Question Answering on the SQuAD Dataset

Laëtitia Shao
lshao1@stanford.edu

Benoît Zhou
benzhou@stanford.edu

Abstract

In this paper, we implemented a model that answers questions on a given sample text, working on the SQuAD dataset. More precisely, we re-implemented the BiDAF model, and obtained on the test set a F1 score of 75.407 and EM score of 65.572 by ensembling over several models we worked on.

1 Introduction

Machine text comprehension is an interesting and challenging problem in natural language processing, as it requires to understand complex interactions between words. It is also a good mark to evaluate the progress made in NLP and measure how well models can understand text. Moreover, text comprehension has commercial applications such as text categorization, news-gathering, voice assistants, summarization or scalable text analysis.

Machine text comprehension is very vast, varying from grasping the gist of a moderately difficult text to gaining a deep and thorough understanding of difficult texts such as poetry or law bills. In our project, we will focus on reading comprehension and question answering. Burges defined machine comprehension tested by question answering by “A machine comprehends a passage of text if, for any question regarding that text that can be answered correctly by a majority of native speakers, that machine can provide a string which those speakers would agree both answers that question, and does not contain information irrelevant to that question.” [1]. Working on the Stanford Question Answering Dataset (SQuAD), which is a reading comprehension dataset, we have been working on a model that, given a paragraph and a question about it as input, will answer the question correctly. An important characteristic of this 100k+ question-answer taken from Wikipedia dataset is that the answer to every question is a segment of text, or span, from the context paragraph. Hence, the model does not have to generate the answer text, but only needs to select the span of text that answers the question.

It is noteworthy that it is hard to evaluate such a model as there are many possible answers that are all correct given a particular question. We will measure the performance of our model using two metrics: the F1 and Exact Match (EM) score. The former is given by the harmonic mean of precision and recall and the latter is a binary variable that measures the exact string match between the predicted answer and the gold label.

2 Related Work

The public leaderboard on the SQuAD website displays many deep learning models built for this dataset. Since the initial paper by Rajpurkar et al. [6], many researchers have come up with different architectures, from Weston et al. with memory networks to Seo et al. with bidirectional attention flow [7], passing by Wang et al. with match-lstm and answer pointer [8] amongst others. The current state of the art model, Hybrid AoA Reader (ensemble) by the Joint Laboratory of HIT and iFLYTEK Research [9], yields a F1 score of 89.281 and an EM score of 82.482, which is very impressive since it is actually better than human performance.

3 Our approach - Model Description

Our models take as input a tokenized context and question and outputs two distributions, one for the starting position of the answer in the context and one for the end position of the answer. In this section we will describe the different architectures we experimented with, starting from the baseline model that was provided as a starter point and from which we progressively built our improvements.

3.1 Baseline model

The baseline model that was provided as starting code is structured as follows. First, the tokenized context and question are embedded in dimension d using pre-trained GloVe vectors ([5]) to get a vector representation $C \in \mathbb{R}^{N \times d}$ and $Q \in \mathbb{R}^{M \times d}$ where N is the maximum context length, M is the maximum question length. These vector representations are then successively fed into the following layers

- Bi-directional encoder layer. The encoder layer is a bi-directional Gated Recurrent Unit network ([2]) with hidden size h that allows the model to learn temporal relation within the context and the question. Both C and Q are fed to this encoder. We concatenate the two outputs of this layer and get a context encoding $H_C \in \mathbb{R}^{N \times 2h}$ and a question encoding $H_Q \in \mathbb{R}^{M \times 2h}$.
- Context to Question attention. The attention layer computes an attention distribution of the context attending to the question. Context to question attention $A \in \mathbb{R}^{N \times 2h}$ is computed using the following equations:

