1. Data preprocessing analysis

What adjustment have you done to the Chest X-ray Images or the Reports? (20 pts)

Have you performed any additional operations to filter out noise or extract key information? (10 pts)

針對report進行tokenize、對image套用強化細節的filter後再進行normalize

```
image= image.filter(ImageFilter.DETAIL)
```

```
image= image.filter(ImageFilter.DETAIL)############################################################ additional operation: image filter
image_array = np.array(image).astype(np.float32)############################################### additional operation: normalize
normalized_array = (image_array - np.min(image_array)) / (np.max(image_array) - np.min(image_array))
image = Image.fromarray((normalized_array * 255).astype(np.uint8))
```

2. Model & Training Method (20 pts)

What models do you choose? Why? (Please introduce the visual model and language model respectively.) (10 pts)

Encoder使用的是被肺炎X光照數據集finetune過的Vit(vision transformer)(sample code使用的模型)

使用這個模型的原因是因為考量到任務，使用類似資料訓練過的模型可以降低訓練成本。

Decoder 則是Bert模型(case sensitive)

會使用case sensitive版本是因為考量到report的格式，如果模型能考慮大小會比較好。

6/20報告老師有提到bert無法生成，實際上我們有考慮過這個問題，所以有深挖過huggingface的doucumentation，我們發現無論是vision encoder decoder 還是正常的 bert model from pretrained又或是openai gpt2 from pretrained這些huggingface提供的框架其實都有generation的method可以呼叫，所以我們推測實際上應該是當初模型的開發團隊跟huggingface有說好這些基於transformer框架的模型都要有text generation的能力，至於bert的開發團隊是如何攻克bert本身的限制，我們因為無法確認bert的實作方法無法繼續深挖下去。但是可以確定huggingface提供的bert的確可以進行text generation。

Your evaluation scores during training. (10 pts)

	Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.027778	0.010676	0.010676	0.002941	0.000735	0.000735
2	0.146022	0.041770	0.041770	0.055788	0.012775	0.012775
3	0.232774	0.057831	0.057831	0.121260	0.023490	0.023490
4	0.279716	0.070337	0.070337	0.147523	0.030518	0.030518
5	0.311127	0.077427	0.077427	0.159299	0.032998	0.032998
6	0.337742	0.084072	0.084072	0.180374	0.037441	0.037441
7	0.351566	0.086682	0.086682	0.186943	0.038436	0.038436
8	0.368109	0.088393	0.088393	0.203983	0.040665	0.040665
9	0.378132	0.090362	0.090362	0.211262	0.042102	0.042102
10	0.388797	0.092355	0.092355	0.221808	0.044221	0.044221
11	0.395034	0.093375	0.093375	0.224713	0.044988	0.044988
12	0.399402	0.093799	0.093799	0.231017	0.045925	0.045925
13	0.408016	0.095141	0.095141	0.238397	0.046893	0.046893
14	0.419593	0.098470	0.098470	0.251592	0.050137	0.050137
15	0.427838	0.101969	0.101969	0.259569	0.053104	0.053104
16	0.430701	0.102590	0.102590	0.258738	0.053071	0.053071
17	0.433625	0.104526	0.104526	0.260634	0.054504	0.054504
18	0.439215	0.105494	0.105494	0.267627	0.055857	0.055857
19	0.441884	0.106055	0.106055	0.270056	0.056427	0.056427
20	0.447298	0.107402	0.107402	0.276213	0.057779	0.057779
21	0.453943	0.109847	0.109847	0.283462	0.060082	0.060082
22	0.455024	0.110265	0.110265	0.285825	0.060518	0.060518
23	0.460906	0.112459	0.112459	0.292671	0.062622	0.062622
24	0.464440	0.113026	0.113026	0.297704	0.063423	0.063423
25	0.465417	0.113163	0.113163	0.299311	0.063627	0.063627
26	0.467730	0.113780	0.113780	0.302046	0.064109	0.064109
27	0.470797	0.114194	0.114194	0.305904	0.064774	0.064774
28	0.472873	0.114989	0.114989	0.308768	0.065564	0.065564
29	0.475055	0.115732	0.115732	0.310746	0.066088	0.066088

