Adrien Cogny – Data Science in The Wild – Assignment 4 – Writeup

1. Bag of Words

The initial phase was collecting all the files and information required to run the models that would then be created. This was done by fetching the assignment files from cmsx, github and slack and congregating them in the git repository so they could be used together to run the various models. In order to check that all these steps were done, the Bag Of Words model provided was used.

2. CNN Creation

A Convolutional Neural Net was then created in order to run on pre-trained GloVE and Word2Vec embeddings. The batch size was set as 64 as this was a good size batch which the GPU could handle with relative ease, this was also done so that there is more iterations over the data. This Convolutional Neural Net was configured to have kernels of size of 3, 4 and 5 by default, although this was experimented with in part 3.

a. CNN with glove

The first run of the CNN with non-modified sentences was done using the GloVe word embeddings. Batch sizes were set to 64 and evaluation was done every 4000 steps. After 40 000 steps, the validation error was found to be 0.44. A new run was done with 60 000 steps and reached a better validation error of 0.4299. A further run with 150 000 steps was performed and reached a validation error of 0.413.

b. CNN with word2vec

A new CNN was trained, this time with the Google word2vec embeddings. For batches of size 64 and an evaluation done every 4000 steps. After 40 000 steps, the evaluation error was 0.45 (this batch size was set to 32 by mistake). After 60 000 steps, the evaluation error was 0.44 (with the correct batch size of 64). After 150 000 steps, the evaluation error was 0.425.

c. CNN with glove and word2vec

Training and predicting using 2 sets of word embeddings proved a little trickier than first thought as it required concatenating embeddings corresponding to the same word. This was done by first finding any common vocabulary between the two embeddings. Then for each common word, the two embeddings were found and concatenated. That is the first embedding (length 300) and the second embedding (length 300) were concatenated to create a single new embedding vector of length 600.

This CNN was then run with batches of size 64 and with an evaluation done every 4000 steps. After 40 000 steps of training, the validation error was 0.46. After 60 000 steps of data, the validation error was 0.457. After 150 000 steps, the validation error was 0.457.

d. Conclusions Basic CNN

The main conclusion to draw from the basic CNN is that, with the same initializations (batch size, step number and evaluation frequency), the GloVe embeddings seem to generate a better model faster. From the log of each run, a graph was generated to show the relationship between step number and validation error. It can clearly be seen that the 3 curves seem to plateau after around 150 000 steps.

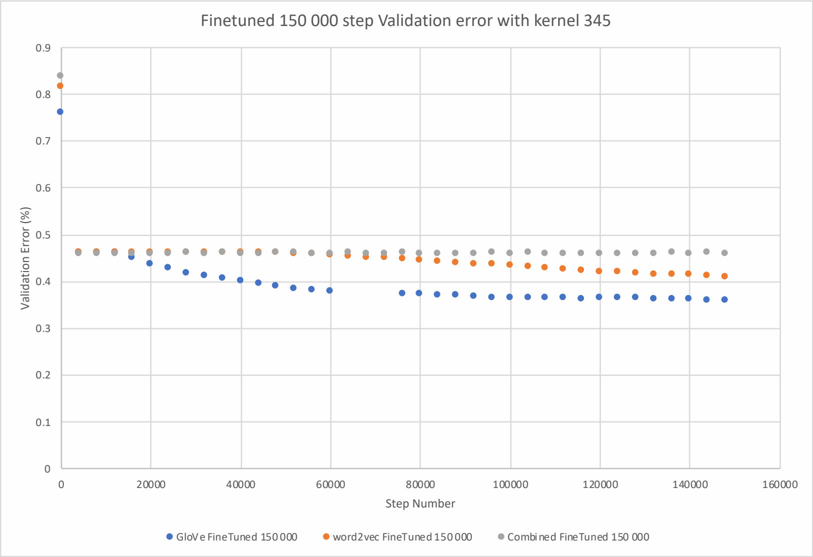


Figure 1 shows the validation error as a function of step for each of the three non-fine-tuned models created.