$$\alpha = \text{softmax}(H_C H_Q^T) \in \mathbb{R}^{N \times M} \tag{1}$$

$$\forall i \leq N, A_i = \sum_{j=1}^{M} \alpha_i^{(j)} (H_Q)_j \tag{2}$$

  The attention matrix is then concatenated with the original context embedding and the attention output is $H = ([H_C, A]) \in \mathbb{R}^{N \times 4h}$.
- Modeling output: the attention output H is fed to a fully connected layer which outputs a downsized representation L of size $\mathbb{R}^{N \times h}$.
- Output layers: The modeling output L is used as input to the output layers which will compute the start and end distributions. Each output layer independently computes the probability distribution as follows:

$$p_{\text{start|end}} = \text{softmax}(L w_{\text{start}|end}) \tag{3}$$

3.2 Bidirectional Attention

The bidirectional attention model replaces the simple context to question attention layer from the baseline model by a bidirectional attention layer that computes a context to question attention output and a question to context attention output ([7]). In practice the context to question attention is computed as follows from the context and question embeddings H_C and H_Q. First, we compute a similarity matrix $S \in \mathbb{R}^{N \times M}$

$$S_{i,j} = [(H_C)_i; (H_Q)_j, (H_C)_i \circ (H_Q)_j]W \tag{4}$$

where $\circ$ is the element-wise multiplication of two vectors and $W \in \mathbb{R}^{6h}$ is a learnable weight matrix.

The logits for the context to question output are computed from the similarity matrix by taking the row-wise softmax. Similarly to the simple context to question attention of the baseline, the logits are used to compute a weighted sum of the question vectors for each token in the context.

The question to context attention is one single weighted sum of the context vectors corresponding to the whole question attending to the context. The logits are computed by first taking the row-wise maximum of the similarity matrix and taking the softmax on the result. The attention outputs are then combined with the context embedded representation and concatenated to get the final output:

$$\forall i \leq N, H_i = [H_C^{(i)}, U^{(i)}, U^{(i)} \circ H_C^{(i)}, V \circ H_C^{(i)}] \in \mathbb{R}^{8h} \tag{5}$$

where $U \in \mathbb{R}^{N \times 2h}$ is the Context to Question attention output and $V \in \mathbb{R}^{2h}$ is the Question to Context attention output.

3.3 Self-Attention

The self-attention layer [4] directly matches the question-aware passage l-dimensional representations v_i against themselves to obtain attention-pooling vectors a_i of the whole passage v:

$$e_j^i = w_0^T \tanh(W_1 v_j + W_2 v_i)$$

$$\alpha^i = \text{softmax}(e^i)$$

$$a_i = \sum_{j=1}^{N} \alpha_j^i v_j$$

where w_0, W_1, W_2 and trainable weight vector and matrices. The attention-pooling vectors a_i are then concatenated with the representations v_i to be fed to a bidirectional GRU modeling layer which outputs the hidden states.

3.4 Stacking

It is possible stack the bidirectional attention and the self-attention layers:

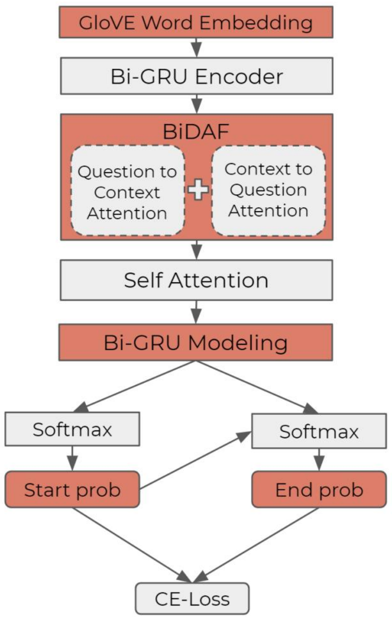

Figure 1: Architecture of our model

To partially alleviate the situation where the span start prediction was located after the span end prediction, we have concatenated the output modeling layer with the softmax start probability to compute the end probability, which reduced the number of occurrences of this particular problem.

4 Experiments

4.1 Training Dataset Exploration

The Stanford Question Answering Dataset([6]) contains 86326 entries of questions, answers, contexts and spans. In order to better know the dataset we were working with, we plotted the distributions of the lengths (in tokens) of the questions, answers, and contexts. (Figure 2)

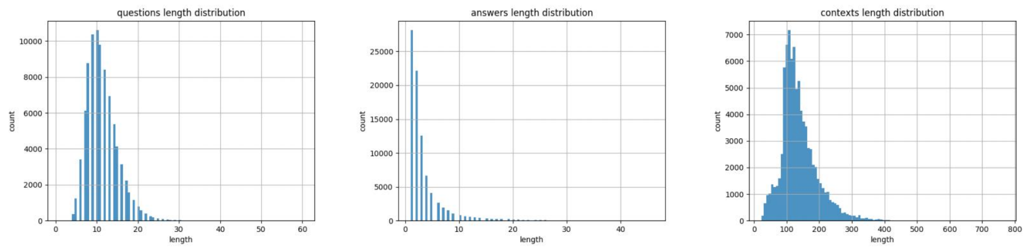

Figure 2: Distribution of lengths (in tokens) of questions (left), answers (center) and contexts (right) in the training data set.

There are only 872 questions of length greater than 22, over a total of 86326 questions (i.e. 1%), hence we set the hyperparameter `question_len` to 22. Similarly, as there are only 842 contexts of length greater than 300, over a total of 86326 contexts (i.e. 1%), we set `context_len` to 325.

Moreover, we also took a look at the occurrences of the answers in the context, and found that answers appear rather at the beginning of the context. (Figure 3)

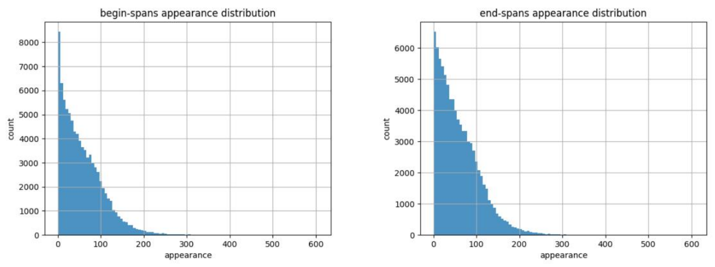

Figure 3: Distribution of positions of the begin (left) and end (right) cursors in the training data set.

Finally, we looked at the distribution of the main different question types and discovered that almost half of the questions were *what* type questions. (Figure 4)

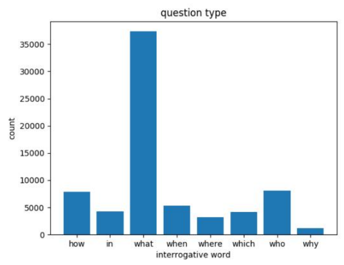

Figure 4: Distribution of positions of question types.

4.2 Training and evaluation

To train our model, we minimize the sum of the average cross entropy loss for the start and end distribution.

$$\text{Loss} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{N} y^{(i)}_{start,j}\log(p^{(i)}_{start,j}) + y^{(i)}_{end,j}\log(p^{(i)}_{end,j})) \tag{6}$$

Where $y^{(j)}_{start}$ and $y^{(j)}_{end}$ are the one-hot vectors of the ground-truth start and end spans and B is the batch size. We trained with AdamOptimizer ([3]) with a learning rate of **0.001**, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for **13000** iterations. We used dropout of **0.15** and trained with batch size **64**, context length **325** and question length **22**.

We evaluate our model on two metrics. The EM score, which indicates whether there's an exact match between the prediction and the ground truth, and the F1 score, a more lenient metric, which rewards partial matches between the prediction and the ground truth.

We choose the model parameters that maximized the F1 score on the development set. The predicted start and end span were selected independently from each other.

$$\hat{y}_{start|end} = \arg\max p_{start|end} \tag{7}$$

Ensembling was done at evaluation time by averaging the output distributions of all the models.

4.3 Results

The result table (1) show the EM and F1 scores obtained from different models we tried. The baseline was trained with hidden size 200. The Bidaf model was trained with hidden size 200 for the encoders. The Self Attention model was trained with hidden size 64 for both the encoders and the self attention layer. The first stacked model was trained with hidden size 50.

Our two best performing models were obtained by increasing the hidden size from 50 to 150 and self attention hidden size to 100. The `Stack-100-pointer` model introduces a direct dependency of the end output layer on the begin output layer and the `Stack-100-modeling layer` introduces a modeling layer before the output layer similar to what was implemented in the original BIDAF model ([7]). The ensemble model is the result of the ensembling of the same models, using both GRUs and LSTMs and different hyperparameters.

Table 1: EM and F1 score on development set

Model	EM Score	F1 Score
Baseline (hidden size 200)	34.38	43.622
BiDAF (hidden size 200)	40.25	50.04
Self Attention (hidden size 64)	51.81	65.12
BiDAF + SelfAttention (hidden size 50)	58.54	69.40
Stack-100	**60.96**	**71.63**
Stack-100 + modeling layer	60.7	71.60
Ensemble	**65.025**	**75.249**

5 Result Analysis

We analyze the results of our best performing model `Stack-100-pointer` model.

5.1 Breakdown by question type

First, we break down the model's performance based on the type of questions contained in the dataset and computing the F1 and EM scores within the category (2 using results of the model on the **preprocessed** dev set which contains 10391 questions.[1]

We see from table 2 that questions containing the keyword *why* have the lowest EM and F1 scores, while questions containing the keyword *who* and *when* have the highest scores. One explanation for this is that questions of the type *why* generally require a longer answer than questions of the type *when* or *who* can be easily answered by finding the right date or person in the context. This is further confirmed in figure 5 where we see that the answer length for questions of type *why* is generally larger than for other question types, while there does not seem to be a distinction between question types for context and question lengths.

Table 2: EM and F1 score on development set by question category

Question type	EM Score	F1 Score
All	51.20	66.12
Why	17.83	48.30
What	48.58	64.10
When	**65.51**	**76.61**
How	48.79	65.55
Who	61.83	70.00
Where	44.33	61.54

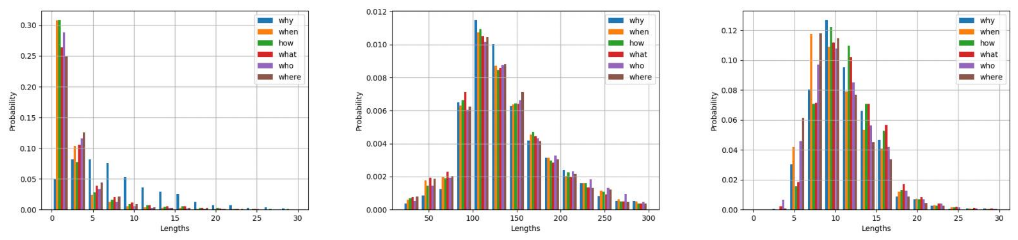

Figure 5: Distribution of answers (left), contexts (center) and questions (right) lengths in number of tokens in the dev set by question type

5.2 Attention visualization

For further analysis, we visualize the output of the Bidirectional Attention layer which gives us two attention outputs: a context to question attention distribution (Figure 6), a question to context distribution (Figure 7) as well as the resulting span distributions (Figure 8) for a failure and success case. Supplementary materials also offers a visualization of the self attention weights. (Figure 9)

- For the failure case the question and context were:

    Who became the king of the Canary Island?

    bethencourt took the title of king of the canary islands, as vassal to henry iii of castile in 1418. jean's nephew maciot de bethencourt sold the rights to the islands to enrique perez de guzman, 2nd count de niebla.

  Our model predicted the answer to be *henry iii of castile* when the correct answer was *bethencourt*.

[1]Note that since we consider the score on the preprocessed development set, this score is different than the one reported by the `official_eval` script.

The context to question attention distribution did correctly map *king of the canary islands* with a lot of certainty but the distribution of answer spans favored *henry iii of castile* over *bethencourt*. It is however noteworthy that the model suggested two answer spans, one of them being the correct one. It therefore was not very far from predicting the correct answer.

- We chose the following positive example:

  who was the nfl commissioner in early 2012?

  in early 2012, nfl commissioner roger goodell stated that the league planned to make the 50th super bowl "spectacular" and that it would be "an important game for us as a league".