3. Analysis (30 pts)

Have you encountered any difficulties? How did you address them? Unlimited (10 pts)

問題一:

一開始本來打算使用LoRA來微調，但是hugging face 的 LoRA method不支援 vision encoder decoder model。

```
File c:\Users\amtom\AppData\Local\Programs\Python\Python310\lib\site-packages\peft\tuners\lora\model.py:4
    410 if peft_config.target_modules is None:
    411     if model_config["model_type"] not in TRANSFORMERS_MODELS_TO_LORA_TARGET_MODULES_MAPPING:
--> 412         raise ValueError("Please specify `target_modules` in `peft_config`")
    413     peft_config.target_modules = set(
    414         TRANSFORMERS_MODELS_TO_LORA_TARGET_MODULES_MAPPING[model_config["model_type"]]
    415     )
    416 return peft_config

ValueError: Please specify `target_modules` in `peft_config`
```

問題二:

在將tokenizer換成bert專用的tokenizer時因為bos token的id設定不好(bos_token_id = 627)導致生出來的text多了”##”

['##Chest PA view shows : Impression : - Suspicious pulmonary ed']

['##Chest PA view : Impression : - Increased both lung markings.']

['##Chest AP view showed : 1. s / p sternotomy and cardiac']

Solution: 將bos_token_id設定成400404

問題三:

在前處理的階段因為使用的API是pillow，所以最好透過pillow內建的工具完成想做的前處理

Solution:

```
image= image.filter(ImageFilter.DETAIL)############################################################ a
image_array = np.array(image).astype(np.float32)#################################################### a
normalized_array = (image_array - np.min(image_array)) / (np.max(image_array) - np.min(image_array))
image = Image.fromarray((normalized_array * 255).astype(np.uint8))
```

問題四:

```
ValueError: `decoder_start_token_id` or `bos_token_id` has to be defined for encoder-decoder generation.
```

Vision encoder decoder model在generate時需要設定bos token id(sample code 沒有這個問題)

Solution: 研究過後發現vision encoder decoder model的config有兩種:

model.config 、model.generation_config。在這裡是因為generation config的部分沒有設定所以出問題。

```
model.config.decoder_start_token_id = tokenizer.bos_token_id
model.generation_config.decoder_start_token_id = tokenizer.bos_token_id
```

備註:也許openai的gpt2有預先設置這個參數，bert沒有所以出現了這個問題

Sample code的rouge socre這麼慘也有可能是因為這樣，因為model.config跟model.generation_config的參數不一樣，導致loss有在降，但是eval還是不好。

為了驗證這個的想法，在sample code中加入對generation_config的設置。

result:

	rouge-L-P	rouge-L-R	rouge-L-F	rouge-2-P	rouge-2-R	rouge-2-F
29	0.298129	0.061448	0.061448	0.111938	0.018391	0.018391

sample code:

rouge-L-P	rouge-L-R	rouge-L-F	rouge-2-P	rouge-2-R	rouge-2-F
0.275969,	0.058301,	0.058301,	0.098075,	0.016161	, 0.016161

可以看到結果有稍微變好。

問題五:Rouge分數再提高

在前面的部分中，我們已經讓rouge分數提高了許多，在ROUGE-L-P，以及ROUGE-2-P都取得了蠻不錯的分數，但是在剩下的幾個分數都偏低，而這有可能是因為，生成最大長度限定在20的關係，實際上所對到的LCS以及二元組其實都沒有到非常多，僅是因為分母小所以被拉高了，這點也可以在後面的ROUGE-L-R和ROUGE-2-R中得到證實，因此我們針對這一部分進行了調整，首先我們將max_length更改為30，而得到的結果如下

