HW1
Section: 001
Adel Alkhamisy
=====Dev Accuracy=====
Accuracy: 696 / 872 = 0.798165

Part 1:
All the ToDo lists are done, and the model achieved 0.79 accuracy on the development dataset which is considered a good score according to the requirements of Homework 1.

=====Train Accuracy=====
Accuracy: 6450 / 6920 = 0.932081
Precision (fraction of predicted positives that are correct): 3217 / 3294 = 0.976624; Recall (fraction of true positives predicted correctly): 3217 / 3610 = 0.891136; F1 (harmonic mean of precision and recall): 0.931924
=====Dev Accuracy=====
Accuracy: 696 / 872 = 0.798165
Precision (fraction of predicted positives that are correct): 323 / 378 = 0.854497; Recall (fraction of true positives predicted correctly): 323 / 444 = 0.727477; F1 (harmonic mean of precision and recall): 0.785888
Time for training and evaluation: 86.47 seconds

Part 2:

1. **Vector dimensions**

Table 1 (Time with seconds)

		Training set				Development Set				
Embedding size	Hidden size	Acc	Prec	Recall	F1	Acc	Prec	Recall	F1	Time
10	10	0.93	0.94	0.92	0.93	0.77	0.79	0.73	0.76	46.56
50	50	0.98	0.98	0.98	0.98	0.80	0.78	0.83	0.81	49.33
100	100	0.98	0.98	0.97	0.98	0.79	0.82	0.75	0.79	52.74
50	300	0.97	0.97	0.97	0.97	0.77	0.79	0.75	0.77	50.19
100	300	0.99	0.99	0.98	0.99	0.78	0.80	0.76	0.78	53.41
300	300	0.93	0.97	0.89	0.93	0.79	0.85	0.72	0.78	86.47
500	300	0.96	0.98	0.94	0.96	0.77	0.76	0.81	0.78	84.61
700	300	0.94	0.99	0.90	0.94	0.79	0.81	0.76	0.78	99.32
1000	300	0.95	0.99	0.91	0.95	0.79	0.81	0.77	0.79	121.33
300	50	0.92	0.96	0.88	0.92	0.79	0.82	0.76	0.78	67.25
300	100	0.99	0.99	0.99	0.99	0.77	0.74	0.83	0.78	67.21
300	500	0.96	0.98	0.94	0.96	0.79	0.80	0.77	0.78	68.79
300	700	0.92	0.98	0.85	0.91	0.79	0.85	0.71	0.77	70.13
300	1000	0.92	0.98	0.86	0.92	0.79	0.83	0.73	0.78	71.42
1000	1000	0.97	0.99	0.95	0.97	0.79	0.80	0.77	0.79	130.03
50	1000	0.97	0.97	0.98	0.98	0.76	0.79	0.73	0.76	51.29

In general, all the scores in table 1 are considered good scores according to Homework requirements. The accuracy fluctuates between 0.80-0.77 and F1 score, which considers a better metric compared to accuracy, fluctuated between 0.81 and 0.76. The model achieved the best score with embedding size 50 and hidden size 50. The reason for that if the embedding size and hidden size are too small, the model cannot capture all the important hidden features between words. moreover, a large number of the embedding size and hidden size induce the model to learn unnecessary hidden features between the words which reduces accuracy and F1 score. Another important aspect to note is that the running time increases proportionally with increasing size of embedding size and hidden size. In large ML tasks, the running time is a crucial factor. Thus, it is important to select the appropriate sizes for those hyper-parameters

2. Optimization methods

Table 2 (only embedding size 300, Hidden size 300 are used)

		Avg loss on Epoch										
Optimizer/Learning rate	F1	0	1	2	3	4	5	6	7	8	9	Time
Adam/0.1	0.63	1.60	0.72	1.36	4.64	0.69	0.69	0.69	0.69	0.69	0.69	95.38
Adam/0.01	0.79	0.63	0.32	0.13	0.07	0.04	0.02	0.01	0.01	0.01	0.01	88.37
Adam/0.001	0.78	0.68	0.60	0.42	0.27	0.19	0.13	0.10	0.07	0.05	0.05	69.73
Adam/0.0001	0.73	0.69	0.68	0.68	0.67	0.67	0.65	0.64	0.62	0.59	0.57	87.31
Adam/0.00001	0.68	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.68	0.68	0.68	89.96
SGD/0.1	0.66	0.69	0.69	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	65.10
SGD/0.01	0.48	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.68	64.89
SGD/0.001	0.67	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	66.10
SGD/0.0001	0.66	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	66.28
SGD/0.00001	0.66	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	65.17