2. CNN Tunning.

The tuned CNN was done by setting the is_static flag to false at first. Changing the kernel size was then experimented. The selected model with the best compromise between validation error and step number was then selected to run for generating the testing data (this is the model that is present in the assign4.py file). The run_test method was then applied to this model in order to generate the predictions found in results.txt.

a. Fine-tuned CNN with glove

The fine tuning was first explored on a CNN with glove embeddings. This lead to a validation error of 0.399 after 40 000 steps. After 60 000 steps, the validation error was 0.378. After 150 000 steps, the validation error was 0.358.

b. Fine-tuned CNN with word2vec

The same fine tuning was then applied to the word2vec embeddings on the CNN. The 40 000 step validation error was 0.46. After 60 000 steps, the validation error was 0.454 After 150 000 steps, the validation error was 0.4088.

c. Fine-tuned CNN with glove and word2vec

The same fine tuning was then applied to the combined embeddings on the CNN. The 40 000 step validation error was 0.46. After 60 000 steps, the validation error was 0.458. After 150 000 steps, the validation error was 0.4584.

d. Changing the kernel

The kernel size was changed to 2,3,4 and the validation errors did not change from the ones using the 3,4,5 kernel. The kernel size was then changed to 4,5,6 and once again, the validation error did not change. The kernel size was then changed to 2,2,2 and the validation errors did not change either. This strongly suggests that the kernel size does not change the validation errors in the model and thus does not improve or worsen the models.

e. Dropout changes

When the kernel sizes were determined not to have an effect on the validation errors, the dropout rates were modified to see whether they could further improve the model. The rates tested on all fine-tuned models with 150 000 steps were 0.5 (standard), 0.6, 0.75 and 0.8. The dropout changes did not have any real effect on the validation errors of the various models.

f. Further training to 200 000 steps

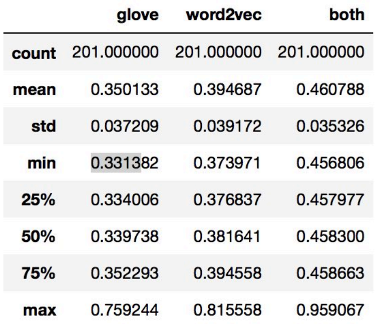
It seemed that there was a plateau that was reachable for the validation error but the 150 000 steps did not seem to reach it and it seemed that there could be improvements made. Because none of the previously tried tuning solutions worked, it was determined that training for longer than 150 000 steps may be interesting. Thus, all three models were trained to 200 000 steps. The validation error for the glove embeddings model lowered to 0.352 which was a small but notable change. The validation error for the word2vec embeddings model lowered to 0.394. The validation error for the model that combined both embeddings was lowered to 0.45.

g. Further Training to 400 000 steps

After 400 000 steps, an even better validation error was reached for both the glove and word2vec. The combined approach seemed to have plateaued however. The best validation error for glove was 0.339, for word2vec was 0.381 and for the combination of the two was 0.457.

h. Further Training to 800 000 steps

In order to try and reach the plateau on the word2vec and glove embeddings, the models were run for 800 000 steps. The table below show the statistics for this run.



	glove	word2vec	both
count	201.000000	201.000000	201.000000
mean	0.350133	0.394687	0.460788
std	0.037209	0.039172	0.035326
min	0.331382	0.373971	0.456806
25%	0.334006	0.376837	0.457977
50%	0.339738	0.381641	0.458300
75%	0.352293	0.394558	0.458663
max	0.759244	0.815558	0.959067

Figure 2 shows the description of the data for the validation error during the 800 000 step run.

This figure shows that for the glove embeddings, the validation error reached a plateau of about 0.331. The word2vec embedding reached 0.37 and the combination reached 0.457.

i. Conclusions Fine-tuning

The following graph shows the progression of the validation error rates for each of the fine-tuned classifiers with the standard 3,4,5 kernel size. Changing the kernel to a lower number with multiple sizes (2,3,4) did not seem to affect the results of the fine-tuned CNN.