  The context to question attention distribution did correctly map *nfl commissioner* and *in early 2012* with a lot of certainty and the distribution of answer spans only returned the correct span with probability one. We can also note that the model also maps context words like "league" or "super bowl" to "NFL", which suggests that it is able to infer a close relation between words that designate the same idea.

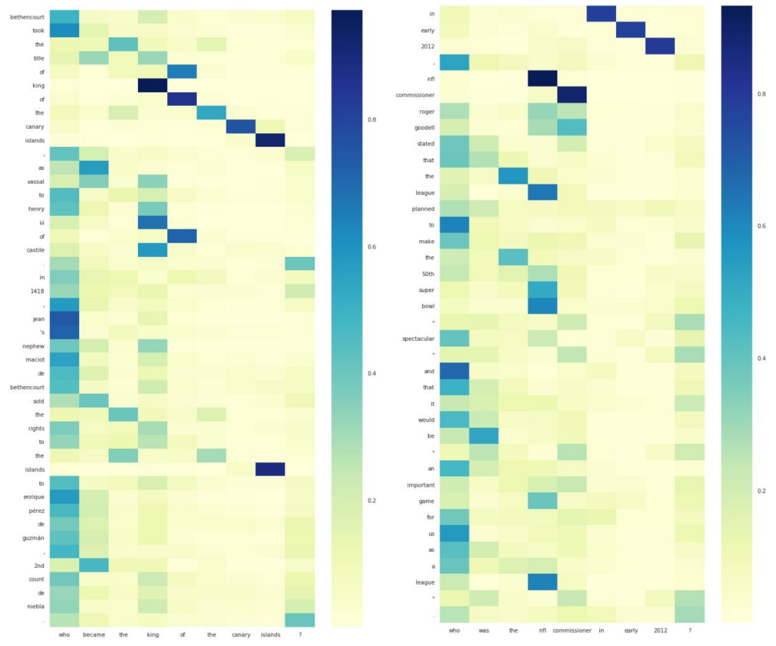

Figure 6: Context to Question Attention distribution from the Bidirectional Attention layer. Left: failure case, Right: success case

6 Conclusion

For this project, we have reimplemented the BiDAF layer that we have stacked with a self-attention layer. Besides, we have showed how important attention was, as the mere self-attention layer increased the baseline scores by 50%! We have also tried running more complex models, such as replacing the final fully connected layer by a RNN for instance, but have faced memory issues.

Overall, we are rather satisfied by our model, as it has achieved good results. The visualizations show that even when our model was wrong, the correct answer was never completely neglected, which is promising for future work. There are indeed still many improvements to be made, and we have only tested a couple of hyperparameters during the time we had to work on our project.

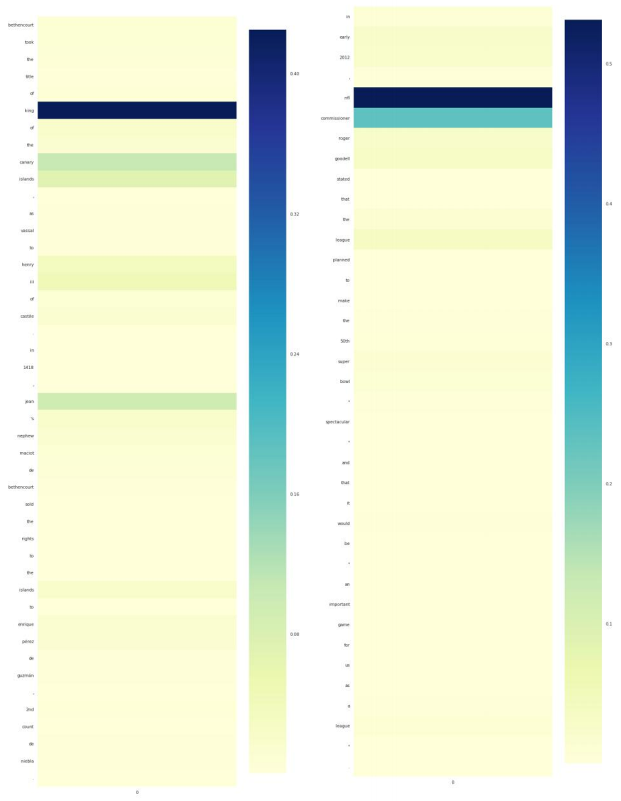

Figure 7: Question to Context Attention distribution from the Bidirectional Attention layer Left: failure case, Right: success case