F1 score on development Set

There is no better optimization method for all problems. However, in general, the Adaptive learning rate optimizers are considered the best amongst optimizers in many problems because they can converge fast with optimal model parameters. Regarding the learning rate, choosing a small rate is considered better than a large learning rate and it can be described as the step size of the optimizer. For example, in Table 2 the average loss rate in the case of Adam with a 0.1 learning rate fluctuated greatly in each epoch because the step size for the optimizer is large; thus, it could not achieve optimal values to update the model parameters. Whereas, in the case of Adam with a 0.01 learning rate, the loss average decreased smoothly from 0.68 in the first epoch to 0.05 in the last epoch. The reason behind this behavior is that the optimizer takes a small step and updates the parameters in each mini-batch in the case of Adam. However, in SGD, the optimizer updates the model parameters after passing over all the dataset and this can explain the steadiness of the average loss rate between epochs. However, too small a rate prevents the optimizer from converging. For example, in the case of Adam with 0.00001, the optimizer did not achieve optimal model parameters because the accuracy was 0.68 compared to 0.78. In conclusion, a small learning rate a better, but we have to be very careful to not choose an extreme small rate.

3. Embedding initialization

I have used GloVe pre-trained word embeddings and I have made the word embedding weight unlearnable by setting the weight.requires_grad = False because the word embeddings are already trained on a huge dataset. Also, I have copied word vectors from GloVe to an embedding_matrix if the words are found in my vocab list only; otherwise, embedding_matrix is filled with zeros except for the unknown words 'UNK' representation. To learn a good representation for unknown words, I computed the average of all the word embeddings in embedding_matrix and use it as a representation for the unknown words because the average of all embeddings is considered a rare occurrence; therefore, it is a good candidate to represent the unknown words in my vocab list. Embedding dimension 300 and hidden units 300 are used. The model performance is almost the same if compared with random initialized word embedding vectors except for one notable result which is the variation between the training and development set performance. The best performance of randomly initialized word embedding is achieved at an accuracy of 0.98 and 0.80 on the training and development sets respectively and the difference between them is 0.18. However, the difference between training and development sets accuracy in the case of using pre-trained word embedding is 0.82-0.78=0.04 which is a very small number, the same observation applies to F1 score. Thus, pre-trained word embedding could be generalized on unseen data better than the randomly initialized word embedding which is learnable. The reason for that is the Glove pre-trained word embedding has been trained on a large dataset and captures more hidden features between words than the randomly initialized word embedding which learned features only from our small-scale dataset.

=====Train Accuracy=====
Accuracy: 5710 / 6920 = 0.825145
Precision (fraction of predicted positives that are correct): 2983 / 3566 = 0.836511; Recall (fraction of true positives predicted correctly): 2983 / 3610 = 0.826316; F1 (harmonic mean of precision and recall): 0.831382
=====Dev Accuracy=====
Accuracy: 688 / 872 = 0.788991
Precision (fraction of predicted positives that are correct): 366 / 472 = 0.775424; Recall (fraction of true positives predicted correctly): 366 / 444 = 0.824324; F1 (harmonic mean of precision and recall): 0.799127
Time for training and evaluation: 106.14 seconds

References:

1. https://pytorch.org
2. https://www.analyticsvidhya.com/blog/2021/06/part-8-step-by-step-guide-to-master-nlp-useful-natural-language-processing-tasks/
3. https://towardsdatascience.com/deep-learning-for-nlp-with-pytorch-and-torchtext-4f92d69052f
4. https://nlp.stanford.edu/projects/glove/
5. https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d
6. Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization (2015), arxiv