

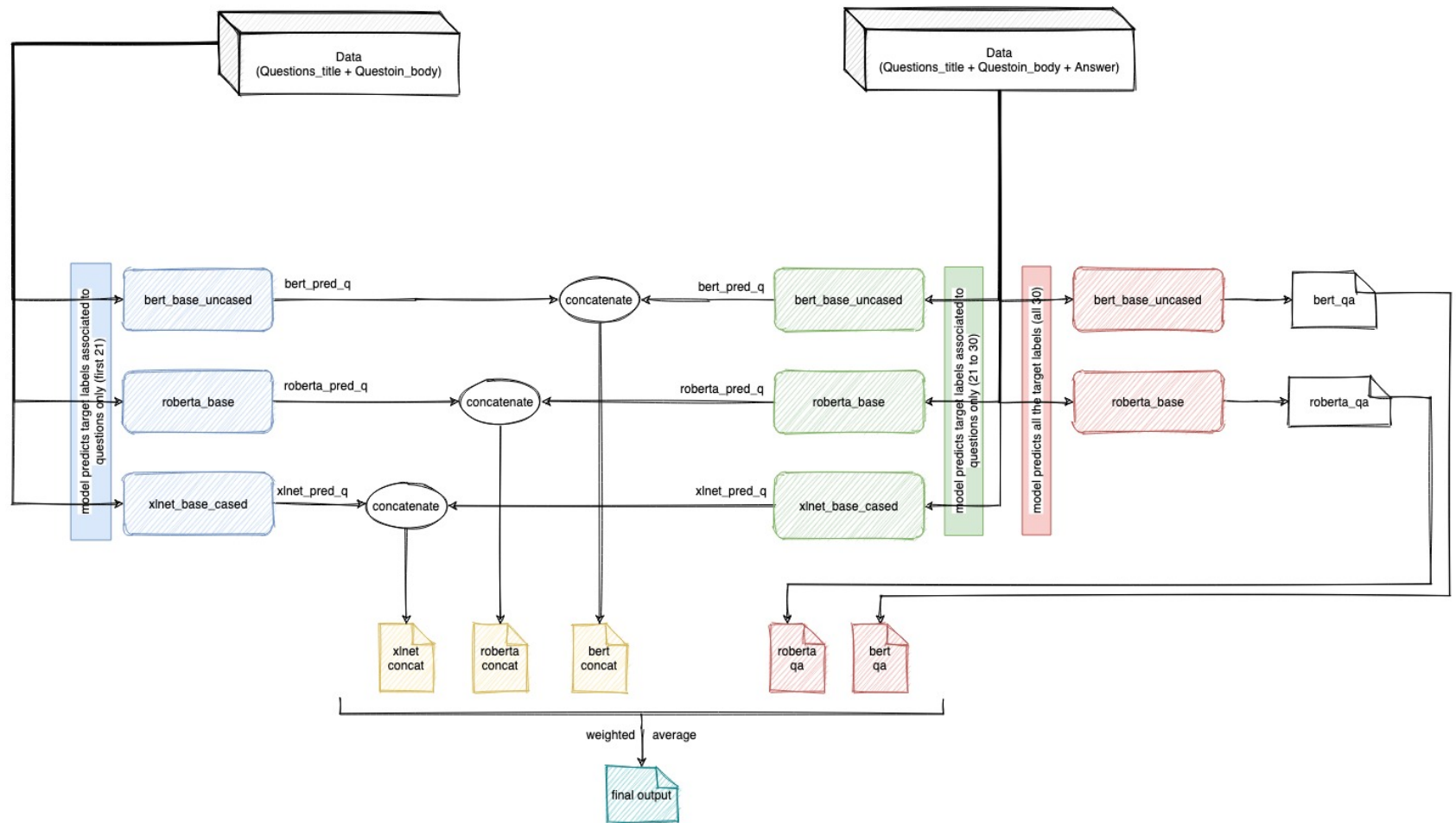
```
In [1]: # importing necessary libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from scipy.stats import spearmanr
import os

%matplotlib inline
from IPython.display import Image
```

In [2]: Image('diagram\_ensemble.jpg')

Out[2]:

### Ensemble of transformers



In [3]: *# performance comparasion of the transformers used in the final ensemble*

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["model", "dataset", "train loss", "cv loss", "train rhos", "cv rhos"]
x.add_row(['bert_base_uncased', 'questions', 0.3393, 0.3302, 0.5543, 0.6013])
x.add_row(['bert_base_uncased', 'answer', 0.3320, 0.3278, 0.4967, 0.5438])
x.add_row(['bert_base_uncased', 'question+answer', 0.3287, 0.3166, 0.5511, 0.6109])

x.add_row(['roberta_base', 'questions', 0.3542, 0.3400, 0.4953, 0.5674])
x.add_row(['roberta_base', 'answer', 0.3430, 0.3253, 0.3927, 0.4993])
x.add_row(['roberta_base', 'question+answer', 0.3546, 0.3397, 0.4305, 0.5082])

x.add_row(['xlnet_base_cased', 'questions', 0.3662, 0.3412, 0.4679, 0.5685])
x.add_row(['xlnet_base_cased', 'answer', 0.3611, 0.3401, 0.3531, 0.4702])
print(x)
```

model	dataset	train loss	cv loss	train rhos	cv rhos
bert_base_uncased	questions	0.3393	0.3302	0.5543	0.6013
bert_base_uncased	answer	0.332	0.3278	0.4967	0.5438
bert_base_uncased	question+answer	0.3287	0.3166	0.5511	0.6109
roberta_base	questions	0.3542	0.34	0.4953	0.5674
roberta_base	answer	0.343	0.3253	0.3927	0.4993
roberta_base	question+answer	0.3546	0.3397	0.4305	0.5082
xlnet_base_cased	questions	0.3662	0.3412	0.4679	0.5685
xlnet_base_cased	answer	0.3611	0.3401	0.3531	0.4702

```
In [4]: def get_data():
        if 'transformers_data' not in os.listdir():
            print('downloading data...')
            !wget 'https://github.com/SarthakV7/Kaggle_google_quest_challenge/blob/master/transformers_data.zip?raw=true'
            !mv 'transformers_data.zip?raw=true' 'transformers_data.zip'
            !unzip transformers_data

        # Importing the test data and sample submission data
        print('collecting kaggle data...')
        train = pd.read_csv('transformers_data/train.csv')
        test = pd.read_csv('transformers_data/test.csv')
        submission = pd.read_csv('transformers_data/sample_submission.csv')

        # Importing the predicted labels of bert
        print('collecting bert_base_uncased data...')
        bert_test_q = pd.read_csv('transformers_data/bert_pred_q_test.csv')
        bert_test_a = pd.read_csv('transformers_data/bert_pred_a_test.csv')
        bert_test_qa = pd.read_csv('transformers_data/bert_pred_qa_test.csv')
        bert_test_concat = pd.concat([bert_test_q, bert_test_a], axis=1)

        # Importing the predicted labels of roberta
        print('collecting roberta_base data...')
        roberta_test_q = pd.read_csv('transformers_data/roberta_pred_q_test.csv')
        roberta_test_a = pd.read_csv('transformers_data/roberta_pred_a_test.csv')
        roberta_test_qa = pd.read_csv('transformers_data/roberta_pred_qa_test.csv')
        roberta_test_concat = pd.concat([roberta_test_q, roberta_test_a], axis=1)

        # Importing the predicted labels of xlnet
        print('collecting xlnet_base_cased data...')
        xlnet_test_q = pd.read_csv('transformers_data/xlnet_pred_q_test.csv')
        xlnet_test_a = pd.read_csv('transformers_data/xlnet_pred_a_test.csv')
        xlnet_test_concat = pd.concat([xlnet_test_q, xlnet_test_a], axis=1)

        return bert_test_q, bert_test_a, bert_test_qa, bert_test_concat, roberta_test_q, roberta_test_a, roberta_test_qa, roberta_test_concat, xlnet_test_q, xlnet_test_a, xlnet_test_concat, train, test, submission
```

```
In [5]: # For binning, I've used the below code from:
# https://www.kaggle.com/markpeng/ensemble-5models-v4-v7-magic/notebook?select=submission.csv#Do-Inference
'''
Here the author has created 60 bins that correspond to 60 euqally spaced percentile values (between 1
-100)
of the 25 distinct target labels. The idea is to take the predicted values as an input and then prepr
ocess
them such that the final values are all from the 60 bins. This helps in making the predicted data muc
h more
structured/ordered.
'''
def optimize_ranks(preds, unique_labels):
    new_preds = np.zeros(preds.shape)
    for i in range(preds.shape[1]):
        interpolate_bins = np.digitize(preds[:, i],
                                       bins=unique_labels,
                                       right=False)

        if len(np.unique(interpolate_bins)) == 1:
            new_preds[:, i] = preds[:, i]
        else:
            new_preds[:, i] = unique_labels[interpolate_bins]

    return new_preds
```

```
In [6]: def get_exp_labels(train):
X = train.iloc[:, 11:]
unique_labels = np.unique(X.values)
denominator = 60
q = np.arange(0, 101, 100 / denominator)
exp_labels = np.percentile(unique_labels, q) # Generating the 60 bins.

