Abstract Segmentation Final Project

Christian Martin, Awais Naeem, Allison Pujol

Problem Statement

Goal = to improve the readability and structure of “tricky” abstracts

Audience = medical students and researchers in the medical field; other researchers as well

Purpose = reading an abstract helps a researcher determine whether a lengthy paper may be worth reading

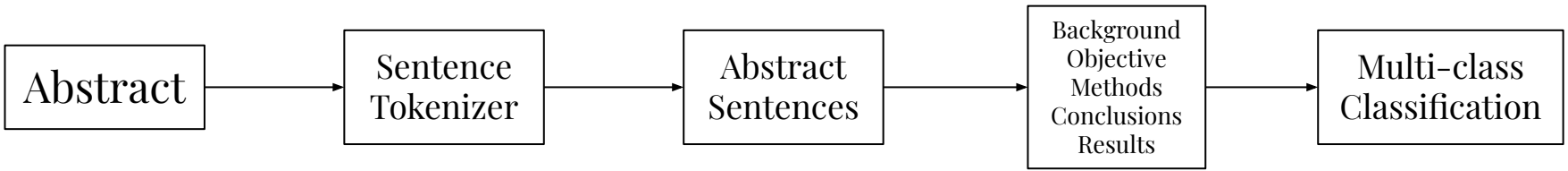
Context = many medical research abstracts use highly technical language that slows down overall readability and sentence comprehension; we may also want to “filter” by section of abstract to compare different methods or results

Approach towards the Problem Statement

Abstract Segmentation

- Background
- Objective
- Methods
- Conclusions
- Results

It's a simple Sequence Classification problem...



Dataset

- **200,000** Medical Research Paper Abstracts from PubMed (Kaggle Dataset)
- Abstract splitted into sentences with a distinct label assigned to each sentence i.e., *“background”, “objective”, “methods”, “results”, “conclusions”*

###20497432
BACKGROUND The aim of this study was to evaluate the efficacy , safety and complications of orbital steroid injection versus oral steroid therapy in the management of thyroid-related ophthalmopathy .
METHODS A total of 29 patients suffering from thyroid ophthalmopathy were included in this study .
METHODS Patients were randomized into two groups : group I included 15 patients treated with oral prednisolone and group II included 14 patients treated with peribulbar triamcinolone orbital injection .
METHODS Only 12 patients in both groups (16 female and 8 male) completed the study .
RESULTS Both groups showed improvement in symptoms and in clinical evidence of inflammation with improvement of eye movement and proptosis in most cases .
RESULTS Mean exophthalmometry value before treatment was 22.6 1.98 mm that decreased to 18.6 0.996 mm in group I , compared with 23 1.86 mm that decreased to 19.08 1.16 mm in group II .
RESULTS Mean initial clinical activity score was 4.75 1.2 and 5 1.3 for group I and group II before treatment , respectively , which dropped to 0.83 1.2 and 0.83 1.02 , 6 months after treatment , respectively .
RESULTS There was no change in the best-corrected visual acuity in both groups .
RESULTS There was an increase in body weight , blood sugar , blood pressure and gastritis in group I in 66.7 % , 33.3 % , 50 % and 75 % , respectively , compared with 0 % , 0 % , 8.3 % and 8.3 % in group II .
RESULTS No adverse local side effects were observed in group II .
CONCLUSIONS Orbital steroid injection for thyroid-related ophthalmopathy is effective and safe .
CONCLUSIONS It eliminates the adverse reactions associated with oral corticosteroid use .

Dataset Preprocessing

- Initial cleanup to remove blank lines and unwanted comments
- Regular expression based filtering to get *Abstract Name, Text and Text Label*
- Tokenize the text and filter the ones having **5** or more tokens

	Abstract Name	Text	Label
0	###24491034	The emergence of HIV as a chronic condition me...	BACKGROUND
1	###24491034	This paper describes the design and evaluation...	BACKGROUND
2	###24491034	This study is designed as a randomised control...	METHODS
3	###24491034	The intervention group will participate in the...	METHODS
4	###24491034	The program is based on self-efficacy theory a...	METHODS

- Originally: ~**200k** abstracts
- Post-Preprocessing: ~**1 million** labelled texts
  - Splitted into Train, Test and Validation datasets