0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.137887	0.058369	0.058369	0.048333	0.018173	0.018173
2	0.233927	0.088581	0.088581	0.102361	0.033813	0.033813
3	0.271286	0.107523	0.107523	0.110484	0.041731	0.041731
4	0.306906	0.118008	0.118008	0.134714	0.048276	0.048276
5	0.326278	0.123911	0.123911	0.141946	0.049827	0.049827
6	0.355888	0.131337	0.131337	0.171320	0.057569	0.057569
7	0.370949	0.137964	0.137964	0.182959	0.062889	0.062889
8	0.380044	0.140658	0.140658	0.194532	0.066261	0.066261
9	0.389008	0.143808	0.143808	0.202030	0.069534	0.069534
10	0.396299	0.146230	0.146230	0.213960	0.073129	0.073129
11	0.408550	0.150790	0.150790	0.226870	0.077416	0.077416
12	0.409598	0.151730	0.151730	0.226763	0.078402	0.078402
13	0.419418	0.155408	0.155408	0.237320	0.082143	0.082143
14	0.424038	0.157628	0.157628	0.242750	0.084425	0.084425
15	0.429806	0.160499	0.160499	0.246006	0.086287	0.086287
16	0.434402	0.162070	0.162070	0.251024	0.087676	0.087676
17	0.439395	0.163236	0.163236	0.256613	0.088982	0.088982
18	0.441656	0.164083	0.164083	0.260677	0.090506	0.090506
19	0.443518	0.166055	0.166055	0.264065	0.092716	0.092716
20	0.447863	0.168503	0.168503	0.269392	0.095211	0.095211
21	0.449714	0.170001	0.170001	0.272909	0.097688	0.097688
22	0.449289	0.169196	0.169196	0.273043	0.097291	0.097291
23	0.449437	0.168586	0.168586	0.274217	0.097047	0.097047
24	0.452381	0.169266	0.169266	0.277602	0.097746	0.097746
25	0.455206	0.171098	0.171098	0.281026	0.099545	0.099545
26	0.458695	0.173267	0.173267	0.284638	0.101441	0.101441
27	0.458843	0.172826	0.172826	0.284913	0.101071	0.101071
28	0.458901	0.172357	0.172357	0.285769	0.100814	0.100814
29	0.460535	0.173205	0.173205	0.287888	0.101775	0.101775

雖然在準確率的方面有略微下降，但是後面的召回率不管是ROUGE-L還是ROUGE-2都有明顯的上升，表示實際上的結果有所改善，而在這之後，我們又嘗

試了在訓練時以及實際測試時不同長度的作法，分別是訓練時max_length=30，測試時max_length=40，還有訓練時max_length=30，測試時max_length=40，而結果分別如下

	Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0	0.140741	0.099577	0.099577	0.069231	0.052124	0.052124
1	0.249008	0.154022	0.154022	0.114744	0.074308	0.074308
2	0.283686	0.170183	0.170183	0.127778	0.079790	0.079790
3	0.322513	0.188785	0.188785	0.164790	0.096702	0.096702
4	0.258011	0.151028	0.151028	0.131832	0.077362	0.077362
5	0.255902	0.149161	0.149161	0.128077	0.075228	0.075228
6	0.273941	0.154840	0.154840	0.138499	0.078444	0.078444
7	0.239699	0.135485	0.135485	0.121186	0.068638	0.068638
8	0.213065	0.120431	0.120431	0.107721	0.061012	0.061012
9	0.191759	0.108388	0.108388	0.096949	0.054911	0.054911
10	0.189021	0.106813	0.106813	0.096427	0.054626	0.054626
11	0.173270	0.097912	0.097912	0.088391	0.050074	0.050074
12	0.159941	0.090380	0.090380	0.081592	0.046222	0.046222
13	0.165051	0.088757	0.088757	0.085158	0.045413	0.045413
14	0.156226	0.083917	0.083917	0.079771	0.042524	0.042524
15	0.146461	0.078673	0.078673	0.074785	0.039867	0.039867
16	0.139892	0.074986	0.074986	0.071455	0.038002	0.038002
17	0.146667	0.079479	0.079479	0.077723	0.041691	0.041691
18	0.159692	0.086260	0.086260	0.086766	0.046287	0.046287
19	0.164229	0.087657	0.087657	0.087500	0.046216	0.046216
20	0.178692	0.094539	0.094539	0.097983	0.051257	0.051257
21	0.170570	0.090242	0.090242	0.093529	0.048927	0.048927
22	0.178025	0.094250	0.094250	0.099358	0.051921	0.051921
23	0.183846	0.097653	0.097653	0.101910	0.053837	0.053837
24	0.194804	0.101780	0.101780	0.110118	0.056731	0.056731
25	0.200951	0.103940	0.103940	0.114208	0.057952	0.057952
26	0.209678	0.108320	0.108320	0.120979	0.061209	0.061209
27	0.216431	0.111931	0.111931	0.125490	0.063690	0.063690
28	0.226969	0.116894	0.116894	0.133746	0.067444	0.067444
29	0.219403	0.112998	0.112998	0.129288	0.065196	0.065196

