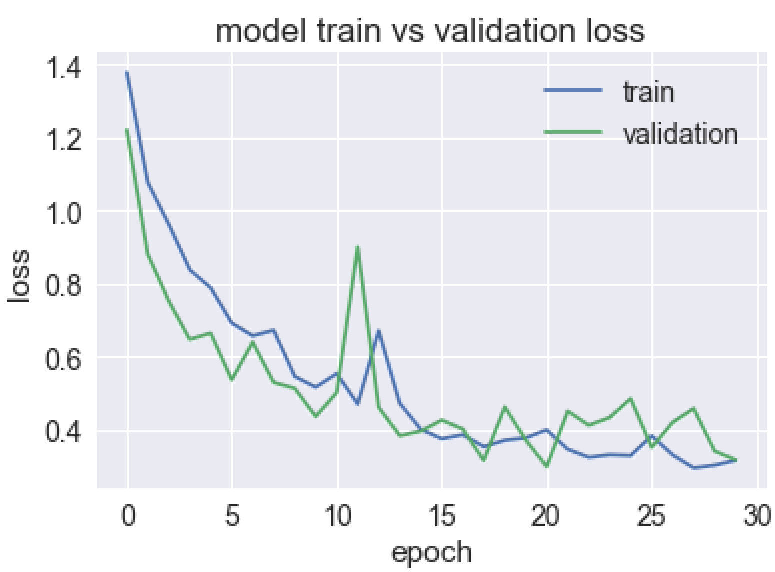
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 128, 64)           18944     
_________________________________________________________________
lstm_4 (LSTM)                (None, 32)                12416     
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 198       
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [35]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 61s 8ms/step - loss: 1.3778 - ac
c: 0.4510 - val_loss: 1.2208 - val_acc: 0.5005
Epoch 2/30
7352/7352 [==============================] - 58s 8ms/step - loss: 1.0764 - ac
c: 0.5736 - val_loss: 0.8800 - val_acc: 0.7011
Epoch 3/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.9621 - ac
c: 0.6246 - val_loss: 0.7530 - val_acc: 0.7211
Epoch 4/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.8379 - ac
c: 0.6616 - val_loss: 0.6478 - val_acc: 0.7414
Epoch 5/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.7886 - ac
c: 0.6895 - val_loss: 0.6650 - val_acc: 0.7343
Epoch 6/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.6926 - ac
c: 0.7282 - val_loss: 0.5373 - val_acc: 0.8290
Epoch 7/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.6576 - ac
c: 0.7477 - val_loss: 0.6405 - val_acc: 0.8426
Epoch 8/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.6726 - ac
c: 0.7854 - val_loss: 0.5301 - val_acc: 0.8738
Epoch 9/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.5463 - ac
c: 0.8183 - val_loss: 0.5141 - val_acc: 0.8568
Epoch 10/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.5170 - ac
c: 0.8304 - val_loss: 0.4368 - val_acc: 0.8904
Epoch 11/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.5541 - ac
c: 0.8390 - val_loss: 0.5020 - val_acc: 0.8741
Epoch 12/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.4709 - ac
c: 0.8588 - val_loss: 0.9005 - val_acc: 0.8436
Epoch 13/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.6720 - ac
c: 0.8398 - val_loss: 0.4620 - val_acc: 0.8694
Epoch 14/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.4730 - ac
c: 0.8629 - val_loss: 0.3843 - val_acc: 0.8806
Epoch 15/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.4016 - ac
c: 0.8842 - val_loss: 0.3968 - val_acc: 0.9077