Proposed Methods

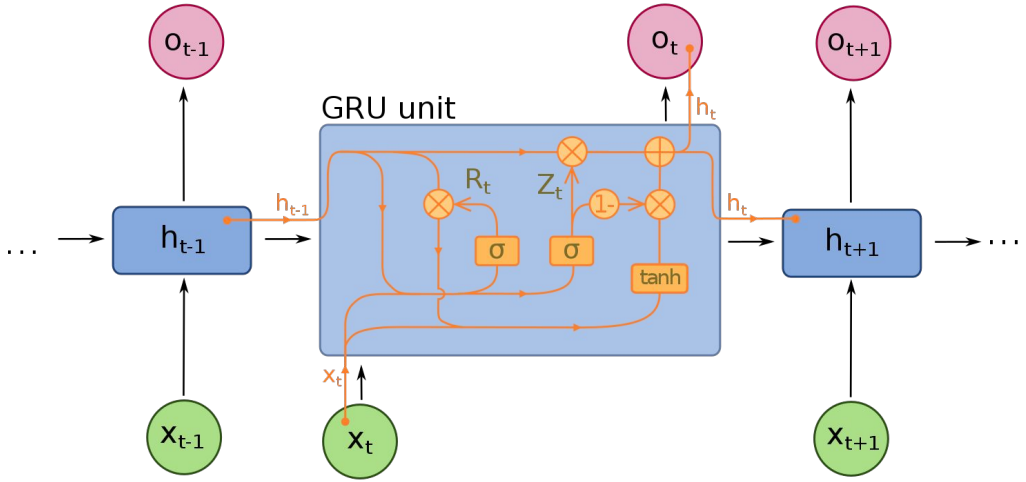
A. Gated Recurrent Unit (GRU) Neural Network
B. Fine-tuned LLM Model
C. Word2Vec + Logistic Regression

Gated Recurrent Unit (GRU) - Data Preparation

- Indexed Vocabulary of unique words using training data
  - Vocabulary Size: ~300k
  - Max Sequence Length: 338
- Text data padded or truncated to max sequence length
- Labels/Text data converted to PyTorch tensors
- Custom data loaders to read *text data* and *output labels*
  - Apply transforms to *text* and *labels*
  - Train Batch: 2048
  - Test Batch: 64
  - Validation Batch: 64

Gated Recurrent Unit (GRU) - Network Architecture

1. Embedding Layer
   a. Input Size: 300k
   b. Embedding Size: 200
2. GRU Layer
   a. Hidden Size: 64
   b. Num Layers: 2
3. FC Layer 1
   a. Input: 64
   b. Output: 32
4. FC Layer 2
   a. Input: 32
   b. Output: 16
5. FC Layer 3
   a. Input: 16
   b. Output: 5 (number of classes)