0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.138700	0.048346	0.048346	0.074769	0.023359	0.023359
3	0.209569	0.073318	0.073318	0.113080	0.035978	0.035978
4	0.252386	0.093435	0.093435	0.123244	0.043111	0.043111
5	0.271651	0.101497	0.101497	0.125252	0.045036	0.045036
6	0.295495	0.109804	0.109804	0.142811	0.051126	0.051126
7	0.300511	0.113410	0.113410	0.145313	0.053251	0.053251
8	0.312244	0.121214	0.121214	0.151940	0.058370	0.058370
9	0.332132	0.130746	0.130746	0.165075	0.064877	0.064877
10	0.339259	0.134221	0.134221	0.167240	0.065868	0.065868
11	0.345336	0.136830	0.136830	0.170453	0.067041	0.067041
12	0.342595	0.136100	0.136100	0.168981	0.066611	0.066611
13	0.354234	0.140982	0.140982	0.178301	0.070629	0.070629
14	0.360897	0.144172	0.144172	0.185395	0.073884	0.073884
15	0.365997	0.145977	0.145977	0.190964	0.075584	0.075584
16	0.375185	0.149910	0.149910	0.199622	0.079090	0.079090
17	0.381501	0.152648	0.152648	0.207331	0.081957	0.081957
18	0.390060	0.156291	0.156291	0.216976	0.085617	0.085617
19	0.397308	0.159543	0.159543	0.223848	0.088502	0.088502
20	0.404631	0.162224	0.162224	0.230034	0.090776	0.090776
21	0.413096	0.166773	0.166773	0.239967	0.096240	0.096240
22	0.419002	0.169234	0.169234	0.245008	0.098420	0.098420
23	0.426400	0.173270	0.173270	0.253866	0.103269	0.103269
24	0.431416	0.175818	0.175818	0.258259	0.106165	0.106165
25	0.435742	0.177945	0.177945	0.263456	0.108316	0.108316
26	0.439848	0.179989	0.179989	0.267919	0.110247	0.110247
27	0.441979	0.181665	0.181665	0.269651	0.111949	0.111949
28	0.446450	0.183567	0.183567	0.274578	0.113810	0.113810
29	0.448640	0.184311	0.184311	0.275800	0.114179	0.114179

而上述兩種的結果，在第一種時雖然前面分數有較快的成長，但是隨後分數卻下降許多，甚至在最後的結果相較原本max_length=20時差了不少，但在第二種情況，表現甚至接近均設定為30時，而這可能是因為，訓練時雖然只設定了長度30，但後面測試時變為40，不僅將訓練的部分完整用出，甚至後面多的一段也多少有對中一部分內容，且模型在生成時也未必要達到最大長度，而前期分數進步較

慢也是因為還沒把控好長度，相反的，訓練時比測試時設定的最大長度還要長，可能會導致部分訓練的重點內容沒有辦法表現出來，進而導致分數下降，也有可能是輸出最後段的內容表現較好，被截掉導致分數下降，使得雖然後面有再回升，但也無法達到跟之前相同程度

而在這之後，我們又分別將max_length設為50 100 150並進行了測試，而得到的結果如下

2	0.100315	0.062338	0.062338	0.063899	0.037804	0.037804
3	0.133633	0.085379	0.085379	0.077562	0.046476	0.046476
4	0.188816	0.121235	0.121235	0.112274	0.067966	0.067966
5	0.221124	0.140290	0.140290	0.131314	0.079863	0.079863
6	0.244117	0.151312	0.151312	0.142601	0.084512	0.084512
7	0.259799	0.163172	0.163172	0.149953	0.090124	0.090124
8	0.273919	0.169595	0.169595	0.159173	0.094302	0.094302
9	0.284679	0.175423	0.175423	0.165400	0.097789	0.097789
10	0.296882	0.179910	0.179910	0.174550	0.101225	0.101225
11	0.302460	0.185494	0.185494	0.174533	0.102313	0.102313
12	0.303391	0.186322	0.186322	0.170489	0.100982	0.100982
13	0.307730	0.188086	0.188086	0.171548	0.101627	0.101627
14	0.310993	0.189339	0.189339	0.172329	0.101830	0.101830
15	0.320015	0.194491	0.194491	0.181369	0.106671	0.106671
16	0.325747	0.196125	0.196125	0.185086	0.107782	0.107782
17	0.333041	0.198664	0.198664	0.192592	0.110541	0.110541
18	0.335300	0.199586	0.199586	0.194777	0.111407	0.111407
19	0.339235	0.201294	0.201294	0.197254	0.112506	0.112506
20	0.343422	0.203525	0.203525	0.200670	0.114346	0.114346
21	0.347469	0.205731	0.205731	0.203268	0.115959	0.115959
22	0.354978	0.210122	0.210122	0.209416	0.119665	0.119665
23	0.358139	0.212003	0.212003	0.212524	0.121432	0.121432
24	0.359315	0.211957	0.211957	0.214473	0.121857	0.121857
25	0.363660	0.214803	0.214803	0.217637	0.123919	0.123919
26	0.369186	0.216863	0.216863	0.222823	0.125954	0.125954
27	0.370498	0.217645	0.217645	0.223812	0.126666	0.126666
28	0.373241	0.219227	0.219227	0.226390	0.128203	0.128203
29	0.374496	0.219305	0.219305	0.227986	0.128596	0.128596

	Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.076471	0.084384	0.084384	0.036980	0.038188	0.038188
3	0.116153	0.137362	0.137362	0.060555	0.069650	0.069650
4	0.146011	0.167390	0.167390	0.075101	0.083861	0.083861
5	0.165988	0.183030	0.183030	0.083622	0.089256	0.089256
6	0.190250	0.199816	0.199816	0.100797	0.100932	0.100932
7	0.202408	0.207379	0.207379	0.108528	0.105012	0.105012
8	0.210945	0.217672	0.217672	0.113037	0.112265	0.112265
9	0.220641	0.224434	0.224434	0.121851	0.119165	0.119165
10	0.234633	0.235443	0.235443	0.132984	0.128838	0.128838
11	0.239264	0.241311	0.241311	0.135815	0.133587	0.133587
12	0.246305	0.247643	0.247643	0.140940	0.138854	0.138854
13	0.256297	0.252282	0.252282	0.148629	0.143261	0.143261
14	0.265078	0.258260	0.258260	0.155945	0.148913	0.148913
15	0.274442	0.260719	0.260719	0.164662	0.152803	0.152803
16	0.281010	0.266621	0.266621	0.169721	0.157512	0.157512
17	0.289040	0.273440	0.273440	0.176464	0.163509	0.163509
18	0.295242	0.277714	0.277714	0.181332	0.167066	0.167066
19	0.298213	0.278972	0.278972	0.183681	0.167991	0.167991
20	0.305484	0.284919	0.284919	0.190010	0.173270	0.173270
21	0.310265	0.287816	0.287816	0.192510	0.174956	0.174956
22	0.316149	0.291703	0.291703	0.198230	0.179038	0.179038
23	0.323496	0.297959	0.297959	0.205013	0.184954	0.184954
24	0.327252	0.299511	0.299511	0.207877	0.186381	0.186381
25	0.332524	0.303435	0.303435	0.212857	0.190390	0.190390
26	0.337756	0.307600	0.307600	0.218261	0.194858	0.194858
27	0.341925	0.309904	0.309904	0.222266	0.197231	0.197231
28	0.347409	0.313896	0.313896	0.227186	0.201190	0.201190
29	0.352481	0.316449	0.316449	0.232246	0.204335	0.204335

