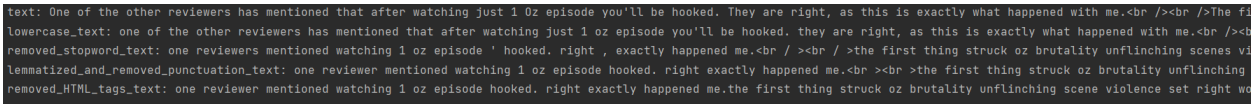
1. Preprocessing method:
   I. Turn to lowercase
   II. Remove stopwords
   III. Apply Lemmatization
   IV. Remove punctuation
   V. Remove HTML Tags

Preprocessing method 效果如下



   VI. 討論:

   雖然處理方法眾多，但在測試後，發現只有 Turn to lowercase 有較好的效果，其餘方法反而造成 language model 有更差的表現，在使用 IMDB Dataset 與運用 Bigram+Sklearn.GaussianNB model 的詳細數據如下表(feature_num=500)。

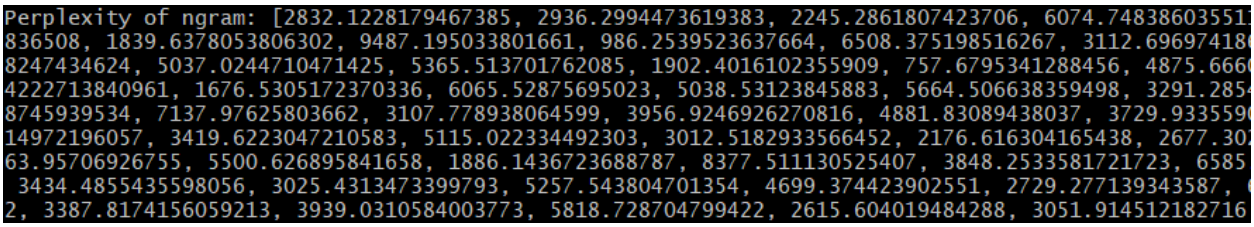
	F1-Score	Precision	Recall
Without Preprocessing	0.7057	0.7088	0.7065
Only turn to lowercase	0.7094	0.7133	0.7104
Lowercase+Lemmatization	0.7079	0.7119	0.7089
Lowercase+Remove Punctuation	0.7033	0.7094	0.7049
Lowercase+Remove HTML Tags	0.7094	0.7027	0.7102
Lowercase+Remove Stopwords	0.7094	0.7133	0.7104

2. Tuning parameter ‘feature_num’ (default=500)

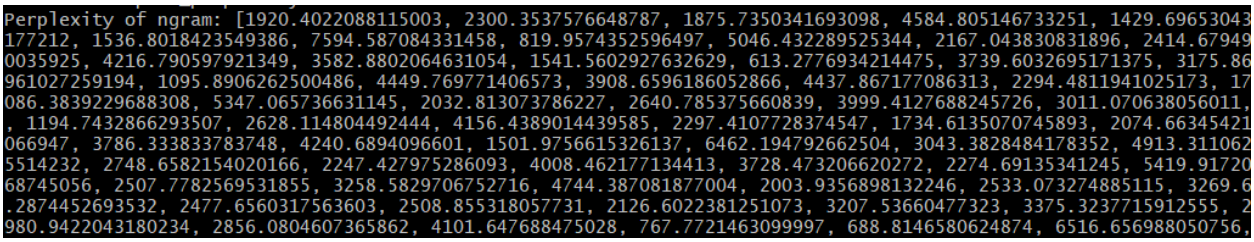
討論: 在有 Preprocessing 的情況下，調整 feature_num 大小的數據如下表，可以看見 feature_num 越高，模型的表現越好，但同時花費的時間也提升不少。

	F1-Score	Precision	Recall
feature_num=100	0.6116	0.6206	0.6156
feature_num=300	0.6823	0.6864	0.6835
feature_num=500	0.7094	0.7133	0.7104
feature_num=700	0.7191	0.7238	0.7202
feature_num=900	0.7395	0.7435	0.7403
feature_num=1500	0.7637	0.7647	0.7639