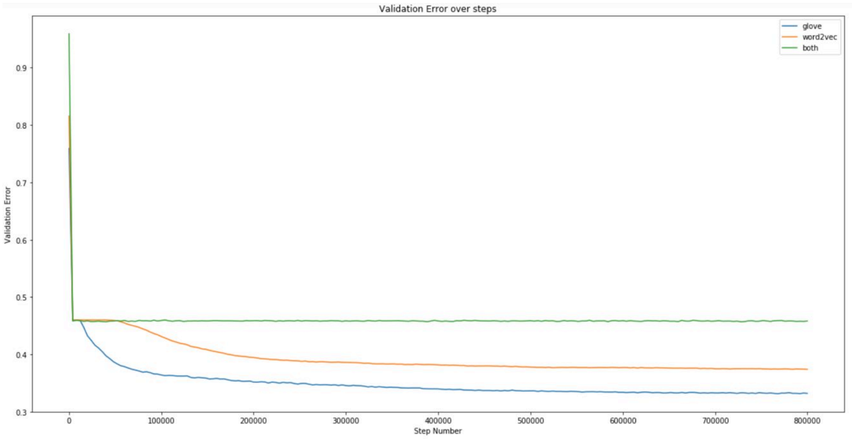


Figure 3 shows the validation error as it progresses through the number of steps for the three fine-tuned CNNs

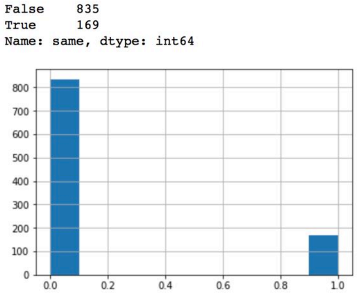
This clearly shows that the GloVe word embeddings worked better for this task. Although, as the number of steps increases, the word2vec embeddings also seems to become better at the task, but never as good as Glove.

3. Model Per label performance:

Focusing on the fine-tuned models as they were more performant on validation error than any of the first models tested, the per label performance was computed for each model. These models were GloVe, word2vec and the combined embeddings and the standard parameters as described above.

a. Label 0

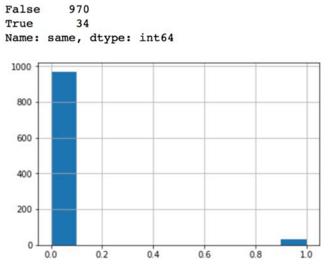
Label 0 was first analyzed. The first image shows the histogram of the true (1) and false(0) predictions of label 0 by the GloVe embeddings CNN. This shows that the classifier misclassified the label 0 as something else 83% of the time. In fact only 169 label 0s were correctly classified by the GloVe CNN classifier.



```
False    835
True     169
Name: same, dtype: int64
```

Figure 4 shows the histogram of good (1) and bad (0) predictions for the validation set for label 0 using the Glove embedding.

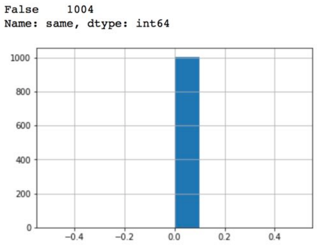
The word2vec embeddings did worse than the GloVe Embeddings as they only classified 34 examples of label 0s correctly.



```
False    970
True      34
Name: same, dtype: int64
```

Figure 5 shows the histogram of good (1) and bad (0) predictions for the validation set for label 0 using the word2vec embeddings.

Surprisingly, the combination of GloVe and word2vec misclassified label 0 in all cases.



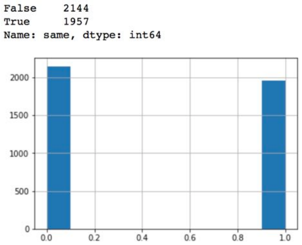
```
False    1004
Name: same, dtype: int64
```

Figure 6 shows the histogram of good (1) and bad (0) predictions for the validation set for label 0 using both embeddings

b. Label 1

Similarly, label 1 was analyzed.

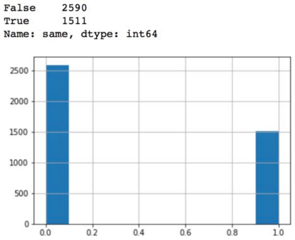
For the Glove Embeddings, the CNN missed about 52% of the classifications for label 1.



```
False    2144
True     1957
Name: same, dtype: int64
```

Figure 7 shows the histogram of good (1) and bad (0) predictions for the validation set for label 1 using the Glove embedding.

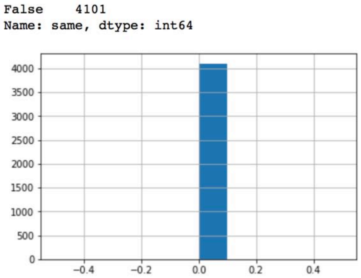
For the word2vec embeddings, the CNN missed 63% of the classifications for label 1.



```
False    2590
True     1511
Name: same, dtype: int64
```

Figure 8 shows the histogram of good (1) and bad (0) predictions for the validation set for label 1 using the word2vec embeddings.

For the combined embeddings, the CNN missed all of the classifications for label 1.

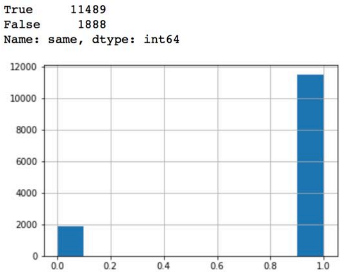


```
False    4101
Name: same, dtype: int64
```

Figure 9 shows the histogram of good (1) and bad (0) predictions for the validation set for label 1 using both embeddings.

c. Label 2

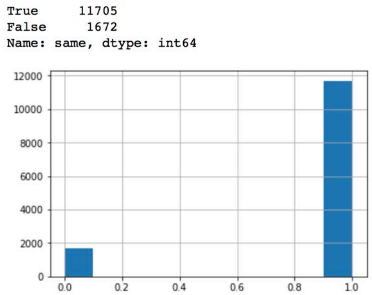
Glove classifications for label 2 are shown below. Label 2 has classified most of the dataset correctly.



```
True     11489
False     1888
Name: same, dtype: int64
```

Figure 10 shows the histogram of good (1) and bad (0) predictions for the validation set for label 2 using the Glove embedding.

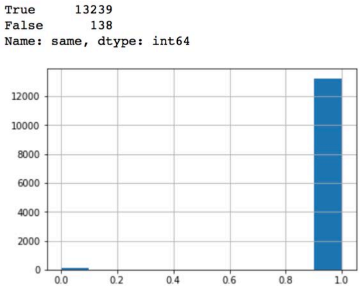
The word2vec also did better on label 2 than on either label 0 or 1, even doing better than glove on label 2. It classified 11705 examples correctly whereas the glove embeddings only was able to classify 11489 examples correctly.



```
True      11705
False      1672
Name: same, dtype: int64
```

Figure 11 shows the histogram of good (1) and bad (0) predictions for the validation set for label 2 using the word2vec embeddings.

The combination of both embeddings combined also seems to have done even better than either separate embeddings, only misclassifying 138 examples.

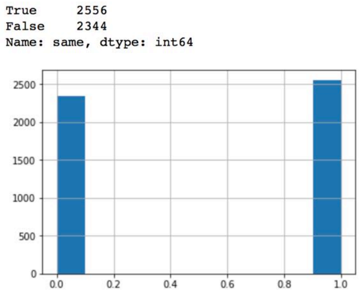


```
True      13239
False       138
Name: same, dtype: int64
```

Figure 12 shows the histogram of good (1) and bad (0) predictions for the validation set for label 1 using both embeddings.

d. Label 3

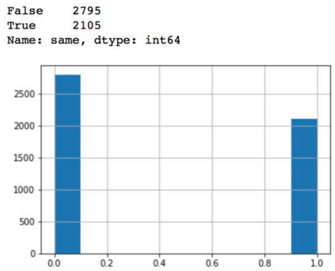
For label 3, the Glove embeddings used in the CNN were able to predict the correct label a little over 50% of the time.



```
True      2556
False     2344
Name: same, dtype: int64
```

Figure 13 shows the histogram of good (1) and bad (0) predictions for the validation set for label 3 using the Glove embedding.

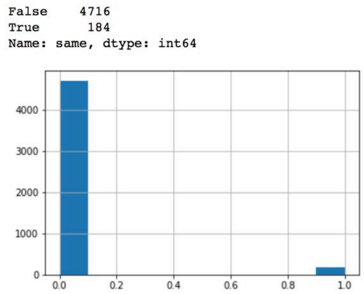
The CNN used with the word2vec embeddings were able to properly identify label 3 about 43% of the time.



```
False    2795
True     2105
Name: same, dtype: int64
```

Figure 14 shows the histogram of good (1) and bad (0) predictions for the validation set for label 3 using the word2vec embeddings.

