SAR Sentence Classification

Gabriel Okazaki - 101512611

Problem Statement

As a newcomer to Canada and wanting to experience the job market, studying the interviewing process for jobs here was one the first things I did.

A common framework used in interviews here is the **SAR** (Situation, Action, Result) or STAR (Situation, Task, Action Result).

These frameworks help candidates convey information in a organized and logical manner.

Could we create a model that helps us evaluate if an interview answer is compliant with this structure?

SAR Framework

The SAR method is an interview technique that gives you a straightforward format you can use to tell a story for answering common interview questions such as “Tell me about a time when…”

Although STAR is more commonly known than SAR, the scope of this project was limited to SAR in order to simplify the complex semantics needed to solve this problem.

SAR Components

Situation

Set the scene and give the necessary details of your example.

"Recently, when I was serving a customer behind the checkout, I had a customer shout at me for not being eligible for a refund."

Action

Explain exactly what steps you took to address it.

"I asked my colleague to take over for me so that I could talk to the customer one-on-one and better explain the situation."

Result

Share what outcomes your actions achieved.

"In the end, he began to calm down and accepted my explanation."

Dataset

Even with extensive research, I was not able to find a dataset that attended to my need (sentences labeled as Situation, Action or Result).

A dataset was generated by the following methods:

- 90% sentences generated and labeled by ChatGPT 4o
- 10% of sentences scrapped from websites and labelled by hand

Dataset head

	Sentence	Category
0	The project deadline was approaching rapidly.	Situation
1	I conducted a thorough market analysis to iden...	Action
2	We successfully exceeded our sales target by 25%.	Result
3	The weather was unpredictable.	Situation
4	I decided to implement a new training program.	Action

Number of rows

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1187 entries, 0 to 1186
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Sentence  1187 non-null   object
 1   Category  1187 non-null   object
dtypes: object(2)
memory usage: 18.7+ KB
```

Category distribution

```
Category
Situation    412
Result       397
Action       378
Name: count, dtype: int64
```

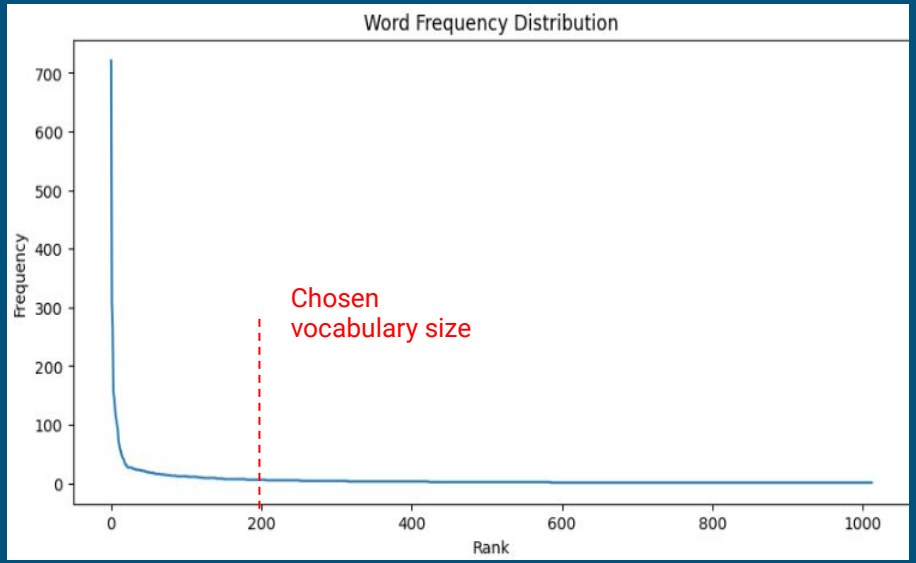
Pre-processing

- Labels were mapped to integers: {'Situation':0, 'Action':1, 'Result': 2}
- Dataset was split into Train and Test, with test_size of 20% and stratification by category.

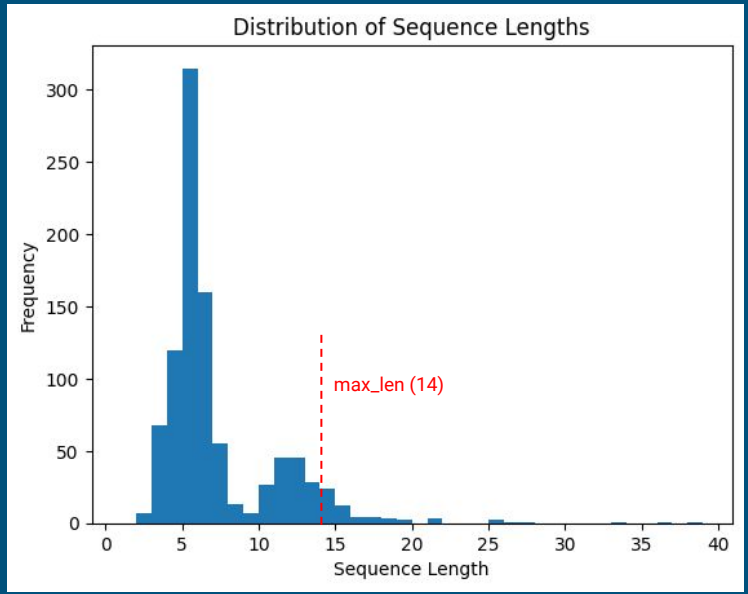
```
Train labels distribution: label
0    329
2    318
1    302
Name: count, dtype: int64
Test labels distribution: label
0    83
2    79
1    76
Name: count, dtype: int64
```

Tokenization

Word frequency distribution was analyzed in order to determine vocabulary size



Distribution of sequence lengths was used to determine max length of sequences



Model

The model was trained from scratch.

The main layer chosen was a Bi-directional LSTM for its flexibility and good capacity for sequence labeling.

Dropout and batch normalization were added

The last layer was a softmax with 3 outputs as we have 3 categories

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 14, 16)            3200

 bidirectional_3 (Bidirecti  (None, 64)                12544
 onal)

 dropout_3 (Dropout)         (None, 64)                0

 batch_normalization_3 (Bat  (None, 64)                256
 chNormalization)

 dense_3 (Dense)             (None, 3)                 195

=================================================================
Total params: 16195 (63.26 KB)
Trainable params: 16067 (62.76 KB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

Hyper-parameter tuning

These were the experiments done during hyper parameter tuning:

- **n_lstm (number of neurons in the LSTM layer)**: Started with 50, but reduced to 32 considering the simplicity of the task and to avoid overfitting
- **l2_penalty (Value of penalty applied in L2 regularizatoin)**: Started with 0.01 but incresed to 0.02 to avoid overfitting
- **drop_lstm (% of drop in Dropout layer after LSTM layer)**: Started with 0.15 but increased to 0.2 to avoid overfitting
- **num_epochs**: Set at 30 but early stopping always stopped training before 25 epochs.
- **early_stopping_patience**: Started at 3 but reduced to 2 to avoid overfitting.
- Sparse categorical cross-entropy was used as the loss function since our labels were encoded as integers

```
Epoch 1/30
30/30 - 7s - loss: 1.9738 - accuracy: 0.6122 - val_loss: 1.9542 - val_accuracy: 0.4538 - 7s/epoch - 238ms/step
Epoch 2/30
30/30 - 0s - loss: 1.3256 - accuracy: 0.7924 - val_loss: 1.6688 - val_accuracy: 0.5126 - 445ms/epoch - 15ms/step
Epoch 3/30
30/30 - 0s - loss: 0.9199 - accuracy: 0.8830 - val_loss: 1.4655 - val_accuracy: 0.4034 - 467ms/epoch - 16ms/step
Epoch 4/30
30/30 - 0s - loss: 0.6579 - accuracy: 0.9189 - val_loss: 1.3093 - val_accuracy: 0.6134 - 461ms/epoch - 15ms/step
Epoch 5/30
30/30 - 0s - loss: 0.5254 - accuracy: 0.9210 - val_loss: 1.1716 - val_accuracy: 0.8782 - 481ms/epoch - 16ms/step
Epoch 6/30
30/30 - 0s - loss: 0.4332 - accuracy: 0.9315 - val_loss: 1.0543 - val_accuracy: 0.8782 - 480ms/epoch - 16ms/step
Epoch 7/30
30/30 - 0s - loss: 0.3745 - accuracy: 0.9336 - val_loss: 0.9715 - val_accuracy: 0.8403 - 463ms/epoch - 15ms/step
Epoch 8/30
30/30 - 0s - loss: 0.3348 - accuracy: 0.9305 - val_loss: 0.8525 - val_accuracy: 0.8025 - 470ms/epoch - 16ms/step
Epoch 9/30
30/30 - 0s - loss: 0.2879 - accuracy: 0.9473 - val_loss: 0.8174 - val_accuracy: 0.8067 - 461ms/epoch - 15ms/step
Epoch 10/30
30/30 - 0s - loss: 0.2587 - accuracy: 0.9484 - val_loss: 0.7000 - val_accuracy: 0.8151 - 467ms/epoch - 16ms/step
Epoch 11/30
30/30 - 0s - loss: 0.2504 - accuracy: 0.9399 - val_loss: 0.5267 - val_accuracy: 0.9034 - 459ms/epoch - 15ms/step
Epoch 12/30
30/30 - 1s - loss: 0.2383 - accuracy: 0.9420 - val_loss: 0.4968 - val_accuracy: 0.8782 - 505ms/epoch - 17ms/step
Epoch 13/30
30/30 - 0s - loss: 0.2143 - accuracy: 0.9463 - val_loss: 0.4279 - val_accuracy: 0.8697 - 458ms/epoch - 15ms/step
Epoch 14/30
30/30 - 0s - loss: 0.2101 - accuracy: 0.9505 - val_loss: 0.7198 - val_accuracy: 0.7353 - 489ms/epoch - 16ms/step
Epoch 15/30
30/30 - 0s - loss: 0.2072 - accuracy: 0.9547 - val_loss: 0.3293 - val_accuracy: 0.8950 - 481ms/epoch - 16ms/step
Epoch 16/30
30/30 - 0s - loss: 0.1780 - accuracy: 0.9621 - val_loss: 0.3192 - val_accuracy: 0.8992 - 482ms/epoch - 16ms/step
Epoch 17/30
30/30 - 0s - loss: 0.1727 - accuracy: 0.9610 - val_loss: 0.2999 - val_accuracy: 0.9076 - 470ms/epoch - 16ms/step
Epoch 18/30
30/30 - 0s - loss: 0.1791 - accuracy: 0.9621 - val_loss: 0.3541 - val_accuracy: 0.8697 - 446ms/epoch - 15ms/step
Epoch 19/30
30/30 - 0s - loss: 0.1664 - accuracy: 0.9652 - val_loss: 0.3750 - val_accuracy: 0.8613 - 493ms/epoch - 16ms/step
```

Predicting sentences

```
[44] # Example sentences that should be predicted as Situation
     predict_category(["Customers started being affected by a bug due to our most recent deploy"], verbose=True)
     predict_category(["One of my co-workers was set in his ways ansd was losing his patience in the discussion with the other technical team"], verbose=True)

     1/1 [==============================] - 1s 805ms/step
     Prediction array: [[0.941909   0.04496868 0.01312219]]
     Predicted class: Situation
     1/1 [==============================] - 0s 24ms/step
     Prediction array: [[0.97127837 0.01434894 0.01437267]]
     Predicted class: Situation
     'Situation'

[45] # Example sentences that should be predicted as Action
     predict_category(["I promptly opened an incident and alerted all parties involved"], verbose=True)
     predict_category(["I reminded everyone that we were in the same team and that we each can contribute with different perspectives"], verbose=True)

     1/1 [==============================] - 0s 22ms/step
     Prediction array: [[5.1926469e-05 9.9965549e-01 2.9252836e-04]]
     Predicted class: Action
     1/1 [==============================] - 0s 24ms/step
     Prediction array: [[8.5002906e-04 9.9828488e-01 8.6506392e-04]]
     Predicted class: Action
     'Action'

     # Example sentences that should be predicted as Result
     predict_category(["My team's initiative saved the company 1 million dollars per year"], verbose=True)
     predict_category(["In the end, the project saved the company 1 million dollers per year"], verbose=True)

     1/1 [==============================] - 0s 23ms/step
     Prediction array: [[0.41846022 0.08887616 0.49266368]]
     Predicted class: Result
     1/1 [==============================] - 0s 21ms/step
     Prediction array: [[0.12978202 0.01093934 0.8592786 ]]
     Predicted class: Result
     'Result'
```

Analyzing interview answers

- Receive the answer as a text
- Break down the text in sentences
- Classify each sentence in Situation, Action or Result
- Verify if the answer contains at least one of each category
- Points out missing category if there are any

```
[51] # Should be judged as a complete answer
     answer1 = "There was an issue with supply chain. I analyzed data and provided my boss two options and they decided to dig into one area, ordering. Based on my findings and the work we did,

     analyse_answer(answer1)
```

```
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
Sentence: There was an issue with supply chain.; Category: Situation
Sentence: I analyzed data and provided my boss two options and they decided to dig into one area, ordering.; Category: Action
Sentence: Based on my findings and the work we did, the turnaround time from ordering all the way to shipping reduced from 2 weeks to 5 days.; Category: Result
All categories present in answer. The answer follows the SAR format
```

```
# Lacking a Result
answer2 = "Revenue was down in the quarter. I analyzed revenue streams for the previous six months to identify bottlenecks or gaps in our sales cycle."

analyse_answer(answer2)
```

```
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 138ms/step
Sentence: Revenue was down in the quarter.; Category: Result
Sentence: I analyzed revenue streams for the previous six months to identify bottlenecks or gaps in our sales cycle.; Category: Action
The following category/categories are missing from the answer: ['Situation']
```

+ Code + Text

Conclusion

Resource utilization: The model is very lightweight, and the dataset is not huge. Only CPU was enough

Nexts steps:

- **Improve dataset**: Get a larger dataset with a bigger variety of sentences, encapsulating different ways of writing
- **Make model more complex:** With a richer dataset, it is possible to add more layers and more neurons to each layer to capture more patterns without overfitting.
- **Capture the order of the sentences**: A sentence might be a Situation or Result if it is placed in the beginning or the and of a complete answer. Currently, the model can only evaluate each sentence separately, but analyzing the order of the sentences would improve the model a lot.

Lessons learned

A dataset that is too small makes every sentence very important, especially if you are training an LSTM from scratch

LLMs are very efficient tools for helping in generating datasets. However, it is necessary to be wary that the sentences produced might be too similar or might all be written in a similar "manner of speech". This might heavily affect your model.

Semantics like the difference between a Situation in a Result can be quite subjective, even for humans. However, this Bi-Directional LSTM did a good job of capturing these nuances given the circumstances.