0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.103148	0.099179	0.099179	0.052758	0.048537	0.048537
2	0.129961	0.159300	0.159300	0.064709	0.078252	0.078252
3	0.143391	0.202200	0.202200	0.074281	0.104417	0.104417
4	0.159072	0.214939	0.214939	0.084453	0.112445	0.112445
5	0.169195	0.225117	0.225117	0.089386	0.116565	0.116565
6	0.188176	0.235724	0.235724	0.101579	0.123841	0.123841
7	0.197948	0.239252	0.239252	0.108241	0.127374	0.127374
8	0.209039	0.243732	0.243732	0.114472	0.129955	0.129955
9	0.214056	0.249372	0.249372	0.116933	0.132543	0.132543
10	0.219735	0.258395	0.258395	0.122431	0.141042	0.141042
11	0.230493	0.258684	0.258684	0.129088	0.141582	0.141582
12	0.237989	0.258003	0.258003	0.130779	0.140444	0.140444
13	0.247570	0.258701	0.258701	0.138565	0.142824	0.142824
14	0.254719	0.263765	0.263765	0.145769	0.148278	0.148278
15	0.269860	0.270336	0.270336	0.159249	0.155453	0.155453
16	0.274895	0.271918	0.271918	0.162846	0.157446	0.157446
17	0.281332	0.275630	0.275630	0.167478	0.161346	0.161346
18	0.284885	0.276606	0.276606	0.169776	0.162061	0.162061
19	0.288437	0.280934	0.280934	0.172972	0.165748	0.165748
20	0.291530	0.281927	0.281927	0.174636	0.166567	0.166567
21	0.293403	0.281903	0.281903	0.175541	0.166525	0.166525
22	0.297428	0.283977	0.283977	0.179646	0.169044	0.169044
23	0.299023	0.284051	0.284051	0.180785	0.169734	0.169734
24	0.301777	0.285727	0.285727	0.182983	0.171473	0.171473
25	0.303243	0.286426	0.286426	0.182881	0.171425	0.171425
26	0.304160	0.286838	0.286838	0.183080	0.171591	0.171591
27	0.306834	0.288815	0.288815	0.185061	0.173162	0.173162
28	0.310971	0.291979	0.291979	0.188750	0.176536	0.176536
29	0.313554	0.295290	0.295290	0.190988	0.179625	0.179625

在上面的三種結果中，我們可以看到max_length在100時表現是全部裡面最好的，而max length從30到50，以及50到100時，都可以看到最後結果的改善，但是到了150時，卻反而下降，而且我們也可以看到，在max_length為100時，不管是在rouge-L，還是rouge-2的分數，準確率跟召回率都非常的接近，表示這時候可能生成的長度也比較接近，而到了150時，雖然也相當接近，但分數卻都較100時低，表示max_length到了一定的程度之後，都會自己去偏向和原報告接近的字數，但表現上卻不一定會更好，超過一個程度反而會下降，而還有一點可看的是，進行訓練所需要的時間，在max_length是30的時候，訓練僅需11分鐘，但到了100時，訓練需要23分鐘，而150時更到了27分鐘，表示max_length愈長所需的訓練時間也會越長，在緊緊訓練了30epoch的情況下就有了明顯的差距，若需要進行

大量訓練，如何權衡分數和時間也是一個課題

備註: Rouge test 的程式碼保有在 rouge_score_test.ipynb中、其生成解結果保有在 rouge_score_test_generated_output.csv中

Proposal:

胸腔食療助手: 根據你的胸腔X光照提供飲食建議的幫手。

implementation 方向1：使用gpt2 API

嘗試使用gpt2，將report輸入到模型，並且給予一個prompt，請gpt2扮演醫生的角色生成飲食建議，但是最後輸出的結果並不好，準確率很低，常常回答得不知所以然。(我們認為是GPT2本身能力有限)

- prompt

```
    prompt = (
        f"You are a doctor reviewing the following chest X-ray report: '{report_text}'. "
        "Based on this report, provide one sentence of dietary advice to the patient to suppo
rt their lung health."
    )
```

- gpt2生成出來的結果(勉強可以的)

```
Recommend clinical correlation. '. Based on this report, provide one sentence of dietary a
dvice to the patient to support their lung health. Include any dietary supplements or othe
r treatments that their treating physician has recommended. A single sentence of dietary a
```

- 不準確的輸出

```
What is the most important factor that affects hemoglobin? What are the signs and symptoms
of anemia?

>**_Answer:_** _Anemia occurs when the body does not have enough hemoglobin to carry oxyge
n into the bloodstream to the cells._
>

P.3. The following
```

原本想使用gpt3模型，但是gpt3模型需要付費，所以選擇用ChatGPT手動生成建議，再自行做訓練。

(方向1: 失敗)

implementation 方向2：利用GPT生成dataset再比照sample code方法訓練

ChatGPT

dataset : 將助教提供的dataset的report給chatGPT生成飲食建議

給ChatGPT的提示詞:

you are a doctor. now you are going to give advice to a patient based on provided chest x-ray report i am going to provide you. please conduct your opinion in one sentence. your advice should focus on diet.