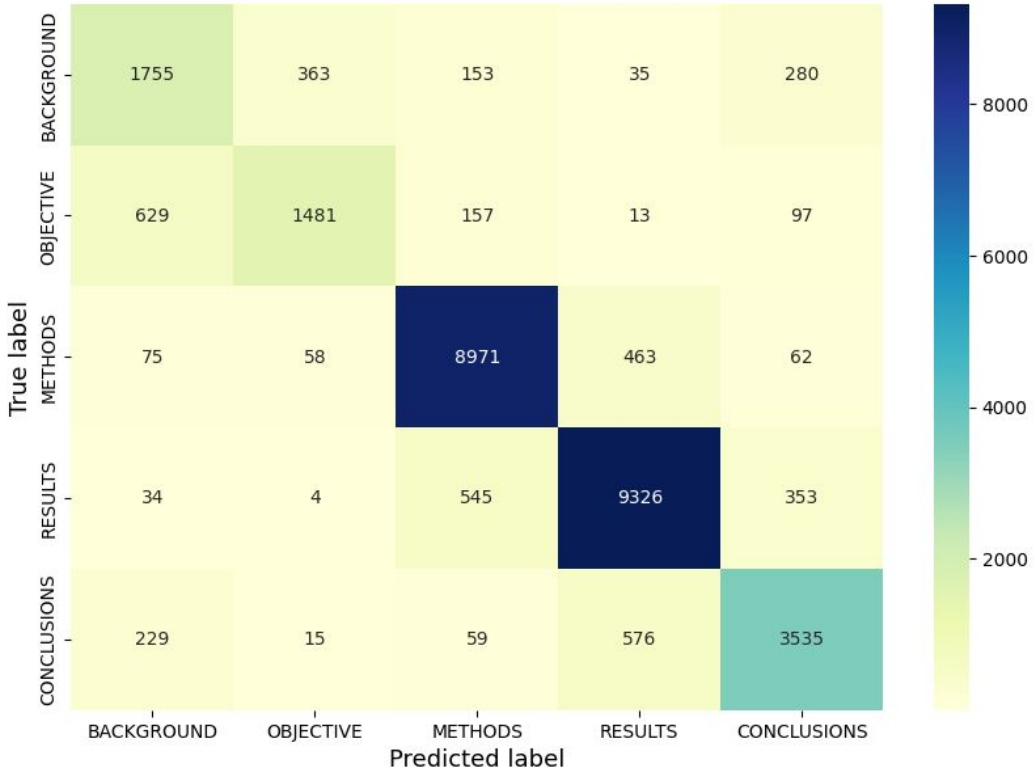
Gated Recurrent Unit (GRU) – Training

- Adam Optimizer
  - Learning Rate: 0.001
- Cross Entropy Loss
  - Multi-Class Classification
  - 5 output labels
- Total Records: ~1 million
  - 5 Epochs
  - Batch Size: 2048
- Initial Loss: 0.5212
- Final Loss: 0.2789
- Training Time
  - Google Colab Pro (~4-5 hours)

```
Epoch [1/5], Step [361/1072], Loss: 0.5212
Epoch [1/5], Step [362/1072], Loss: 0.4864
Epoch [1/5], Step [363/1072], Loss: 0.4927
Epoch [1/5], Step [364/1072], Loss: 0.4783
Epoch [1/5], Step [365/1072], Loss: 0.4852
Epoch [1/5], Step [366/1072], Loss: 0.5271
Epoch [1/5], Step [367/1072], Loss: 0.5046
Epoch [1/5], Step [368/1072], Loss: 0.4932
Epoch [1/5], Step [369/1072], Loss: 0.4970
Epoch [1/5], Step [370/1072], Loss: 0.4820
Epoch [1/5], Step [371/1072], Loss: 0.4948
Epoch [1/5], Step [372/1072], Loss: 0.4758
Epoch [1/5], Step [373/1072], Loss: 0.4881
Epoch [1/5], Step [374/1072], Loss: 0.5133
Epoch [1/5], Step [375/1072], Loss: 0.5058
Epoch [1/5], Step [376/1072], Loss: 0.5078
Epoch [1/5], Step [377/1072], Loss: 0.5096
Epoch [1/5], Step [378/1072], Loss: 0.4714
Epoch [1/5], Step [379/1072], Loss: 0.5101
Epoch [1/5], Step [380/1072], Loss: 0.4617
Epoch [1/5], Step [381/1072], Loss: 0.4585
Epoch [1/5], Step [382/1072], Loss: 0.5125
Epoch [1/5], Step [383/1072], Loss: 0.5162
Epoch [1/5], Step [384/1072], Loss: 0.4753
...
Epoch [5/5], Step [1069/1072], Loss: 0.3203
Epoch [5/5], Step [1070/1072], Loss: 0.3370
Epoch [5/5], Step [1071/1072], Loss: 0.3040
Epoch [5/5], Step [1072/1072], Loss: 0.2789
```

Gated Recurrent Unit (GRU) – Results

	precision	recall	f1-score	support
BACKGROUND	0.64	0.68	0.66	2586
OBJECTIVE	0.77	0.62	0.69	2377
METHODS	0.91	0.93	0.92	9629
RESULTS	0.90	0.91	0.90	10262
CONCLUSION	0.82	0.80	0.81	4414
accuracy			**0.86**	29268
macro avg	0.81	0.79	0.80	29268
weighted avg	0.86	0.86	0.86	29268



Fine-Tuned LLM Model – Training

We fine-tuned several models to see the effectiveness of different approaches

The first approach, which used just the sentence itself, has a fairly simple setup of T5 with a custom task prefix:

```
from transformers import T5Tokenizer, T5ForSequenceClassification

tokenizer = T5Tokenizer.from_pretrained("t5-small")
model = T5ForSequenceClassification.from_pretrained("t5-small", num_labels=len(labels))
```

```
def prep(df):
  text = [f"{task_prefix}: {text}" for text in df['Text'].to_list()]
  inputs = [tokenizer(f"{task_prefix}: {text}", padding=True, truncation=True, return_tensors="pt").input_ids[0] for text in df['Text'].to_list()]
  y = [labels.index(y) for y in df['Label'].to_list()]
  return Dataset.from_dict({"input_ids": inputs, "text": text, "labels": y})
```

Fine-Tuned LLM Model – Training

We also trained three more models which received the full abstract text. T5 was used for one model, like before, and DistilBERT for the other two.

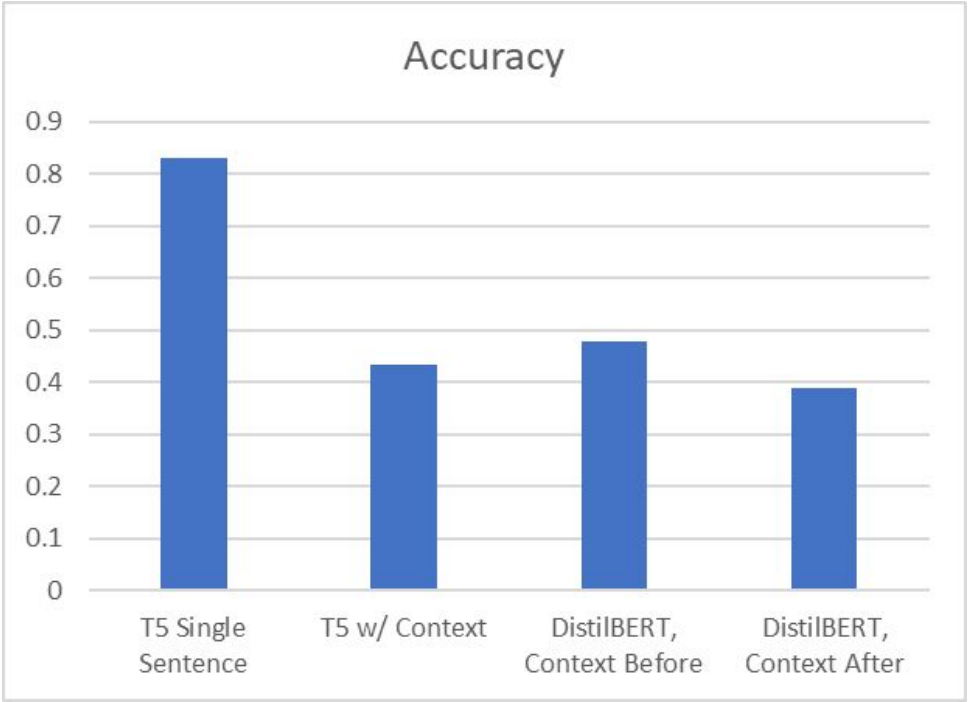
1. A T5 model which had the sentence, a “ | ” as a separator, then abstract
2. A DistilBERT model which had the sentence, “[SEP]” token, then abstract
3. A DistilBERT model which has the abstract, “[SEP]” token, then sentence

(Note that in DistilBERT, “[SEP]” is a special separator token; T5 does not implement this so an arbitrary “ | ” was used)

Fine-Tuned LLM Model – Results

Accuracies:

T5 Single Sentence	0.83
T5 + Context	0.43
DistilBERT, Context Before	0.48
DistilBERT, Context After	0.39



Fine-Tuned LLM Model – Results (Best vs Worst)

T5 Single Sentence - Confusion Matrix

TRUE \ PREDICTED	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	**1402**	483	483	479	39
CONCLUSIONS	298	**3292**	101	14	709
METHODS	71	51	**8944**	94	469
OBJECTIVE	468	172	174	**1546**	17
RESULTS	24	378	698	7	**9155**

DistilBERT, Context After - Confusion Matrix

TRUE \ PREDICTED	BACKGROUND	CONCLUSIONS	METHODS	OBJECTIVE	RESULTS
BACKGROUND	**530**	50	886	40	1080
CONCLUSIONS	358	**73**	2174	88	1721
METHODS	537	53	**6316**	142	2581
OBJECTIVE	191	17	1258	**135**	776
RESULTS	399	110	5282	176	**4295**

Word2Vec + Logistic Regression – Training

Finally, we trained a multi-class Logistic Regression using Word2Vec for creating average word vectors

```
# Build Word2Vec vocabulary
word2vec_model = Word2Vec(sentences=data['Text'].to_list(), vector_size=100, window=5, min_count=1, workers=4)
#word2vec_model.build_vocab(data['Text'])

# Train Word2Vec model
word2vec_model.train(data['Text'], total_examples=word2vec_model.corpus_count, epochs=10)  # Train the model
```

```
X = [average_word_vectors(words, word2vec_model, word2vec_model.wv.index_to_key, 100) for words in data['Text']]
y = data['Label']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Logistic Regression using Word2Vec embeddings
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)

# Predict on test set
y_pred = classifier.predict(X_test)
```

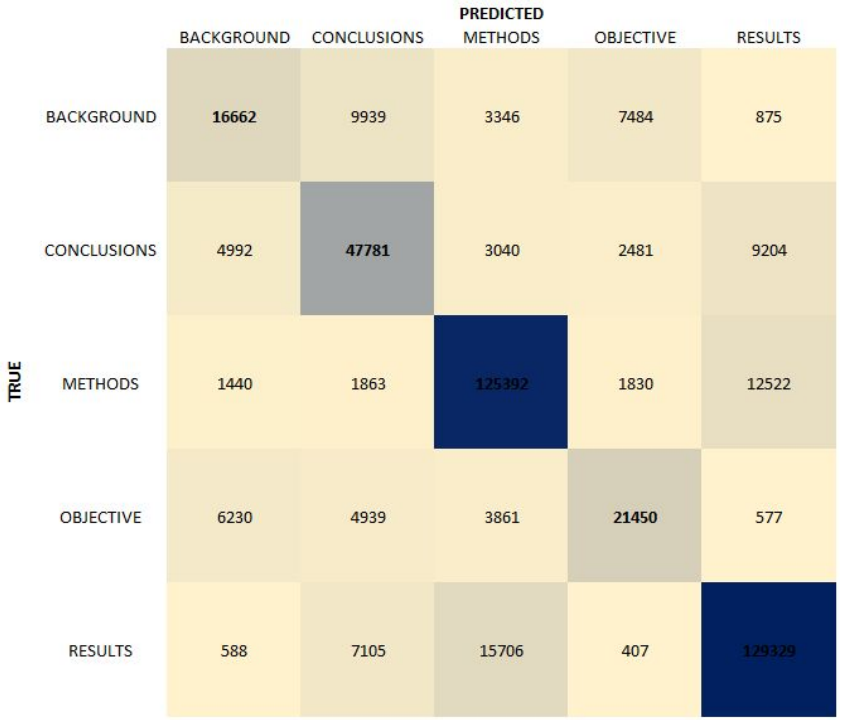
Word2Vec + Logistic Regression – Results

	precision	recall	f1-score	support
BACKGROUND	0.56	0.43	0.49	38306
CONCLUSIONS	0.67	0.71	0.69	67498
METHODS	0.83	0.88	0.85	143047
OBJECTIVE	0.64	0.58	0.61	37057
RESULTS	0.85	0.84	0.85	153135
accuracy			0.78	439043
macro avg	0.71	0.69	0.70	439043
weighted avg	0.77	0.78	0.77	439043

(Note: counts are different from the other models due to accidentally using a different test dataset)

BACKGROUND 56% precision = when classifier predicted a sentence as "BACKGROUNDS" it was correct 56% of the time

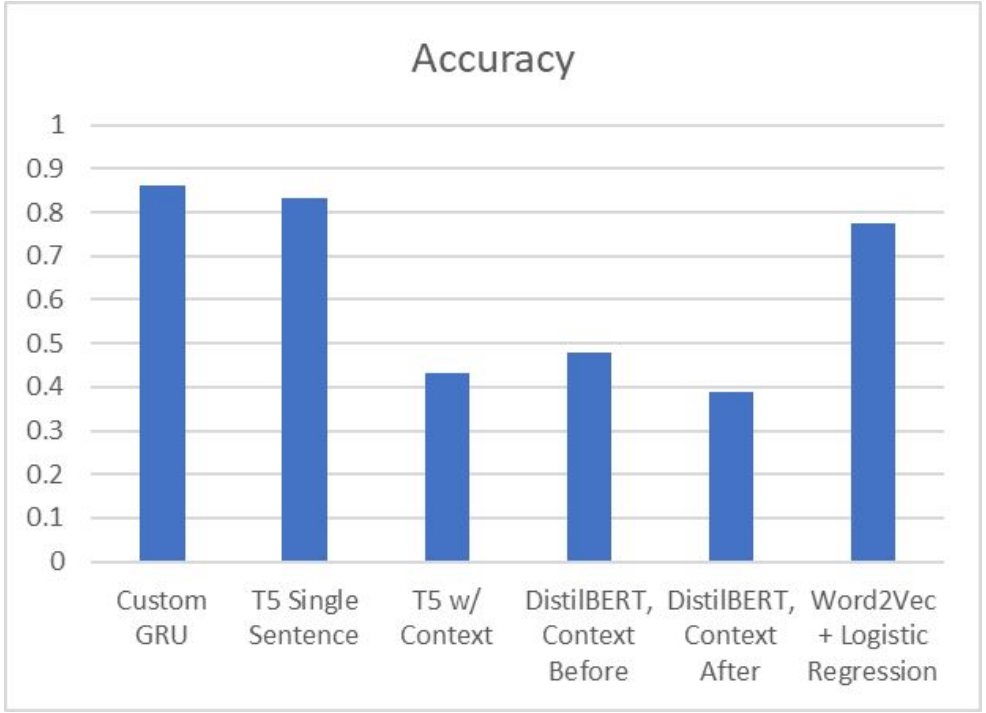
BACKGROUNDS 43% recall = classifier predicted correctly for 43% of the instances of BACKGROUNDS SENTENCES



Comparing the models

The custom GRU model performed best, with the single sentence T5 in close second.

The T5 & DistilBERT models with the full abstract context performed the worst, with about half the accuracy of the custom GRU & single sentence T5 models.



Comparing the models – Custom GRU vs T5 Single Sentence

Custom GRU model:

	precision	recall	f1-score	support
BACKGROUND	0.64	0.68	0.66	2586
OBJECTIVE	0.77	0.62	0.69	2377
METHODS	0.91	0.93	0.92	9629
RESULTS	0.90	0.91	0.90	10262
CONCLUSION	0.82	0.80	0.81	4414
accuracy			0.86	29268
macro avg	0.81	0.79	0.80	29268
weighted avg	0.86	0.86	0.86	29268

Fine-tuned T5 single sentence model:

	precision	recall	f1-score	support
BACKGROUND	0.62	0.54	0.58	2586
OBJECTIVE	0.72	0.65	0.68	2377
METHODS	0.89	0.93	0.91	9629
RESULTS	0.88	0.89	0.89	10262
CONCLUSIONS	0.75	0.75	0.75	4414
accuracy			0.83	29268
macro avg	0.77	0.75	0.76	29268
weighted avg	0.83	0.83	0.83	29268

Conclusion

- We were limited to predicting one label per sentence due to the dataset
  - We do recognize, however, there could be value in predicting multiple labels per sentence
- We were able to achieve good performance using relatively simple methods; using a custom GRU model allowed us get a little more performance
  - (+3% compared to single sentence T5 & +8% compared to Word2Vec + Logistic Regression)
- Trying to predict a sentence with its full abstract for context appears to confuse the model more than help
  - The model tends to over-predict methods & results labels, and under-predict objectives & conclusions labels.

DEMO

1. A desktop application developed using **Python Tkinter**
2. Take an abstract as input via UI (human input)
3. Sentence tokenizer to extract individual sentences
4. For each sentence
    a. Perform a forward pass through the trained model
    b. Assign the predicted label to the sentence
    c. Bucket the sentences having the same label in a separate list/string
5. Display the sentences assigned to each section (Background, Objective, Methods, Results, Conclusions) on the UI

Q and A: