



# Master Thesis update

Voice input based story generation

10.10.2020 by:

Rohit Keshav Bewoor (11011831)

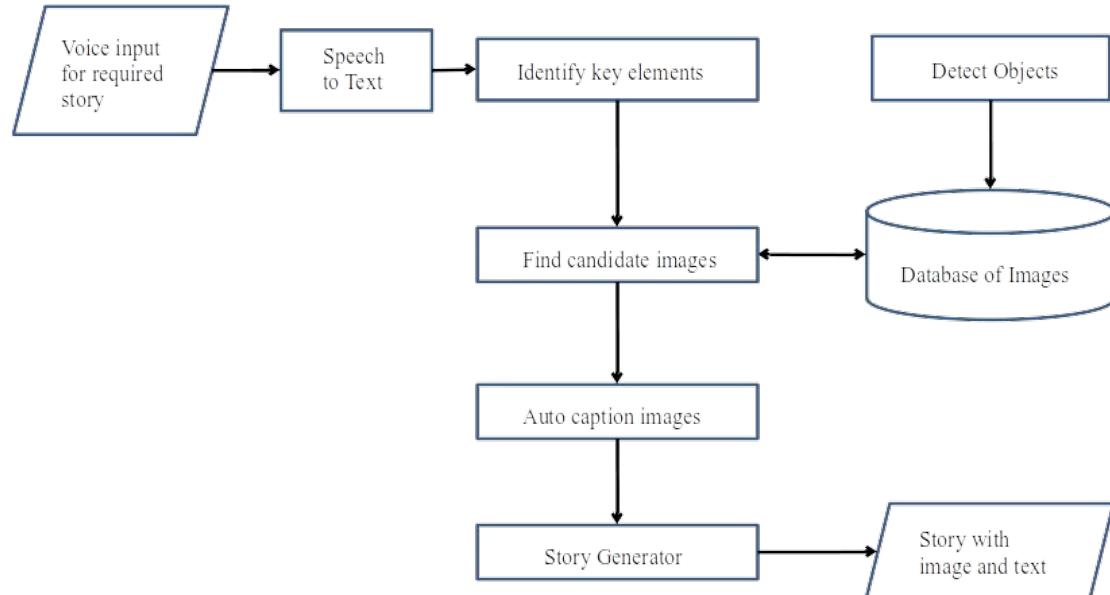
Big Data and Business Analytics 2018-20 batch  
SRH Hochschule Heidelberg

# Block Diagram

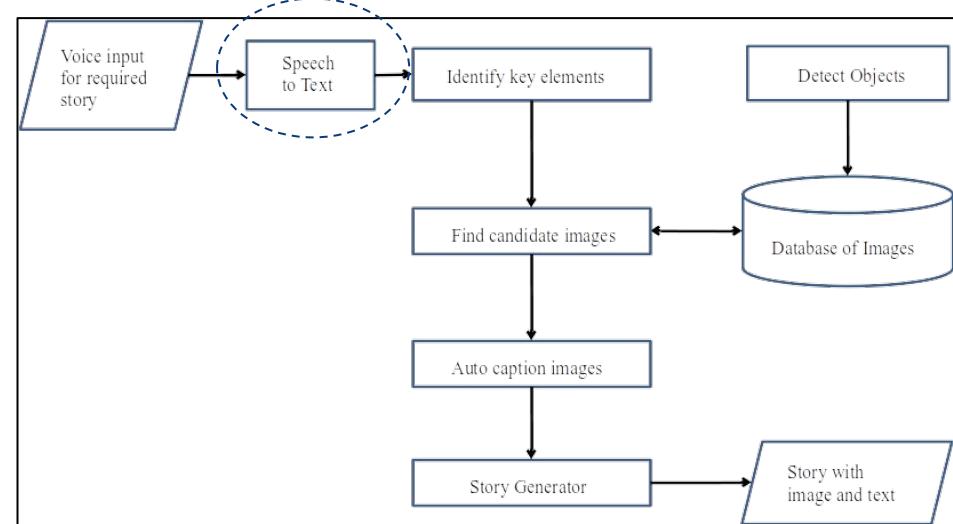
- **Goal:** Accept a voice input from user and use it to create a story with images selected from a database.