(有刻意避免summarize的字樣防止結果不夠口語化)

訓練方法-方向2:

與sample code一致。

model: 與先前提到的一致: vision encoder decoder

encoder: vit 、deocder: bert

結果:

跑了evaluation dataset中的圖片後:

```
['Based on your chest X - ray report, I recommend adopting a nutrient - dense diet that includes plenty o
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray report, I recommend following a heart - healthy and anti - inflammatory die
['Based on your chest X - ray report, I recommend maintaining a balanced diet rich in fruits, vegetables,
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray findings, consider a diet rich in antioxidants, omega - 3 fatty acids, and
['Based on your chest X - ray report, I recommend following a heart - healthy and anti - inflammatory die
```

會這樣的原因是因為一開始的report有相當一部分是健康的人的，因此他們的report也差不多結果是我們最後做好的dataset有一部分長得差不多，才有這樣的結果。

備註: Proposal保有在proposal.ipynb中、

其dataset保有在data/proposal_train & data/proposal_valid中

生成結果保有在proposal_generated_result.csv中

rouge score:

	Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.050306	0.039638	0.039638	0.027319	0.023533	0.023533
2	0.208053	0.205477	0.205477	0.143922	0.146075	0.146075
3	0.296460	0.285270	0.285270	0.207806	0.203399	0.203399
4	0.367655	0.353918	0.353918	0.268596	0.261422	0.261422
5	0.416747	0.397763	0.397763	0.309179	0.297458	0.297458
6	0.437209	0.422120	0.422120	0.320633	0.311948	0.311948
7	0.457801	0.443727	0.443727	0.335788	0.328047	0.328047
8	0.471896	0.461090	0.461090	0.345468	0.340221	0.340221
9	0.483980	0.474441	0.474441	0.355627	0.351358	0.351358
10	0.490182	0.478233	0.478233	0.360211	0.354378	0.354378
11	0.500580	0.489440	0.489440	0.368592	0.363476	0.363476
12	0.505790	0.494994	0.494994	0.372191	0.368016	0.368016
13	0.507287	0.491704	0.491704	0.371138	0.363787	0.363787
14	0.509641	0.491559	0.491559	0.372661	0.363423	0.363423
15	0.513704	0.494066	0.494066	0.375899	0.365467	0.365467
16	0.515908	0.495422	0.495422	0.376911	0.365929	0.365929
17	0.517619	0.497297	0.497297	0.378443	0.367712	0.367712
18	0.519025	0.497451	0.497451	0.378866	0.367384	0.367384
19	0.518194	0.495612	0.495612	0.377682	0.365341	0.365341
20	0.515829	0.491761	0.491761	0.374852	0.361302	0.361302
21	0.520529	0.491486	0.491486	0.378185	0.361269	0.361269
22	0.520676	0.491700	0.491700	0.378441	0.361718	0.361718
23	0.524165	0.493496	0.493496	0.381149	0.363129	0.363129
24	0.523176	0.491276	0.491276	0.379718	0.360749	0.360749
25	0.524532	0.491747	0.491747	0.381292	0.361630	0.361630
26	0.524868	0.492235	0.492235	0.381130	0.361729	0.361729
27	0.524828	0.493285	0.493285	0.381053	0.362621	0.362621
28	0.526210	0.494750	0.494750	0.382300	0.363957	0.363957
29	0.526709	0.494674	0.494674	0.382363	0.363683	0.363683

分數會變成這樣主要是因為有特別去調整長度讓output跟target text長度差距不要過大。除此之外還有因為GPT給的建議開頭也長的相似，另外還有一部分的飲食建議其實是相似的所導致。

備註: 這裡rouge target的text來源於助教提供的validation dataset report經過上面我們生proposal需要的dataset一樣的流程生出來的(一樣的提示詞)。