Epoch 16/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.3758 - ac
c: 0.8940 - val_loss: 0.4270 - val_acc: 0.9036
Epoch 17/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.3870 - ac
c: 0.8890 - val_loss: 0.4024 - val_acc: 0.8975
Epoch 18/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.3544 - ac
c: 0.8927 - val_loss: 0.3169 - val_acc: 0.9043
Epoch 19/30
7352/7352 [==============================] - 78s 11ms/step - loss: 0.3716 - a
```

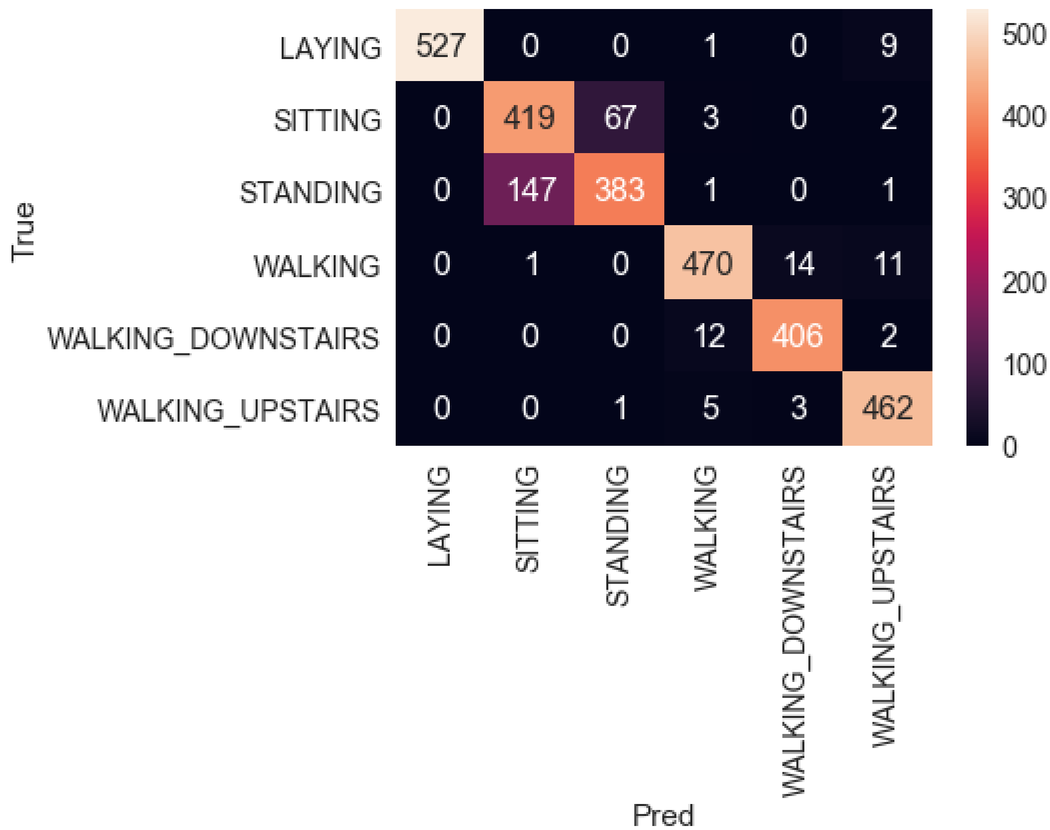
```
cc: 0.8951 - val_loss: 0.4631 - val_acc: 0.8938
Epoch 20/30
7352/7352 [==============================] - 121s 16ms/step - loss: 0.3785 -
acc: 0.8984 - val_loss: 0.3724 - val_acc: 0.9040
Epoch 21/30
7352/7352 [==============================] - 118s 16ms/step - loss: 0.3999 -
acc: 0.8928 - val_loss: 0.2999 - val_acc: 0.9087
Epoch 22/30
7352/7352 [==============================] - 114s 15ms/step - loss: 0.3472 -
acc: 0.9004 - val_loss: 0.4515 - val_acc: 0.8941
Epoch 23/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.3251 -
acc: 0.9017 - val_loss: 0.4128 - val_acc: 0.8955
Epoch 24/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.3322 -
acc: 0.9038 - val_loss: 0.4342 - val_acc: 0.8955
Epoch 25/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.3300 -
acc: 0.8991 - val_loss: 0.4859 - val_acc: 0.9077
Epoch 26/30
7352/7352 [==============================] - 123s 17ms/step - loss: 0.3837 -
acc: 0.8987 - val_loss: 0.3525 - val_acc: 0.9087
Epoch 27/30
7352/7352 [==============================] - 124s 17ms/step - loss: 0.3318 -
acc: 0.9074 - val_loss: 0.4210 - val_acc: 0.9050
Epoch 28/30
7352/7352 [==============================] - 125s 17ms/step - loss: 0.2960 -
acc: 0.9082 - val_loss: 0.4589 - val_acc: 0.9128
Epoch 29/30
7352/7352 [==============================] - 120s 16ms/step - loss: 0.3036 -
acc: 0.9068 - val_loss: 0.3422 - val_acc: 0.9002
Epoch 30/30
7352/7352 [==============================] - 126s 17ms/step - loss: 0.3167 -
acc: 0.9047 - val_loss: 0.3181 - val_acc: 0.9050
```



In [36]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4aad47cc0>



In [37]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 7s 2ms/step
```

In [38]:
```
score
```

Out[38]: [0.3180937306507453, 0.9049881235154394]

- With a 3 layer architecture we got 90.49% accuracy and a loss of 0.31

(B) 3 layer with Dropout rate 0.9

In [39]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim),return_sequences=True))
model.add(LSTM(32))
# Adding a dropout layer
model.add(Dropout(0.9))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

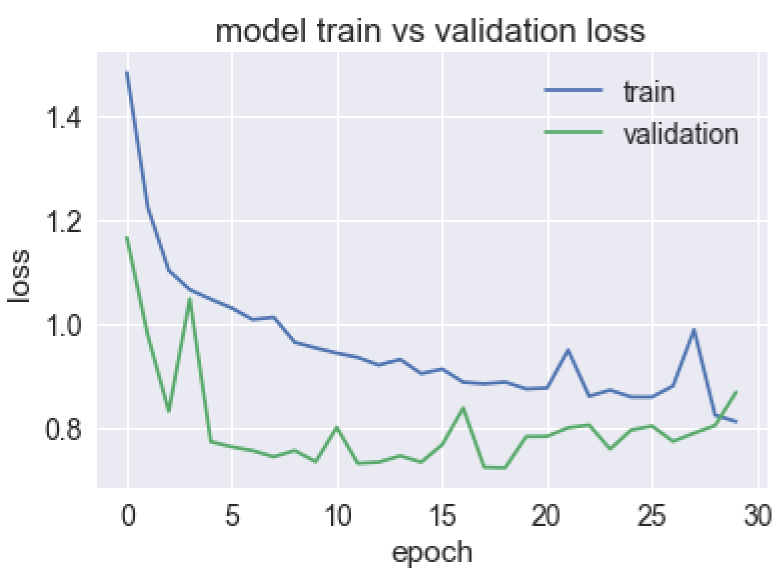
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
lstm_6 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 198
=================================================================
Total params: 31,558
Trainable params: 31,558
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [41]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 129s 18ms/step - loss: 1.4826 -
acc: 0.3882 - val_loss: 1.1651 - val_acc: 0.5850
Epoch 2/30
7352/7352 [==============================] - 129s 18ms/step - loss: 1.2229 -
acc: 0.4800 - val_loss: 0.9770 - val_acc: 0.5806
Epoch 3/30
7352/7352 [==============================] - 128s 17ms/step - loss: 1.1018 -
acc: 0.5033 - val_loss: 0.8310 - val_acc: 0.6060
Epoch 4/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.0650 -
acc: 0.5166 - val_loss: 1.0473 - val_acc: 0.6074
Epoch 5/30
7352/7352 [==============================] - 126s 17ms/step - loss: 1.0466 -
acc: 0.5188 - val_loss: 0.7727 - val_acc: 0.6121
Epoch 6/30
7352/7352 [==============================] - 129s 18ms/step - loss: 1.0296 -
acc: 0.5165 - val_loss: 0.7627 - val_acc: 0.6105
Epoch 7/30
7352/7352 [==============================] - 128s 17ms/step - loss: 1.0074 -
acc: 0.5201 - val_loss: 0.7553 - val_acc: 0.6250
Epoch 8/30
7352/7352 [==============================] - 103s 14ms/step - loss: 1.0119 -
acc: 0.5301 - val_loss: 0.7436 - val_acc: 0.5894
Epoch 9/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.9637 - ac
c: 0.5288 - val_loss: 0.7556 - val_acc: 0.6067
Epoch 10/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.9528 - ac
c: 0.5339 - val_loss: 0.7341 - val_acc: 0.6318
Epoch 11/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.9430 - ac
c: 0.5335 - val_loss: 0.8003 - val_acc: 0.5670
Epoch 12/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.9347 - ac
c: 0.5379 - val_loss: 0.7309 - val_acc: 0.6359
Epoch 13/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.9200 - ac