- Various neural networks required:

- > Model 1: Speech to Text (STT)
- > Model 2: Objection Detector and Database to be populated independently
- > Model 3: Auto caption of images
- > Model 4: Story generator



# Speech to Text Block



# Speech to Text

- **Goal:** Accept voice input and output a string for the transcription
  - > Currently, system accepts one pre-recorded wav file per input sentence.
  - > To explore if possible to capture the voice input directly from microphone, save it as a wav file for further processing.
- Microsoft Azure tried earlier
  - > Good accuracy
  - > But is paid service
- Decided to use open source STT implementation - pre-trained model of Deepspeech v0.7.3 from Mozilla foundation
  - > Link: <https://deepspeech.readthedocs.io/en/v0.7.3/DeepSpeech.html>
  - > RNN model with 5 hidden layers
  - > Accepts audio file as input and outputs text transcript
    - > Input Audio format: wav file, 16kHz sampling, mono channel, 16-bit
  - > Output characters set {a,b,c,...,z,apostrophe}

# Speech to Text

- Tested on my own voice with three audio files
- input1.wav transcript: Errors in output transcript
  - > I said:  
*Make me a story about persons sitting at a table. They are playing cards.*
  - > Model Inference:  
*me me a tory about persons sitting at table the blanchards*
- input2.wav transcript: No errors
  - > I said:  
*I want a story about a car on the road. A child plays with a toy.*
  - > Model Inference:  
*i want a story about a car on the road a child plays with a toy*
- input3.wav transcript: No errors
  - > I said:  
*Generate a story about persons walking on the street. A truck is on the road.*
  - > Model Inference:  
*generate a story about persons walking on the street a truck is on the road*

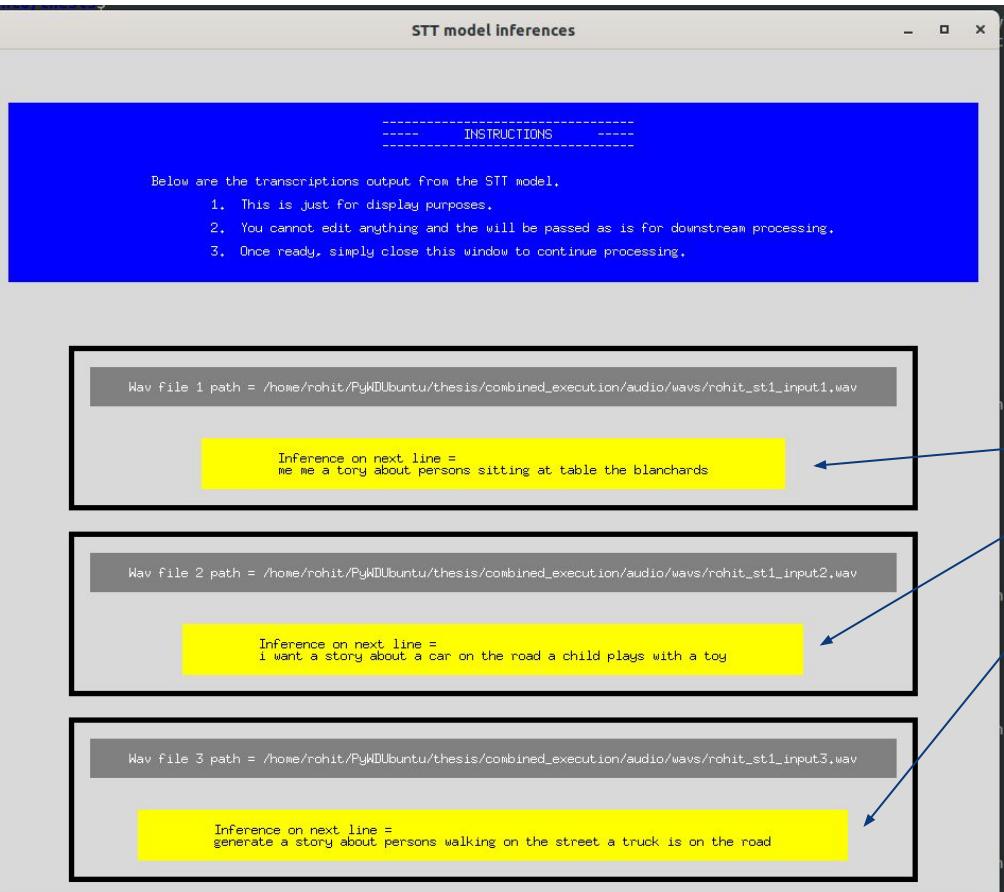


# Speech to Text

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Each wav file is processed and produces output of one long sentence.
  - > Note: Even if the recording actually consists of different sentences with a pause, the model will treat the input as one long sentence.
  - > If possible, later as part of NLP processing, attempt will be made to break a long transcript into sentences.
    - > **Did not find foolproof method to correctly identify individual sentences! So limiting user to speak one sentence per input audio file.**
- E.g. inference on input2.wav file and the transcript produced:
  - > I said:  
*I want a story about a car on the road. A child plays with a toy.*
  - > Model Inference:  
*i want a story about a car on the road a child plays with a toy*
  - > Two sentences spoken in recording, but output is combined sentence.
  - > How to break this reliably into two sentences again?