3. Discuss about using Bigram and Sklearn GaussianNB:
   I. Perplexity:
   II. 在沒有 preprocess 的情況下，部分 corpus 的 perplexity 如下圖。



而在有 preprocess 的情況下，部分 corpus 的 perplexity 如下圖，可發現沒有 preprocess 時的 perplexity 比有 preprocess 時的 perplexity 還大。



   III. F1-Score, Precision, Recall 在 Preprocessing 與否的比較如下表(feature_num=500)

	F1-Score	Precision	Recall
Without Preprocessing	0.7057	0.7088	0.7065
With Preprocessing	0.7094	0.7133	0.7104

   IV. Encounter zero division error:

   運用 Laplace Smoothing 解決，避免計算時分母為零。

4. Bigram and DistilBert
   I. Bigram:

   運用 Markov 假設，當前這個詞僅僅與前一個詞相關，機率如下圖。

   - Bigram Model：

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

   DistilBert:

   較為簡易的 BERT，保留了 97%的語言理解能力，但同時減少空間大小 40%且加速 60%。它是一種深度雙向、無監督、且僅使用純文字進行 Pretrain 的模型，它疊加多層 Transformer Encoder，可以更好地理解語意，並在各 NLP 項目中被不斷使用。

II. Bigram 為當前這個詞僅僅與前一個詞相關，但 BERT 則是利用 Self-Attention 機制，讓訊息長度過長時仍不易丟失訊息，在這方面上可看出 BERT 的表現較 Bigram 好。詳細表現數據如下圖，可發現 DistilBert 的表現確實比 Bigram 好。

	F1-Score	Precision	Recall
Bigram without peprocessing	0.7057	0.7088	0.7065
Bigram with preprocessing	0.7094	0.7133	0.7104
DistilBert without preprocessing	0.9329	0.9333	0.9329
DistilBert with preprocessing	0.9333	0.9331	0.9333

```
ddeng@LAPTOP-DLIGRF6B MINGW64 /c/thomas/NYCU_Courses/Second_Semester/AI/HW2
$ python main.py --model_type BERT --preprocess 0 --part 2
Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilBertMod
ias', 'vocab_projector.weight', 'vocab_projector.bias']
- This IS expected if you are initializing DistilBertModel from the checkpoint of a model trained on another
tForPreTraining model).
- This IS NOT expected if you are initializing DistilBertModel from the checkpoint of a model that you expect
assification model).
second_part
100%|##########| 5000/5000 [11:59:29<00:00,  8.63s/it]
100%|##########| 1/1 [12:35:13<00:00, 45313.67s/it]it]
Epoch: 0, F1 score: 0.9329, Precision: 0.933, Recall: 0.9329, Loss: 0.2285
end from second_part
ddeng@LAPTOP-DLIGRF6B MINGW64 /c/thomas/NYCU_Courses/Second_Semester/AI/HW2
$ python main.py --model_type BERT --preprocess 1 --part 2
Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilBertModel: ['vocab_laye
ias', 'vocab_projector.weight', 'vocab_transform.bias']
- This IS expected if you are initializing DistilBertModel from the checkpoint of a model trained on another task or with anc
tForPreTraining model).
- This IS NOT expected if you are initializing DistilBertModel from the checkpoint of a model that you expect to be exactly i
assification model).
second_part
100%|##########| 5000/5000 [11:43:58<00:00,  8.45s/it]
100%|##########| 1/1 [12:19:36<00:00, 44376.39s/it]it]
Epoch: 0, F1 score: 0.933, Precision: 0.9331, Recall: 0.933, Loss: 0.2297
end from second_part
```