c: 0.5227 - val_loss: 0.7332 - val_acc: 0.6098
Epoch 14/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.9309 - ac
c: 0.5340 - val_loss: 0.7454 - val_acc: 0.6295
Epoch 15/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.9035 - ac
c: 0.5449 - val_loss: 0.7333 - val_acc: 0.6054
Epoch 16/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.9123 - ac
c: 0.5457 - val_loss: 0.7667 - val_acc: 0.6026
Epoch 17/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.8871 - ac
c: 0.5530 - val_loss: 0.8375 - val_acc: 0.5803
Epoch 18/30
7352/7352 [==============================] - 62s 8ms/step - loss: 0.8839 - ac
c: 0.5558 - val_loss: 0.7235 - val_acc: 0.5765
Epoch 19/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.8874 - ac
```

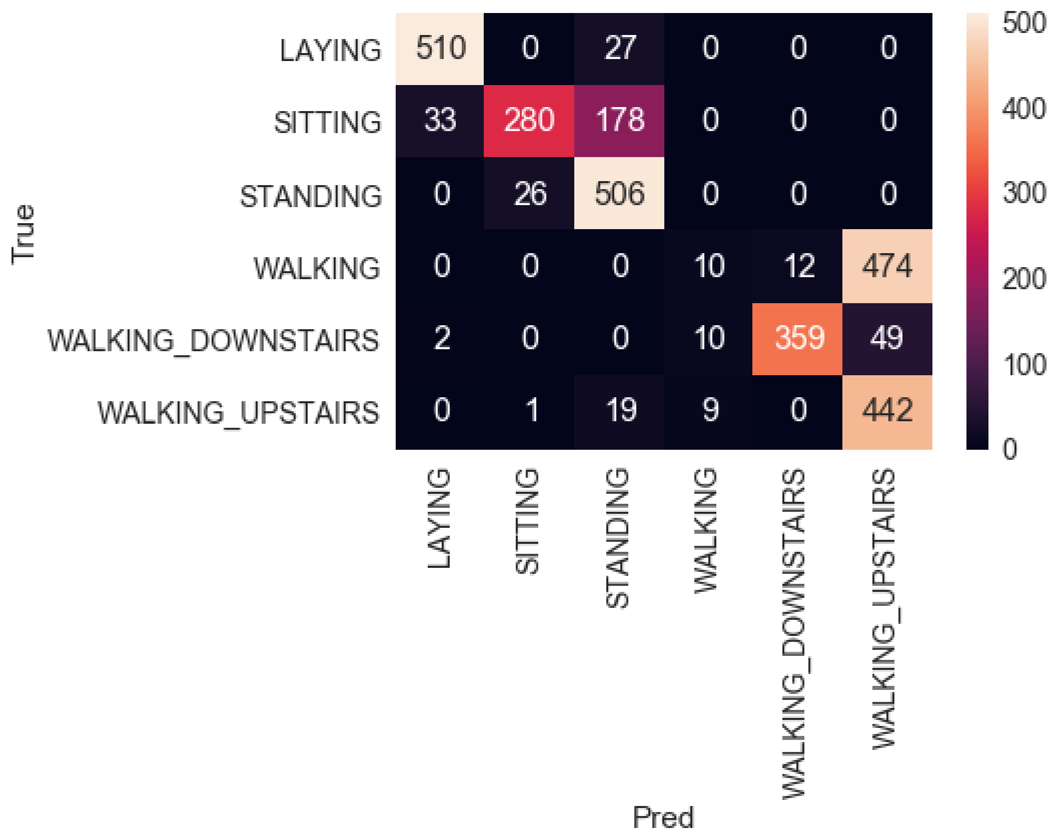
```
c: 0.5525 - val_loss: 0.7223 - val_acc: 0.6983
Epoch 20/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.8741 - ac
c: 0.5584 - val_loss: 0.7824 - val_acc: 0.7031
Epoch 21/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.8759 - ac
c: 0.5579 - val_loss: 0.7830 - val_acc: 0.6916
Epoch 22/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.9489 - ac
c: 0.5483 - val_loss: 0.7994 - val_acc: 0.6722
Epoch 23/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.8600 - ac
c: 0.5558 - val_loss: 0.8048 - val_acc: 0.7004
Epoch 24/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.8719 - ac
c: 0.5662 - val_loss: 0.7584 - val_acc: 0.6966
Epoch 25/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.8587 - ac
c: 0.5650 - val_loss: 0.7950 - val_acc: 0.7017
Epoch 26/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.8586 - ac
c: 0.5624 - val_loss: 0.8027 - val_acc: 0.7204
Epoch 27/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.8799 - ac
c: 0.5626 - val_loss: 0.7732 - val_acc: 0.7228
Epoch 28/30
7352/7352 [==============================] - 64s 9ms/step - loss: 0.9881 - ac
c: 0.5643 - val_loss: 0.7890 - val_acc: 0.6980
Epoch 29/30
7352/7352 [==============================] - 63s 9ms/step - loss: 0.8236 - ac
c: 0.5771 - val_loss: 0.8037 - val_acc: 0.7448
Epoch 30/30
7352/7352 [==============================] - 65s 9ms/step - loss: 0.8113 - ac
c: 0.5797 - val_loss: 0.8678 - val_acc: 0.7150
```



In [42]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4b2930780>



In [43]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 4s 1ms/step
```

In [44]:
```
score
```

Out[44]: [0.8678421316043295, 0.7149643705463183]

- With a 3 layer architecture we got 71.49% accuracy and a loss of 0.86

(C) 3 layer with Dropout rate 0.7

In [45]:
```
# Initiliazing the sequential model
model = Sequential()
# Configuring the parameters
model.add(LSTM(32, input_shape=(timesteps, input_dim),return_sequences=True))
model.add(LSTM(64))
# Adding a dropout layer
model.add(Dropout(0.7))
# Adding a dense output layer with sigmoid activation
model.add(Dense(n_classes, activation='sigmoid'))
model.summary()
```

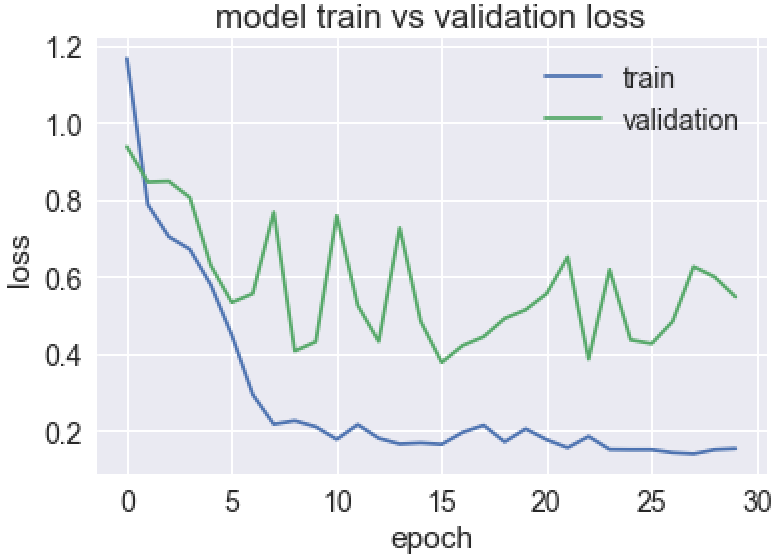
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
lstm_8 (LSTM)                (None, 64)                24832
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 390
=================================================================
Total params: 30,598
Trainable params: 30,598
Non-trainable params: 0
_________________________________________________________________
```

In [46]:
```
# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [47]:
```
# Training the model
history = model.fit(X_train,
          Y_train,
          batch_size=batch_size,
          validation_data=(X_test, Y_test),
          epochs=epochs)
# plot train and validation loss
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('model train vs validation loss')
pyplot.ylabel('loss')
pyplot.xlabel('epoch')
pyplot.legend(['train', 'validation'], loc='upper right')
pyplot.show()
```