# Speech to Text - GUI

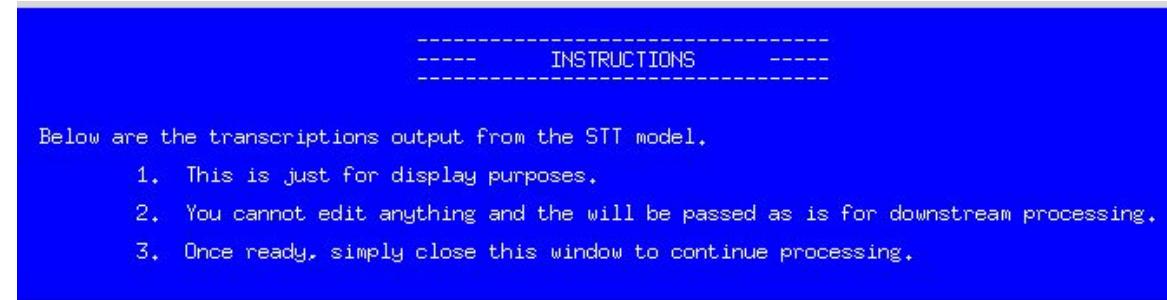


- GUI displays the output from STT
- Nothing editable
  - > Only for display purpose
  - > data is sent as is downstream
- The inferences for each query

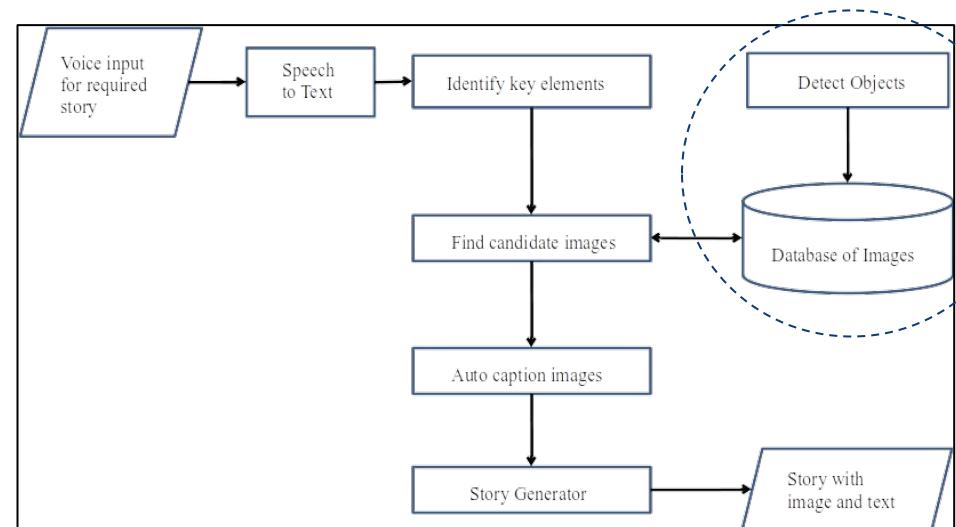
# Speech to Text - GUI

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Transcriptions and the instructions panels



# Database creation using Object Detection Block



# Object Database creation

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- **Goal:** Create a database storing information about the objects detected in the various images.
  - > Should be able to query a large database easily - around 100k images or more targeted
- Datasets currently identified:
  - > Common Objects in Context (COCO):  
<http://cocodataset.org/> , <http://cocodataset.org/#download>
    - > COCO Test 2017 = 41k images, 6GB size
    - > COCO Val 2017 = 5k images, 1GB size
    - > COCO Train 2017 = 118k images, 18GB size
  - > Flickr dataset
    - > 30k images, 4.5 GB size
- Object Detector used: You Only Look Once version 3 (YOLOv3)
  - > A single stage detector so fast compared to 2-stage detectors
  - > Implemented using parallel processing
  - > On personal laptop takes 5 minutes to process 127 images
- Discarded output inference image due to space constraints.

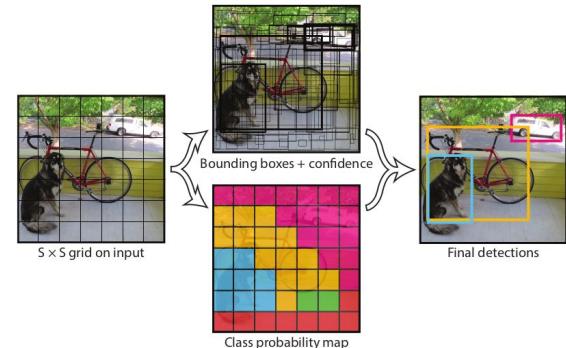


Neo4j database after population

# How YOLO works

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Divides image into a grid of size  $S \times S$  where  $S$  is an integer
- Each pixel evaluated as possible center point of an object
- All detections are evaluated in one pass - very fast algorithm
- Model is trained to identify  $C$  classes of objects
- $B$  is the number of Bounding Boxes detected all over the image (without threshold consideration). Five values are output for each bounding box:
  - Two values for center coordinates
  - Two values for dimensions (height and width)
  - Confidence score
- Can handle multiple bounding boxes and aspect ratios (anchor box concept)
  - Anchor boxes are predefined boxes provided by the user to Darknet which gives the network an idea about the relative position and dimensions of the objects to be detected.
  - These are calculated using the training set Objects.

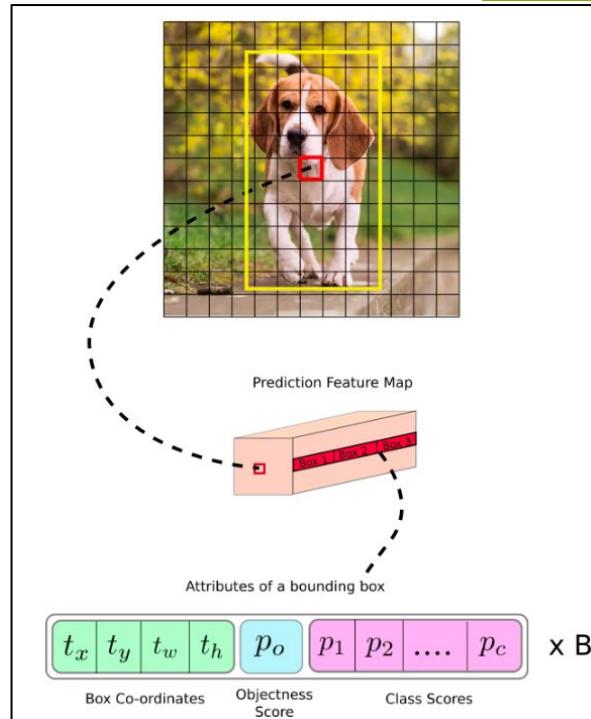


Source: YOLO v1 paper: <https://arxiv.org/abs/1506.02640>

# How YOLO works

STAATLICH  
ANERKANNTE  
HOCHSCHULE

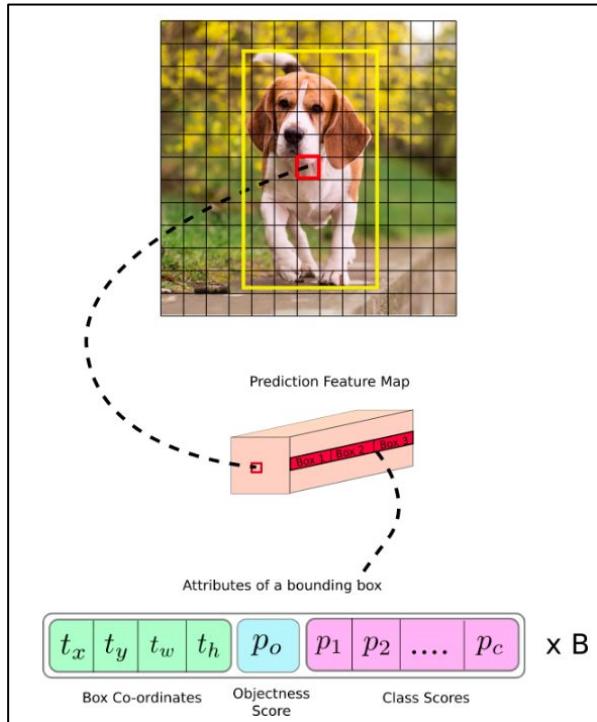
- Usually Non-max suppression used to remove redundant detections
- Total detections per image =  $( S \times S ) * ( B * ( 5 + C ) )$ 
  - Each bounding box has  $5 + C$  attributes
- For example, suppose that:
  - image is divided into  $13 \times 13$  grid (i.e.  $S = 13$ )
  - we want to detect 80 classes for COCO (i.e.  $C = 80$ )
  - 3 boxes predicted ( $B = 3$ )
  - #Detections =  $(13 \times 13) * (3 * (5 + 80)) = 13 \times 13 \times 255$   
 $= 43,095$
- Threshold value used for Confidence Score to evaluate acceptance of object detection