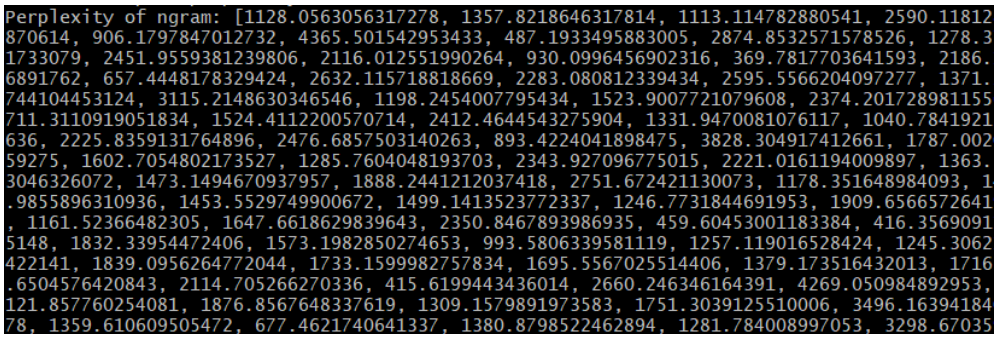
III. Preprocessing method 會增進 Bigram 模型的表現，因為這會萃取出一個句子中重要的詞彙、刪除多餘標點符號、詞型還原等等，這些都能對模型解語意有一定的幫助。

IV. Convert words as [UNK]:

A. Convert all words that appeared less than 10 times as [UNK]: Perplexity 如下圖。

```
Perplexity of ngram: [603.5730264555942, 724.5841506276375, 586.3906653245771, 1278.48
6123, 473.89539514732826, 2177.7158748389575, 269.4816784103672, 1437.8424693736156, 6
26582806, 1252.2735717012981, 1113.7477146274666, 509.97111280749243, 204.897485145005
8830748541894, 358.9994027775824, 1383.9896608058611, 1168.589916637615, 1373.55781306
.127960298605, 1588.0555617296168, 633.7151286559109, 815.6158892444593, 1255.7783914
0796869303726, 775.8774682503217, 1262.5734904860597, 682.9732143785474, 563.367087531
67.6709674935425, 1307.6491963813403, 486.39696384664165, 2036.5249206679448, 925.9953
95, 831.7645716587524, 646.4355948335516, 1214.3774879634552, 1146.2771849004062, 734.
870077, 783.8783141487954, 969.6990514999237, 1402.5083900045004, 612.7917466502346, 7
4989449, 755.6748515462572, 800.7220976351971, 657.5478104284218, 1021.0970282787832,
00460952, 834.4849539423797, 1181.553034558805, 248.89809497925407, 232.696837884671006
24328618594, 830.8480815988617, 530.758211122687, 675.7015089274429, 654.207067791924
858975560554, 898.0936759408656, 919.8431818773006, 736.9341290479138, 891.0852360031
002092345808, 228.50579648394412, 1377.3052340655042, 2181.9267416241537, 891.7567206
12.6358407214939, 680.9513436576775, 890.2526476259181, 1762.2827866212444, 823.60474
364.5335759907654, 697.7929476676472, 646.9593824805971, 1660.0727485812022, 567.70454
25, 585.9819168277257, 1211.0341386177413, 653.0659499670928, 971.147882311628, 993.75
```

B. Converts the words that appeared just once as [UNK]: Perplexity 如下圖。



C. 討論:

在 A.情況中的 preplexity 減少許多，在情況 B.中的 perplexity 減少幅度較小，而表現結果如下表，可看見在情況 A.與 B.下，f1-score 反而較原本的低，但幅度過小因此也難以就此定論。

	F1-Score	Precision	Recall
Bigram without replacing as [UNK]	0.7094	0.7133	0.7104
Convert all words that appeared less than 10 times as [UNK]	0.7046	0.7085	0.7056
Converts the words that appeared just once as [UNK]	0.7069	0.7112	0.7080

5. 這次作業中總共遇到了幾個問題，首先是發現未經過 preprocess 時的 f1-score 比經過 preprocess 時高，且經過 preprocess 時的 perplexity 比未經過 preprocess 時高，因此在試驗過後，留下對本次 Sentiment analysis 有幫助的 preprocess method，成功提升 f1-score。接下來遇到的問題是使用 Bigram 時無論有沒有經過 preprocess 都發現 f1-score=0.3333，因此推論 bug 出現在 train_sentiment 中，最後找到 bug 並改善。