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/30
7352/7352 [==============================] - 61s 8ms/step - loss: 1.1660 - ac
c: 0.5048 - val_loss: 0.9372 - val_acc: 0.6159
Epoch 2/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.7870 - ac
c: 0.6345 - val_loss: 0.8455 - val_acc: 0.6172
Epoch 3/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.7038 - ac
c: 0.6598 - val_loss: 0.8476 - val_acc: 0.6145
Epoch 4/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.6714 - ac
c: 0.6772 - val_loss: 0.8056 - val_acc: 0.6491
Epoch 5/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.5790 - ac
c: 0.7550 - val_loss: 0.6294 - val_acc: 0.7489
Epoch 6/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.4487 - ac
c: 0.8546 - val_loss: 0.5334 - val_acc: 0.8419
Epoch 7/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.2945 - ac
c: 0.9094 - val_loss: 0.5563 - val_acc: 0.8554
Epoch 8/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.2175 - ac
c: 0.9285 - val_loss: 0.7674 - val_acc: 0.8368
Epoch 9/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.2267 - ac
c: 0.9336 - val_loss: 0.4074 - val_acc: 0.8996
Epoch 10/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.2110 - ac
c: 0.9376 - val_loss: 0.4314 - val_acc: 0.9002
Epoch 11/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1784 - ac
c: 0.9408 - val_loss: 0.7585 - val_acc: 0.8836
Epoch 12/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.2166 - ac
c: 0.9316 - val_loss: 0.5260 - val_acc: 0.8901
Epoch 13/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.1813 - ac
c: 0.9408 - val_loss: 0.4323 - val_acc: 0.9111
Epoch 14/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1664 - ac
c: 0.9444 - val_loss: 0.7267 - val_acc: 0.8768
Epoch 15/30
7352/7352 [==============================] - 60s 8ms/step - loss: 0.1693 - ac
c: 0.9463 - val_loss: 0.4845 - val_acc: 0.9111
Epoch 16/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1655 - ac
c: 0.9425 - val_loss: 0.3777 - val_acc: 0.8965
Epoch 17/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1966 - ac
c: 0.9411 - val_loss: 0.4217 - val_acc: 0.9067
Epoch 18/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.2149 - ac
c: 0.9365 - val_loss: 0.4447 - val_acc: 0.9006
Epoch 19/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1719 - ac
```

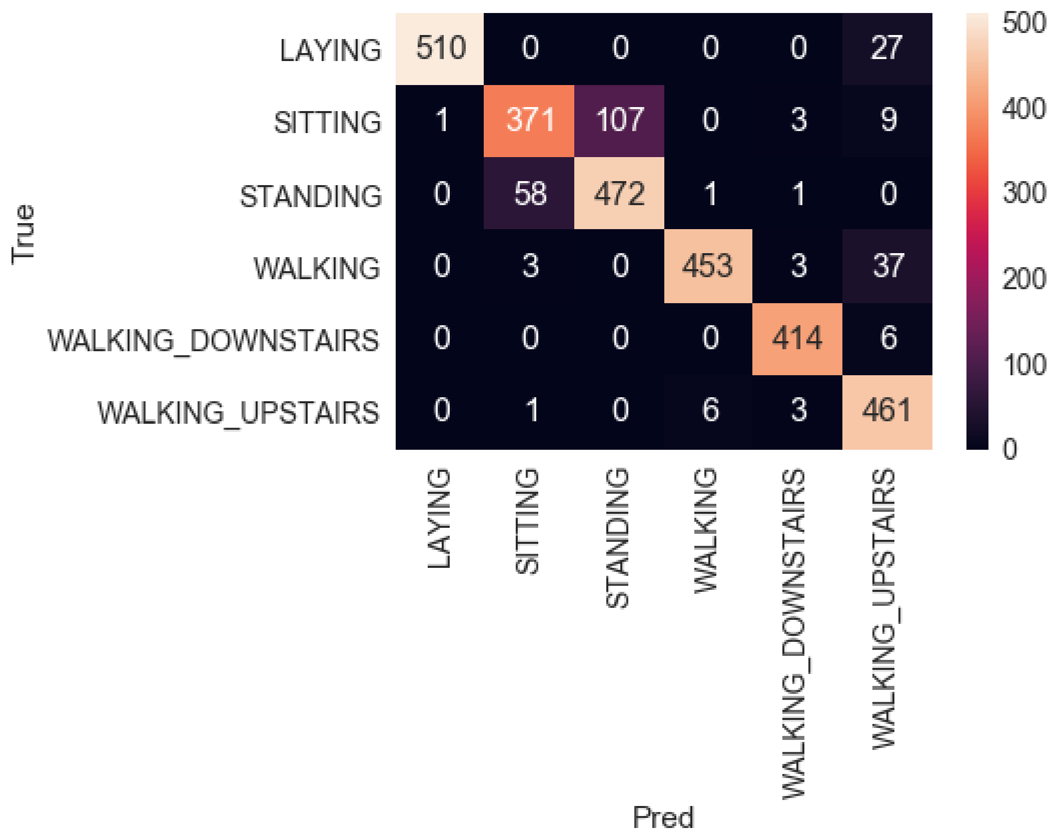
```
c: 0.9430 - val_loss: 0.4919 - val_acc: 0.8972
Epoch 20/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.2056 - ac
c: 0.9402 - val_loss: 0.5143 - val_acc: 0.9097
Epoch 21/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1775 - ac
c: 0.9445 - val_loss: 0.5563 - val_acc: 0.8962
Epoch 22/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1566 - ac
c: 0.9509 - val_loss: 0.6512 - val_acc: 0.8955
Epoch 23/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1860 - ac
c: 0.9440 - val_loss: 0.3863 - val_acc: 0.9128
Epoch 24/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1523 - ac
c: 0.9472 - val_loss: 0.6184 - val_acc: 0.8938
Epoch 25/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1517 - ac
c: 0.9482 - val_loss: 0.4366 - val_acc: 0.9050
Epoch 26/30
7352/7352 [==============================] - 61s 8ms/step - loss: 0.1518 - ac
c: 0.9463 - val_loss: 0.4265 - val_acc: 0.8965
Epoch 27/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1440 - ac
c: 0.9489 - val_loss: 0.4840 - val_acc: 0.8975
Epoch 28/30
7352/7352 [==============================] - 59s 8ms/step - loss: 0.1408 - ac
c: 0.9512 - val_loss: 0.6260 - val_acc: 0.8904
Epoch 29/30
7352/7352 [==============================] - 57s 8ms/step - loss: 0.1518 - ac
c: 0.9508 - val_loss: 0.5996 - val_acc: 0.9016
Epoch 30/30
7352/7352 [==============================] - 58s 8ms/step - loss: 0.1546 - ac
c: 0.9479 - val_loss: 0.5479 - val_acc: 0.9097
```



In [48]:
```
# Confusion Matrix
#print(confusion_matrix(Y_test, model.predict(X_test)))

confusion = pd.DataFrame(confusion_matrix(Y_test, model.predict(X_test)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4b5f71160>



In [49]:
```
score = model.evaluate(X_test, Y_test)
```

```
2947/2947 [==============================] - 3s 981us/step
```

In [50]:
```
score
```

Out[50]: [0.5478853281966495, 0.9097387173396675]

- With a 3 layer architecture we got 90.97% accuracy and a loss of 0.54

conclusions

layer	n_hidden	dropout rate	accuracy	loss
2 layer	64	0.8	89.68%	0.553
2 layer	64	0.7	92.60%	0.427
2 layer	32	0.5	90.93%	0.335
3 layer	64,32	0.8	90.49%	0.318
3 layer	64,32	0.9	71.49%	0.867
3 layer	32,64	0.7	90.97%	0.547

- this data contain 6 classes which has raw data.
- on raw data we applied multiple architecture of LSTM for tuning hyperparameter.
- we applied Dropout for avoid overfit.
- as we see that 3 layer architecture with dropout 0.8 give accuracy of 90.49% and loss of 0.318.
- for 2 layer architecture with dropout 0.7 give accuracy of 92.60% and loss of 0.427.