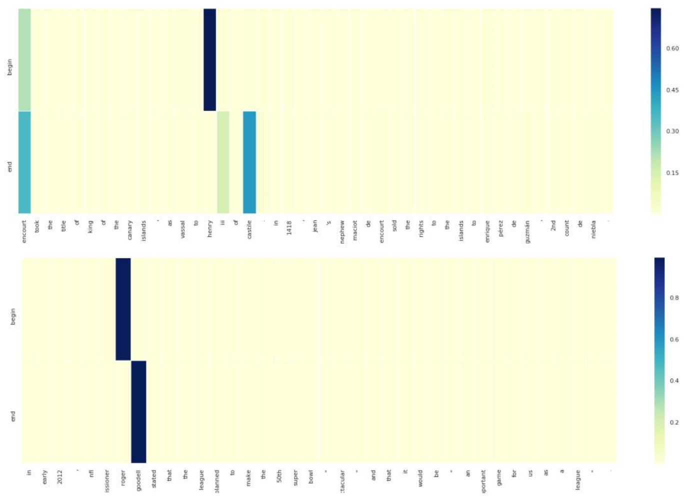

Figure 8: Distribution of answer spans, Left: failure case, Right: success case

Acknowledgments

We would like to thank Richard Socher and the teaching staff for the great course CS224n. It has been a very enjoyable experience, and we really liked working on this project.

References

[1] Christopher J.C. Burges. Towards the machine comprehension of text: An essay. *TechReport:MSR-TR-2013-125*, 2013.

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] Microsoft Research Asia Natural Language Computing Group. R-net: machine reading comprehension with self-matching networks. 2017.

[5] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[7] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[8] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

[9] Si Wei Shijin Wang Ting Liu Guoping Hu Yiming Cui, Zhipeng Chen. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

Supplementary materials

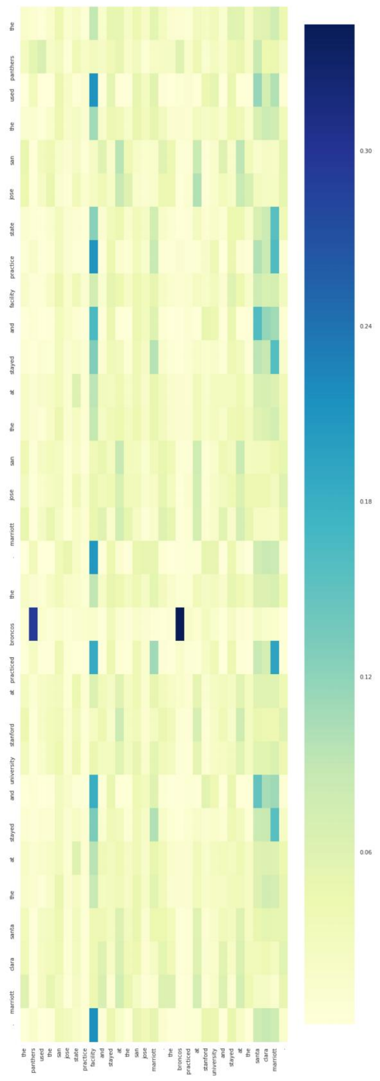

Figure 9: Visualization of the output of the self attention layer for the context ”The panthers used the san Jose state practice facility and stayer at the san jose marriott. The broncos practiced at Stanford University and stayed at the Santa Clara Marriot”. One can see that for the word ”Broncos”, the attention is focused primarily on the words ”Panthers” and ”Broncos” which are the names of two teams.