The combination of the two embeddings yielded a model that only predicted label 3 correctly 3.8% of the time.

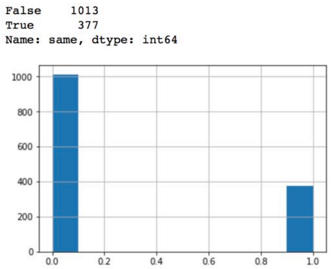


```
False    4716
True      184
Name: same, dtype: int64
```

Figure 15 shows the histogram of good (1) and bad (0) predictions for the validation set for label 3 using both embeddings.

e. Label 4

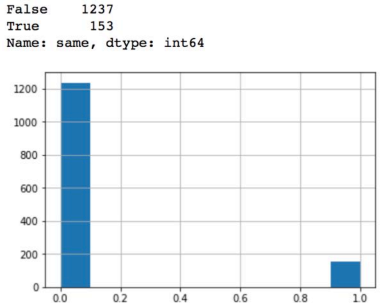
The Glove embeddings and CNN were able to predict label 4 about 27% of the time.



```
False    1013
True      377
Name: same, dtype: int64
```

Figure 16 shows the histogram of good (1) and bad (0) predictions for the validation set for label 4 using the Glove embedding

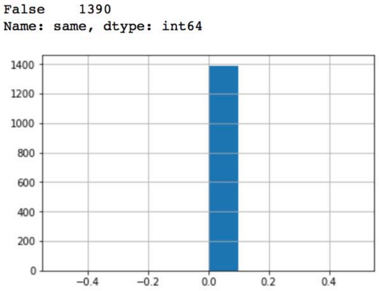
The word2vec embeddings and CNN were able to predict label 4 about 11 % of the time.



```
False    1237
True      153
Name: same, dtype: int64
```

Figure 17 shows the histogram of good (1) and bad (0) predictions for the validation set for label 4 using the word2vec embeddings.

The combined word embeddings were not able to predict any of label 4 correctly.



```
False    1390
Name: same, dtype: int64
```

Figure 18 shows the histogram of good (1) and bad (0) predictions for the validation set for label 4 using both embeddings.

f. Conclusions

In conclusion, while each of the embeddings works fairly well on label 2, the best (or least worst) embedding on all labels is the glove embedding as it is able to classify some examples from labels 0, 4 whereas the other are not. Overall the glove embeddings seems to do better on all labels.

The combination of the two embeddings yielded the best results on label 2 whereas it was fairly bad on the other labels. In order to understand this more profoundly, the training data was analyzed for distribution of labels.

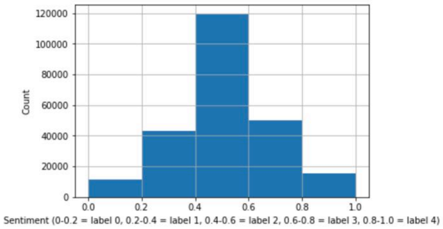


Figure 19 shows the distribution of labels over the training set.

This histogram shows that the training set is very unbalanced. This makes sense as most sentiment is neutral and very little is very extreme. This unbalanced data set however makes the classification tend to go for label 2 (as it is most common).

An interesting finding made by iterating over various number of steps and computing the per label classification accuracy of each model was found. As the number of steps increases, the CNNs seem to react to the unbalanced dataset and learn to predict fewer label 2s and more of the other classes. This accounts for the steady decrease in the models containing individual embeddings.

3. Final Model Built for Submission

When all the tests were finalized, the final model was built with the following parameters and submitted. These parameters were from the best models run with the fine-tuning enabled.

Parameters:

- Batch Size = 64

Adrien Cogny – Data Science in The Wild – Assignment 4 – Writeup

- Kernel Size = 3,4,5
- Log_every = 500
- Eval_every = 4000
- useEmbeddingNumber = 1
- maxNumSteps = 800 000
- fineTuned = True