return exp_labels
```

```
In [7]: def predict_labels():
    bert_test_q, bert_test_a, bert_test_qa, bert_test_concat, roberta_test_q, roberta_test_a, roberta_test_qa, roberta_test_concat, xlnet_test_q, xlnet_test_a, xlnet_test_concat, train, test, submission = get_data()
    bert_weight = 1
    roberta_weight = 1
    xlnet_weight = 1
    print('calculating weighted average...')
    predicted_labels = ((bert_test_qa + bert_test_concat) * bert_weight
                        + (roberta_test_qa + roberta_test_concat) * roberta_weight
                        + (xlnet_test_concat * xlnet_weight)) / (2*bert_weight + 2*roberta_weight + xlnet_weight)

    exp_labels = get_exp_labels(train)
    print('optimizing the predicted labels...')
    optimized_predicted_labels = optimize_ranks(predicted_labels.values, exp_labels)
    print('generating predicted labels dataframe...')
    df = pd.concat([test['qa_id'], pd.DataFrame(optimized_predicted_labels, columns=submission.columns[1:])], axis=1)
    print('done..!')

    return df
```

```
In [8]: output = predict_labels()
```

```
collecting kaggle data...
collecting bert_base_uncased data...
collecting roberta_base data...
collecting xlnet_base_cased data...
calculating weighted average...
optimizing the predicted labels...
generating predicted labels dataframe...
done..!
```

```
In [9]: output.head()
```

Out[9]:

	qa_id	question_asker_intent_understanding	question_body_critical	question_conversational	question_expect_short_answer	question_fa
0	39	0.973333	0.733333	0.16	0.551111	
1	46	0.866667	0.520000	0.08	0.768889	
2	70	0.913333	0.733333	0.08	0.840000	
3	132	0.913333	0.533333	0.08	0.720000	
4	200	0.946667	0.506667	0.08	0.782222	

```
In [10]: Image('score.png', width=1000, height=740)
```

```
Out[10]:
```

The screenshot shows the Kaggle website interface. At the top, there's a navigation bar with the Kaggle logo and a search bar. Below this, the main header for the 'Google QUEST Q&A Labeling' competition is visible, featuring a background image of two people and text indicating a \$25,000 prize money. The competition description mentions 'Improving automated understanding of complex question answer content' and notes it was created by Google with 1,571 teams participating 5 months ago. Below the header, there are tabs for 'Overview', 'Data', 'Notebooks', 'Discussion', 'Leaderboard', 'Rules', 'Team', 'My Submissions', and 'Late Submission'. The 'My Submissions' tab is active, showing a list of 47 submissions for user Sarthak Vajpayee. The submissions are sorted by 'Most recent'. The table lists three submissions, all of which succeeded with a private score of -0.00153 and public scores of 0.43658, 0.41606, and 0.43155 respectively. Each submission has a checkbox for 'Use for Final Score'.

47 submissions for [Sarthak Vajpayee](#) Sort by **Most recent**

**All** Successful Selected

Submission and Description	Status	Private Score	Public Score	Use for Final Score
<a href="#">kernel56f0a07e71</a> (version 38/38) 13 minutes ago by <a href="#">Sarthak Vajpayee</a> From Kernel [ <a href="#">kernel56f0a07e71</a> ]	Succeeded	-0.00153	0.43658	<input type="checkbox"/>
<a href="#">kernel56f0a07e71</a> (version 37/38) 20 minutes ago by <a href="#">Sarthak Vajpayee</a> From Kernel [ <a href="#">kernel56f0a07e71</a> ]	Succeeded	-0.00153	0.41606	<input type="checkbox"/>
<a href="#">kernel56f0a07e71</a> (version 36/38) an hour ago by <a href="#">Sarthak Vajpayee</a>	Succeeded	-0.00153	0.43155	<input type="checkbox"/>



**Using the above architecture, I was able to achieve a score of 0.43658 (top 4.4% in the kaggle leaderboard).**

In [10]: