Beat the Bookies With Machine Learning

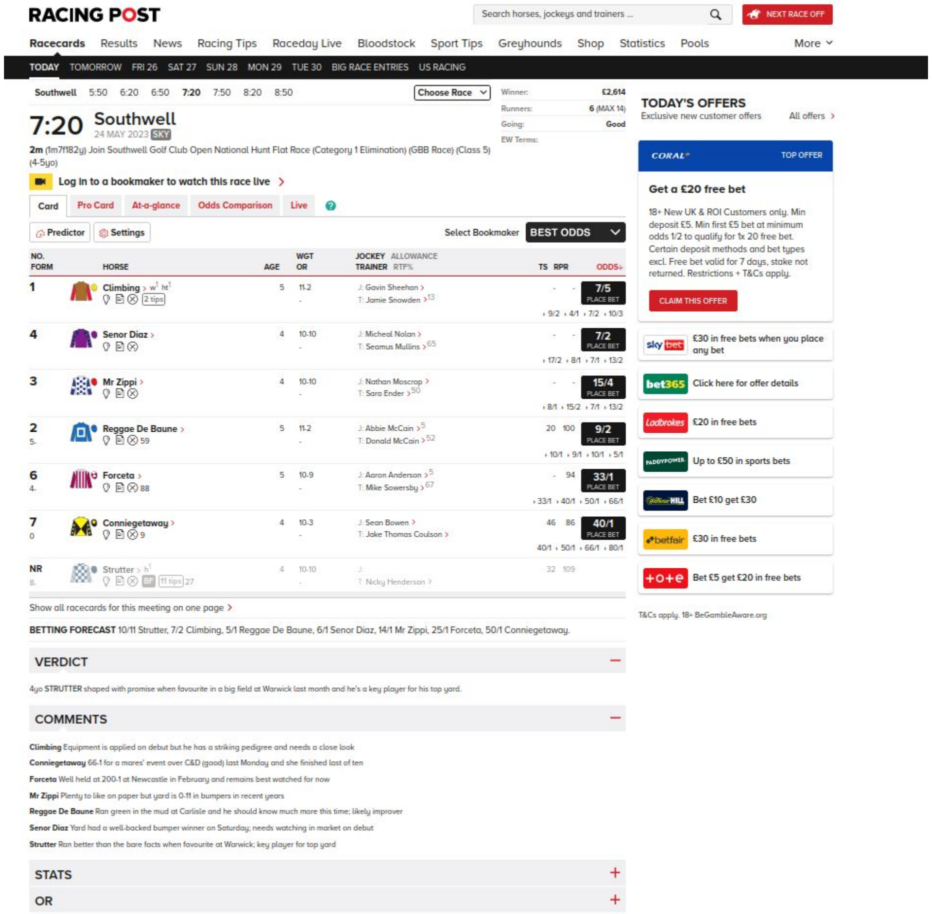
T. Bidewell, J. Brett, N. Evans, J. McInerney

Introduction

- Horse Racing
- Betting

Introduction

- Textual Data (Predictions)
- Numerical Data



Data

Clean text data | Fill missing data with mean of column | Normalize Columns

	Horse Age	Horse Top Speed	Horse RPRS Ranking	Horse ORS Ranking	Horse Weight	Prediction	Jockey Last 14	Trainer Last 14	horse_race_id	Position	Race_Id
0	-0.997221	-2.905166	-0.004418	0.215331	-0.189955	22 over 7f both wins on aw remains open to fur...	0.873726	0.699002	18-04-2023_1:50_Newmarket_Think Climate	9.0	18-04-2023_1:50_Newmarket
1	-0.997221	0.079789	-0.297183	0.170249	-0.389579	ran well in france on final 2yo start more is ...	-1.149677	-0.956598	18-04-2023_1:50_Newmarket_Awtaad Prince	7.0	18-04-2023_1:50_Newmarket
2	-0.997221	0.463569	0.037405	0.000000	-0.389579	one of two runners for charlie appleby made al...	-0.340316	2.189043	18-04-2023_1:50_Newmarket_City Of Kings	6.0	18-04-2023_1:50_Newmarket
3	-0.997221	0.207716	-0.088065	0.000000	-0.389579	solid third in the convivial maiden then won w...	0.974896	-0.128798	18-04-2023_1:50_Newmarket_Hi Royal	4.0	18-04-2023_1:50_Newmarket
4	-0.997221	0.548854	0.204699	0.440739	-0.389579	major player on rprs and looks likely to give ...	-0.542656	0.616222	18-04-2023_1:50_Newmarket_Holguin	2.0	18-04-2023_1:50_Newmarket

Web Scraping

- Static Website - Results Page:
  - Beautiful Soup

- Dynamic Website - Predictions: (Website's content changes for each user - the odds of from the betting site)
  - Selenium

Embeddings

- TF-IDF

- BERT
  - Fine-Tune
  - Frozen

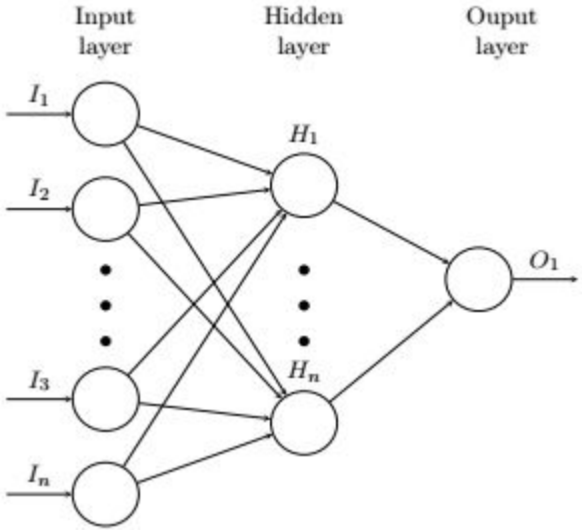
TF-IDF: Models

- Pointwise

- Pairwise

Pointwise (Regression)

- Predict Position of horse
- MLP (Multi-Layered Perceptron)
- 1 Hidden Layer
- Activation Function: ReLU
- Loss function MSE
- Sort based off predicted score



Loss Function - Mean Squared Error (MSE) - Regression

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\underbrace{Y_i}_{\text{True Value}} - \underbrace{\hat{Y}_i}_{\text{Predicted Value}})^2$$

Pairwise (Classification)

- For pairs of horses, predict the winner
- MLP
- 1 Hidden Layer with ReLU activation
- Softmax
- 2 outputs
- Sum the log probabilities to get final score and sort horses based off these
- (ELO)

Loss Function - Negative Log Likelihood (NLL) - Classification

$$
\begin{aligned}
\theta^* &= \underset{\theta}{\operatorname{argmax}}\ P(X|\theta) \\
&= \underset{\theta}{\operatorname{argmax}}\ \textstyle\prod_{t=1}^{T} P(obj = x^{(t)}, class = y^{(t)}|\theta) && \text{\#supposing the examples are all independent} \\
&= \underset{\theta}{\operatorname{argmax}}\ \textstyle\prod_{t=1}^{T} P(obj = x^{(t)})P(class = y^{(t)}|object = x^{(t)}, \theta) && \text{\#by applying the chain rule} \\
&= \underset{\theta}{\operatorname{argmax}}\ \textstyle\prod_{t=1}^{T} P(class = y^{(t)}|obj = x^{(t)}, \theta) && \text{\#because the } P(obj = x^{(t)}) \text{ are constant} \\
&= \underset{\theta}{\operatorname{argmax}}\ \textstyle\sum_{t=1}^{T} log(P(class = y^{(t)}|obj = x^{(t)}, \theta)) && \text{\#because log is an increasing function} \\
&= \underset{\theta}{\operatorname{argmin}}\ \textstyle\sum_{t=1}^{T} \text{-}log(P(class = y^{(t)}|obj = x^{(t)}, \theta)) && \text{\#minus to get a loss}
\end{aligned}
$$

BERT

- Fine Tune

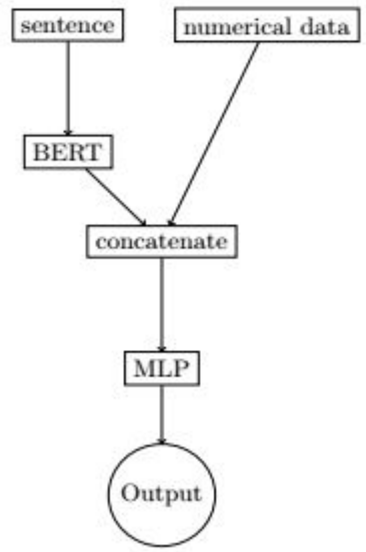
- Frozen

Process

- Tokenize sentence using AutoTokenizer from transformers - add [CLS] token at start of sentence
- Extract input_ids and attention_masks
- Feed into Pre-Trained: `bert-base-uncased`

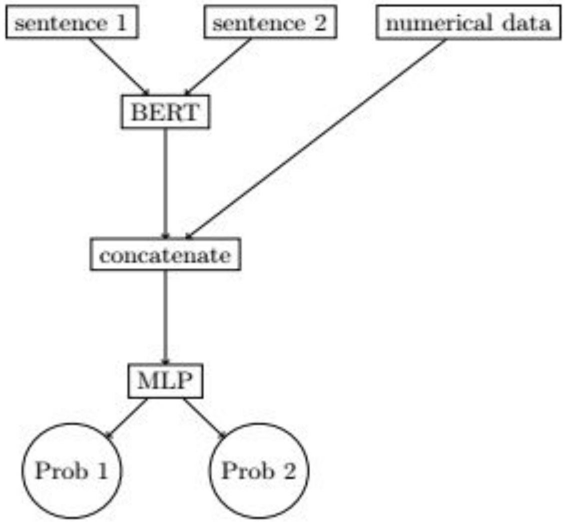
Pointwise

- Regression to predict position of horse
- Concatenate numerical data with CLS token in last hidden state.
- Fully connected layers with one output neuron
- Mean Squared Error



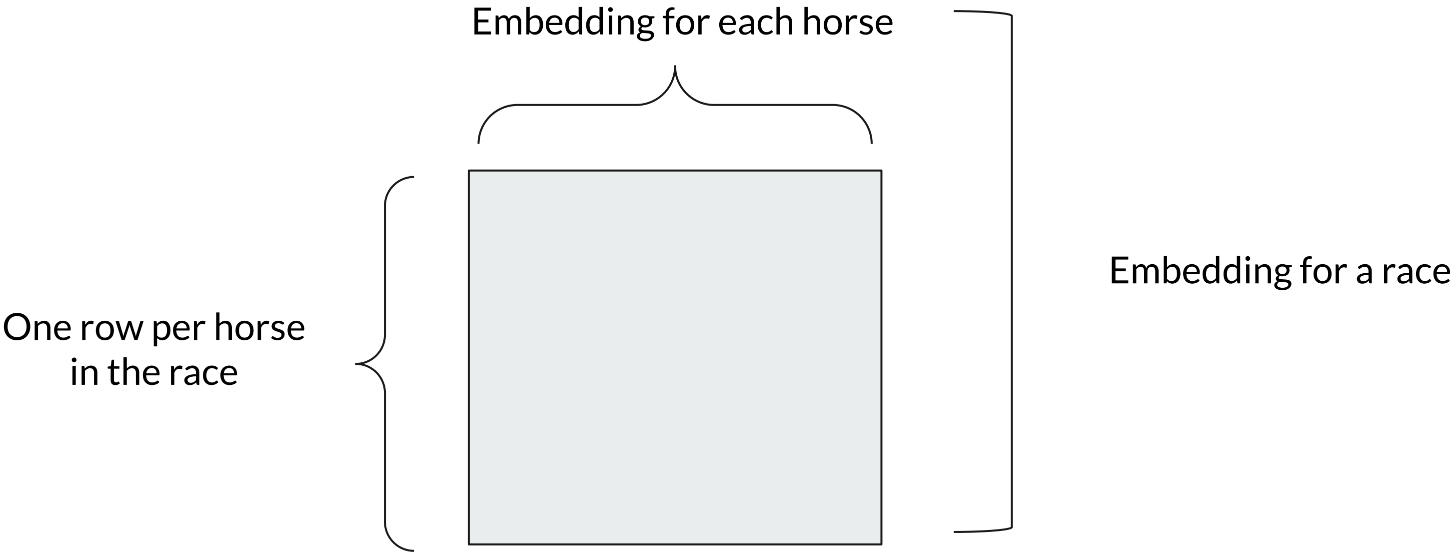
Pairwise

- For pairs of horses in same race, predict winner
- Concatenate numerical data with CLS token in last hidden state
- Fully connected layers with ReLU activation and Softmax
- 2 output neurons
- NLLLoss
- Sum log probabilities to get score for each horse
- Sort on these

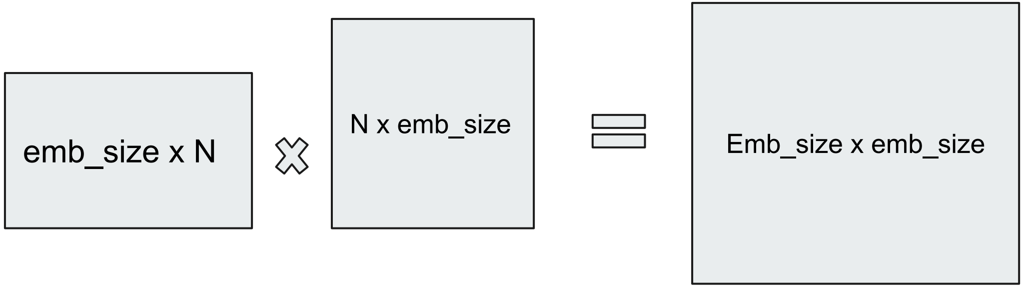


Other Methods: embed race with 2d tensor (custom collate function)

- Using 2d embeddings for a race allowed for further deep learning models.
-



CNN Regression



- CNN Regression:
    - Predict index of winning horse
    - transpose(M) * M has shape (emb_size x emb_size)
    - Initial no. channels = 1, 2 convolutional layers with max pooling before flattening and fully connected layers with one output neuron
    - MSE Loss

Listwise

- N x emb_

```python
class NDCGLoss(nn.Module):
    def __init__(self, k):
        super(NDCGLoss, self).__init__()
        self.k = k

    def forward(self, y_true, y_pred):
        batch_size = y_true.size(0)  # Get the batch size

        all_ndcgs = []

        for i in range(batch_size):

            true_sample = y_true[i].squeeze().float()  # Remove the last dimension (size 1) from true tensor
            pred_sample = y_pred[i].squeeze().float()  # Remove the last dimension (size 1) from pred tensor

            mask_true = true_sample != -1
            true_sample = true_sample[mask_true]
            pred_sample = pred_sample[mask_true]

            true_sample = 1/true_sample

            # Convert continuous values to discrete relevance labels
            labels_pred = pred_sample.argsort(descending=True) + 1
            labels_true = true_sample.argsort(descending=True) + 1

            ndcg_individual = ndcg_score([labels_true.detach().cpu().numpy()], [labels_pred.detach().cpu().numpy()], k=self.k)
            all_ndcgs.append(ndcg_individual.item())

        # Compute the average NDCG for the batch
        average_ndcg = torch.tensor(all_ndcgs, requires_grad=True).mean()

        # Return the negative NDCG as the loss
        loss = 1 - average_ndcg
        return loss
```

nDCG

- Normalized Discounted Cumulative Gain
- Evaluation metric commonly used to compare the effectiveness of ranking algorithms.
- Compares the similarity of the predicted list and the true list.
- Basic principle is that you want the more relevant documents to have the lower ranks.

Award a result by its relevance

For all the results

$$\sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Add the scores

Punish a result by its rank

Results

We used these NDCG score to evaluate 3 different types of language embeddings:

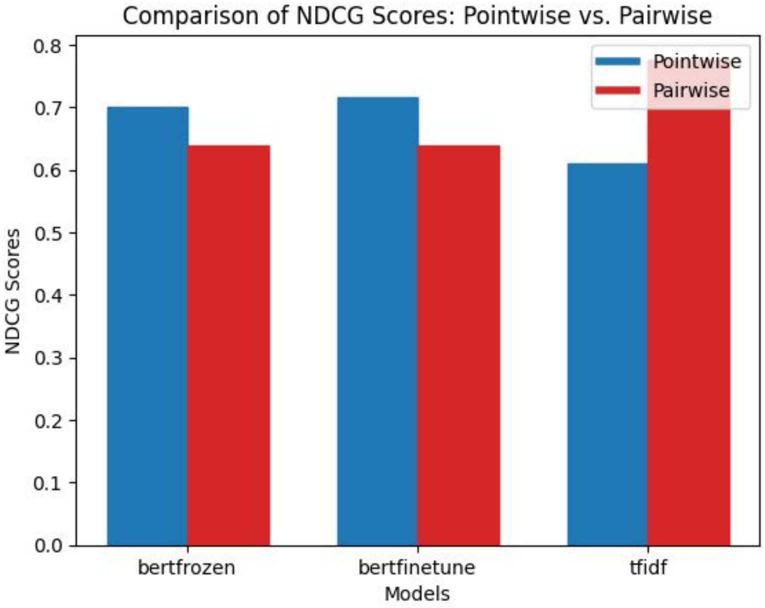
- TF-IDF
- BERT fine-tuned
- BERT frozen

In combination with two different types of ranking models:

- PairWise
- PointWise

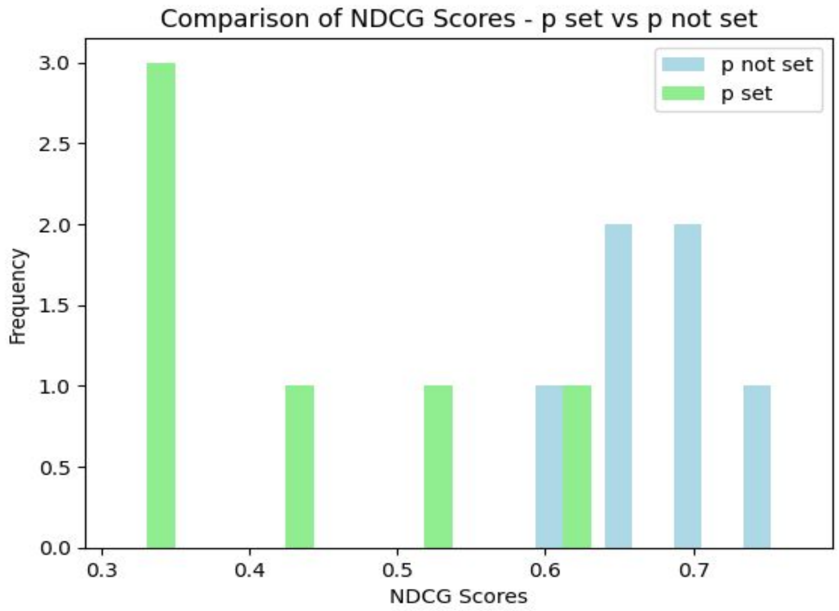
Pointwise vs Pairwise

- We can see the difference in the performance of the pairwise and pointwise ranking models.
- We can also see the difference in the performance of the models in combination with the embedding technique used.



nDCCG Using a Cut Off Point p

- This displays the difference in nDCG scores depending on the amount of horses we take into consideration per race.
- ‘P not set’ : all horses taken into consideration (default)
- ‘P set’: p horse taken into consideration. (p = 3 in the graph displayed)



Conclusions

- Success
  - Proof of Concept
  - Compare our approaches
- Potential Improvements
  - More data
  - Listwise approach