รายงาน

a. Experimental Setup: รายละเอียดการแบ่งข้อมูลสำหรับการ Train และ Dev รวมถึง Label Distribution

ทางผู้จัดทำการตรวจสอบดูว่า Label ของ Dataset ซึ่งคือคอลัมน์ aspectCategory และ Sentiment มีกี่ประเภทและมีการกระจายตัวอย่างไร พร้อมแปลงแต่ละ Label ใน aspectCategory และ Sentiment โดยใช้การ Replace ด้วย Dictionary ดังนี้

```
eva_train["polarity"] = eva_train["polarity"].replace({'0':'neutral',
                                                       '1':'negative',
                                                       '2':'positive',
                                                       '3':'conflict'})
```

ภาพ 1 แปลงค่าในคอลัมน์ polarity

```
df["aspectCategory"] = df["aspectCategory"].replace({'service': 0,
                                                     'food': 1,
                                                     'anecdotes/miscellaneous':2,
                                                     'price':3,
                                                     'ambience':4})
```

ภาพ 2 แปลงค่าในคอลัมน์ aspectCategory

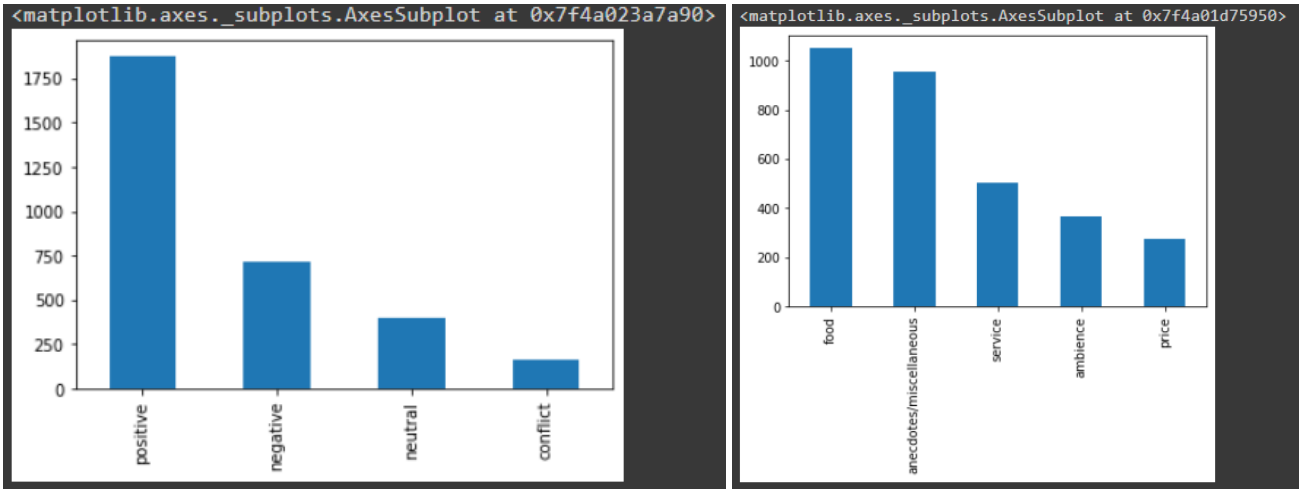
ในส่วนของการแบ่งข้อมูล ทางผู้จัดทำทำการแบ่งข้อมูลเป็นสัดส่วนด้วยอัตราส่วน 80 : 20 ซึ่งเป็น Train และ Dev ตามลำดับ โดยแบ่งข้อมูลโดยใช้ Function ชื่อ Stratified Split ซึ่งจะแบ่งข้อมูลให้มีการกระจายตัวเหมือนกับข้อมูลก่อนที่จะถูกแบ่ง ซึ่งท้ายที่สุดแล้วจะมีข้อมูล Test ของทางอาจารย์ และทางผู้จัดทำจะนำข้อมูล Train Dataset ทั้งหมดมา Training Model

```
def StratifiedSplit(dat,label_name,split_ratio):
  total_row=dat.count()[0]
  labels=dat[label_name].unique()
  ratio_dict={}
  for i in labels:
    ratio_dict[i]=dat.where(dat[label_name]==i).dropna().count()[0]/total_row
  num_sample=int(total_row*split_ratio)
  test_df=pd.DataFrame(columns=dat.columns)
  train_df=pd.DataFrame(columns=dat.columns)
  for i in labels:
    num_sample_label =int(num_sample*ratio_dict[i])
    dat_sm=dat.where(dat[label_name]==i).dropna()
    dat_sm=dat_sm.sample(frac=1)
    msk = np.random.rand(dat_sm.count()[0]) < (num_sample_label/dat_sm.count()[0])
    test_df=test_df.append(dat_sm[msk])
    train_df=train_df.append(dat_sm[~msk])
  return (train_df, test_df)
```

```
train, test=StratifiedSplit(df,"polarity",0.2)
```

```
train, test=StratifiedSplit(df,"aspectCategory",0.2)
```

ภาพ 3 การแบ่งข้อมูล Train และ Dev



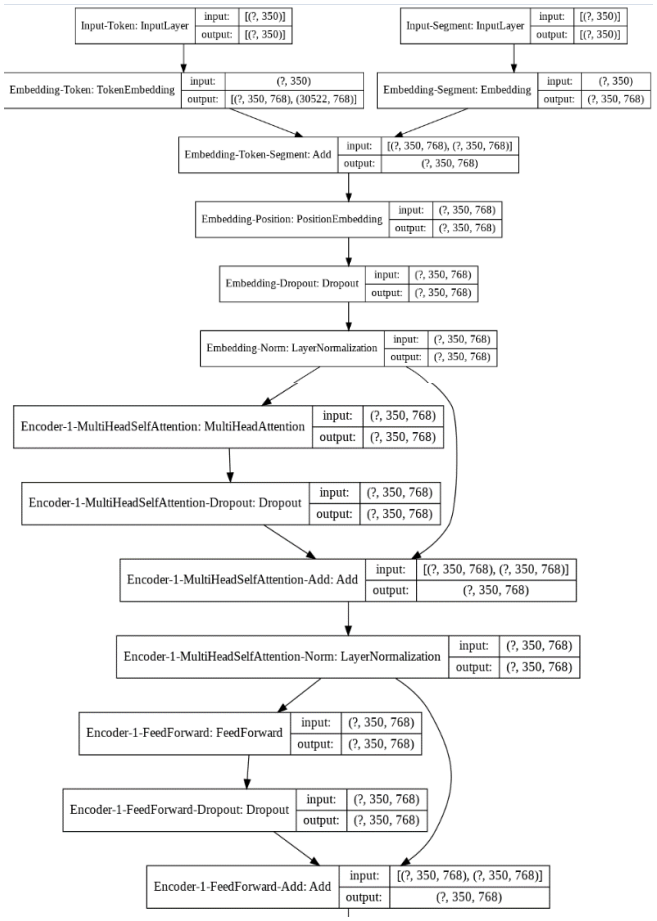
ภาพ 4 Count plot ของ Sentiment และ aspectCategory ตามลำดับ

b. Model: โมเดลที่ได้ทดสอบมาทั้งหมดที่ได้ผลดีและได้ผลไม่ดีและต้องพูดถึงรายละเอียดของ Hyperparameter เช่น ใช้ feature อะไรบ้าง หรือใช้ hidden layer กี่ชั้น ชั้นละกี่ unit หรือ embedding ถูก pretrain มาจากไหนก่อนหรือไม่เพราะอะไรจึงคิดว่าโมเดลเหล่านี้จึงจะมีประสิทธิภาพที่ดี.

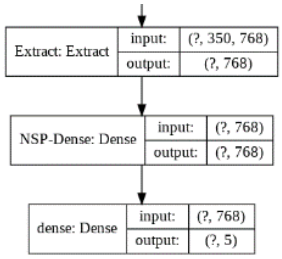
ทางผู้จัดทำทำ Word Embedding โดยใช้ BERT ซึ่งมีการ pre-train มาเรียบร้อบแล้ว

```
downloading pretrained BERT model (uncased_L-12_H-768_A-12.zip)...
[                                                            ]
extracting pretrained BERT model...
done.
```

Model ที่ผู้จัดทำใช้ทั้งหมดได้แก่ CNN และ BERT-CNN Input เป็น textโดยที่ BERT จะมีการทำ tokenize ด้านในและต่อมาจะทำการ encode ทำทั้งหมด 12 รอบแล้วจึงส่งเข้า Dense Layer ตามปกติ คิดว่า BERT-CNN จะมีประสิทธิภาพที่ดีเนื่องจาก BERT มีคลังคำใหญ่มากๆซึ่งจะมากกว่า word embedding จาก wikipedia2vec



(เหมือนส่วนencodeอีก11รอบ)




```
=================================================
Total params: 109,361,669
Trainable params: 109,361,669
Non-trainable params: 0
```

c. Results: ตารางเปรียบเทียบผลการทดลองรวมทั้งข้อสรุปที่ได้และเหตุผลประกอบ

BERT-CNN:

```
=== CLASSIFICATION : ASPECT ===
                class name  precision  recall  F1-score support
0                     food      0.965   0.770     0.857    1051
1                    price      0.950   0.695     0.803     275
2                  service      0.962   0.747     0.841     506
3                ambience       0.948   0.644     0.767     368
4  anecdotes/miscellaneous      0.939   0.888     0.913     954
5                MACRO AVG      0.953   0.749     0.836    3154
6                MICRO AVG      0.953   0.781     0.858    3154

=== CLASSIFICATION : SENTIMENT ===
  class name  precision  recall  F1-score support
0   positive      0.941   0.932     0.936    1546
1   negative      0.853   0.835     0.844     635
2    neutral      0.821   0.698     0.754     387
3   conflict      0.696   0.441     0.540     161
4  MACRO AVG      0.828   0.726     0.769    2729
5  MICRO AVG      0.895   0.847     0.870    2729

=== CLASSIFICATION : OVERALL ===
             precision  recall  F1-score support
0  MICRO AVG      0.839   0.687     0.755    3154
```

CNN: ใช้ word embedding จาก wikipedia2vec.github.io เนื่องจากว่า train จนครบทุก epoch แล้วได้ค่า accuracy 0.65 ใน polarity และ accuracy 0.34 ใน aspectCategory จึงไม่เลือกใช้การทำ word embedding ประเภทนี้เพราะให้ประสิทธิภาพแย่

d. Conclusion: สรุปว่าโมเดลอะไรที่ใช้ในการเอาไปใช้กับ test set บน gradescope

ทางผู้จัดทำใช้ Convolutional Neural Network เพราะว่าให้ประสิทธิภาพที่ดีกว่าทุกโมเดลที่ทางผู้จัดทำลองทดสอบ