Source: <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193>

# How YOLO works

STAATLICH  
ANERKANNTE  
HOCHSCHULE



Source: <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193>

- Image divided into a grid of  $S \times S$  grid-boxes
  - Predictions made at 3 scales where a  $416 \times 416$  pixel input image is divided into  $13 \times 13$ ,  $26 \times 26$  and  $52 \times 52$
- Each grid can predict maximum  $B$  objects if it thinks object center lies in the pixels covered by the grid
- Model outputs regressed values for box information:  
 $t_x, t_y, t_w, t_h$ 
  - Mathematical formulae applied later to get actual bounding box center coordinates and dimensions:  $b_x, b_y, b_w, b_h$

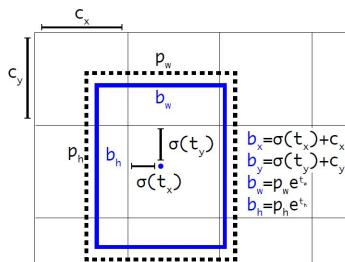


Figure 2. Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

Source: Yolo v3 paper: <https://arxiv.org/abs/1804.02767>

# Use Case for YOLO

STAATLICH  
ANERKANNTE  
HOCHSCHULE

**Goal:** Use the YOLOv3 model to perform inference and store information in Neo4j graph database

## Use Case:

Present a set of new images to a pre-trained YOLOv3 model.

For each image, capture the detected **object class** and the **confidence score**.

Store information in a neo4j graph database:

- Relationship format:  
 $( i:Image\{ name: "Image123.jpg", data: "dataset source" \} ) - [ r:HAS \{ score: confidence score \} ] -> ( o:Object \{ name: "object class" \} )$
- Image node's "data" property value identifies which dataset the image belongs to. E.g. coco\_train\_2017
- HAS relationship's "score" property value is the confidence score of object being present in the image

E.g. Image123.jpg HAS the objects:

- car (score 58.98%),
- person (score 98.34%)
- person (score 93.23%)

Note: Used a threshold score of 0.45 while populating the database. But during image querying later, will use higher threshold.

- if Object -> HAS[score > 0.45] -> Object :: only then insert into Neo4j db.

# Neo4j representation

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Neo4j db after inserts

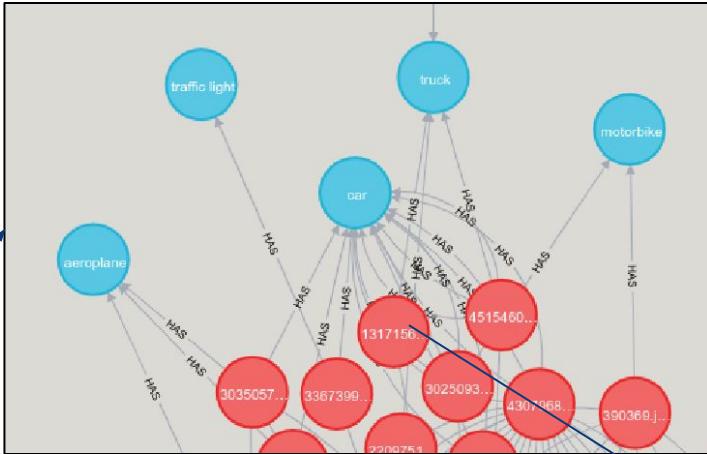
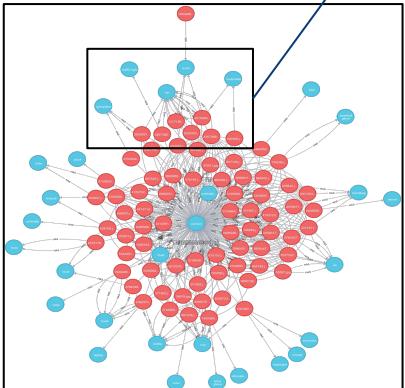


Image (red node) - HAS (line) -> Object (green node)

Detection output:  
1317156\_det.jpg

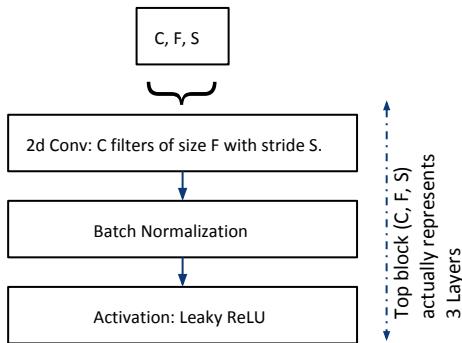
Objects found in images: Traffic light, truck, motorbike, car, etc.

Many to many relationship could exist.

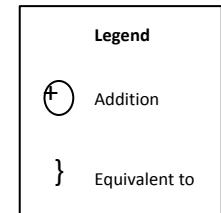
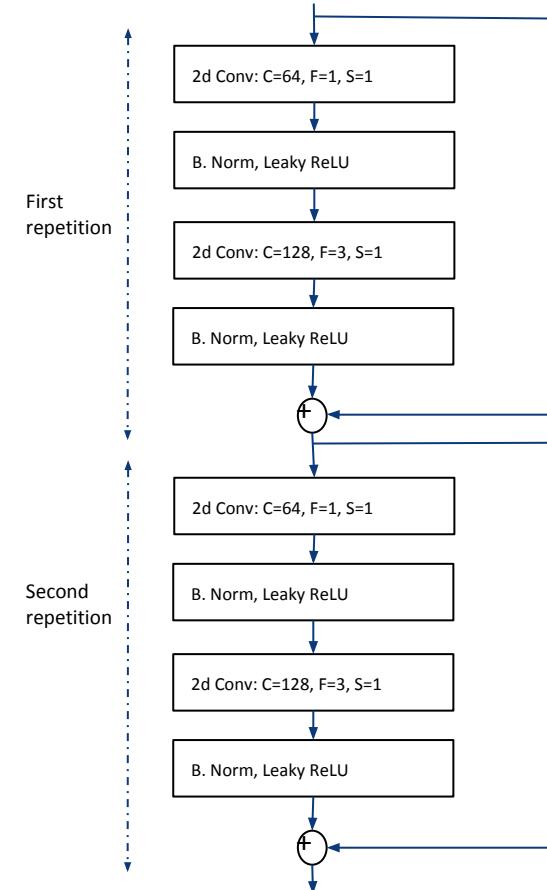
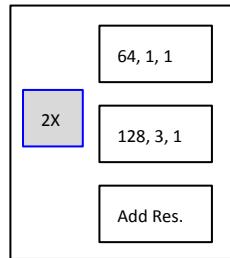


# Model description - elementary blocks

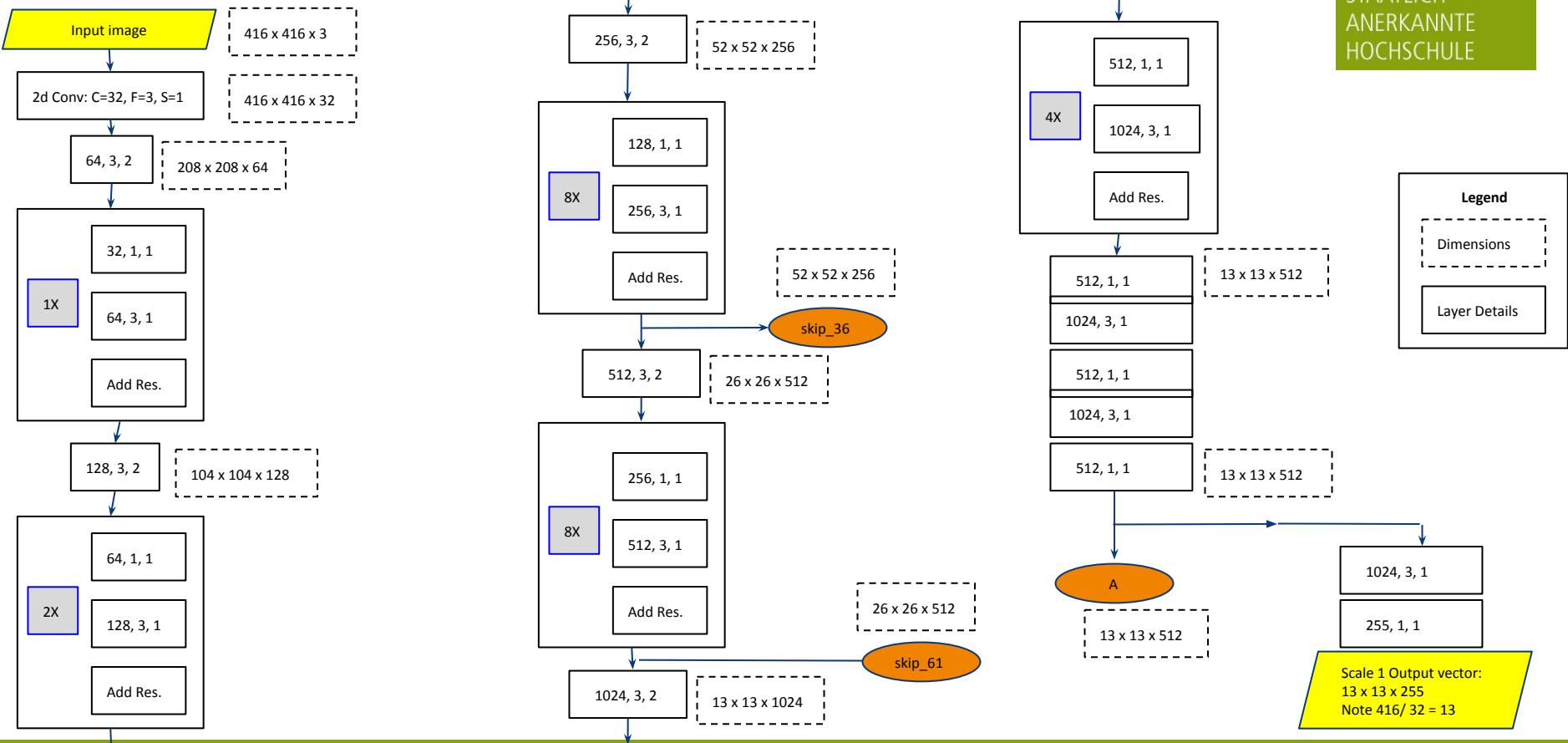
- Series of Convolution, then Batch Normalization, then Leaky ReLU. This is repeated many times.
- Convolution type:
  - Stride = 1, then use “same”
  - Stride = 2, then use “valid”
- Convolution parameters:
  - C: #Filters
  - F: Filter size
  - S: Stride



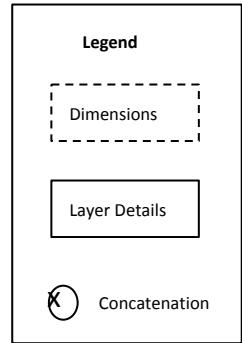
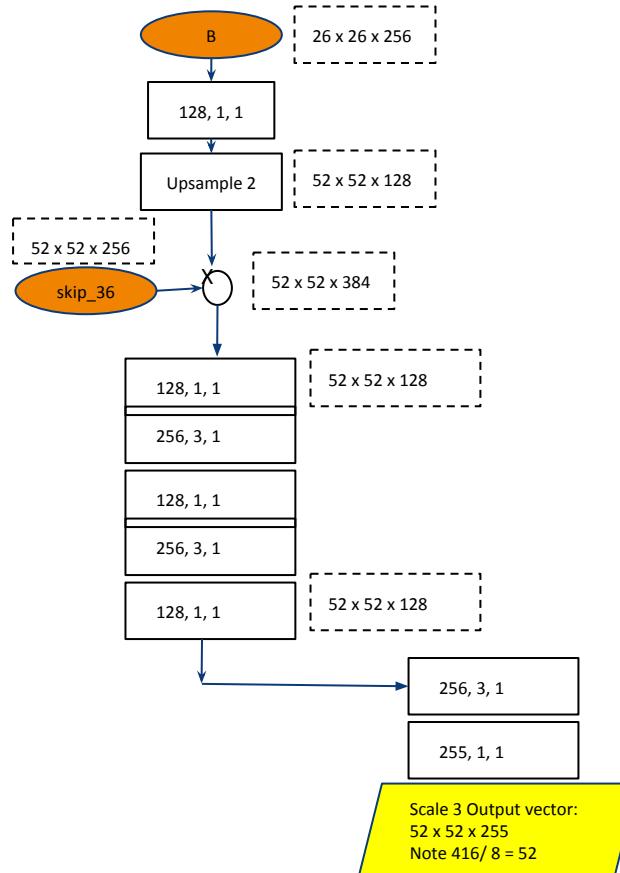
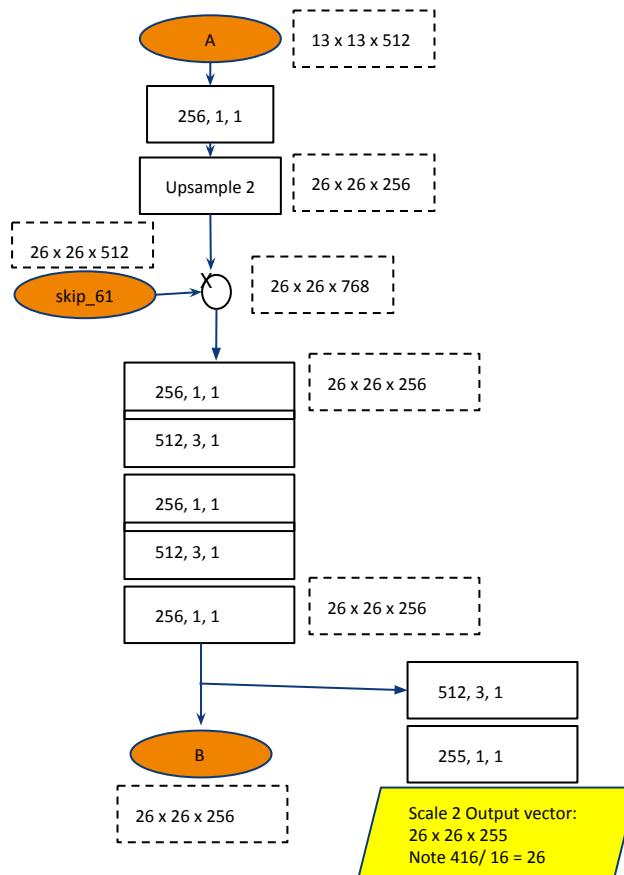
- The 2X means this set of operational blocks is repeated two times.



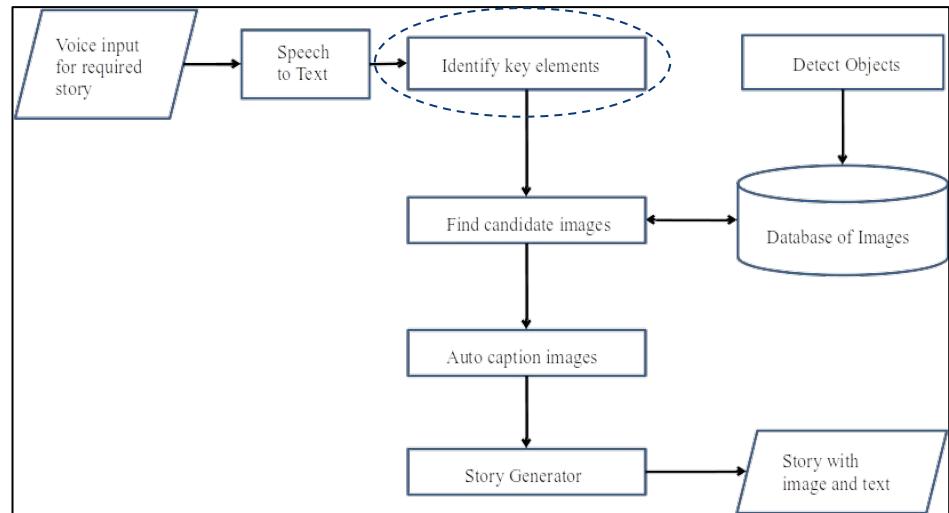
# Model description - i/p o/p feature sizes



# Model description - i/p o/p feature sizes



# Identify Key Elements Block



# Identify Key Elements

- **Goal:** Process the transcriptions from the STT block and figure out the Keywords to pass to the database querying stage.
- Tried using python modules: NLTK and Spacy. Found Spacy better.
  - > Spacy much faster and is production environment ready tool.
  - > Downside of Spacy is less inbuilt algorithms compared to NLTK.
    - > But found the POS tagging very exhaustive for Spacy and working well
  - > Using the “large” model: [https://spacy.io/models/en#en\\_core\\_web\\_lg](https://spacy.io/models/en#en_core_web_lg)
- Logic performed using Spacy:
  - > Word tokenization      Remove all stop-words      POS tagging
  - > Keeping only Noun type words of three tags as the candidate keywords (<https://spacy.io/api/annotation#pos-tagging>):
    - > Tag = “NN” - Noun, singular or mass
    - > Tag = “NNS” - Noun, plural
    - > Tag = “NNP” - Noun, noun proper singular
  - > Extracting the Lemma form of the word, not the original word itself
  - > E.g. Spacy information for the input word = “car”

```
results = {"text": "car", "lemma_": "car", "pos_": "NOUN", "tag_": "NN", "dep_": "compound", "shape_": "xxx", "is_alpha": true, "is_stop": false}
```
- From the set of words kept after inspecting the POS-Tag, retained only words that are “objects in the database”. Only these retained words are presented to user for selection via a GUI later.

# Identify Key Elements

- Sentence tokenization is not reliable.

> Input had two sentences from the output of STT block

```
In [14]: import spacy
nlp = spacy.load('en_core_web_lg')

In [15]: arr = ['i want a story about a car on the road a child plays with a toy',
           'generate a story about persons walking on the street a truck is on the road']

In [16]: doc = nlp("This is a sentence. This is another sentence.")
for sent in doc.sents:
    print(sent.text)

This is a sentence.
This is another sentence.

In [17]: doc = nlp(arr[0])
for sent in doc.sents:
    print(sent.text)

i want a story about a car on the road a child plays with a toy

In [18]: doc = nlp(arr[1])
for sent in doc.sents:
    print(sent.text)

generate a story about persons walking on the street
a truck is on the road
```

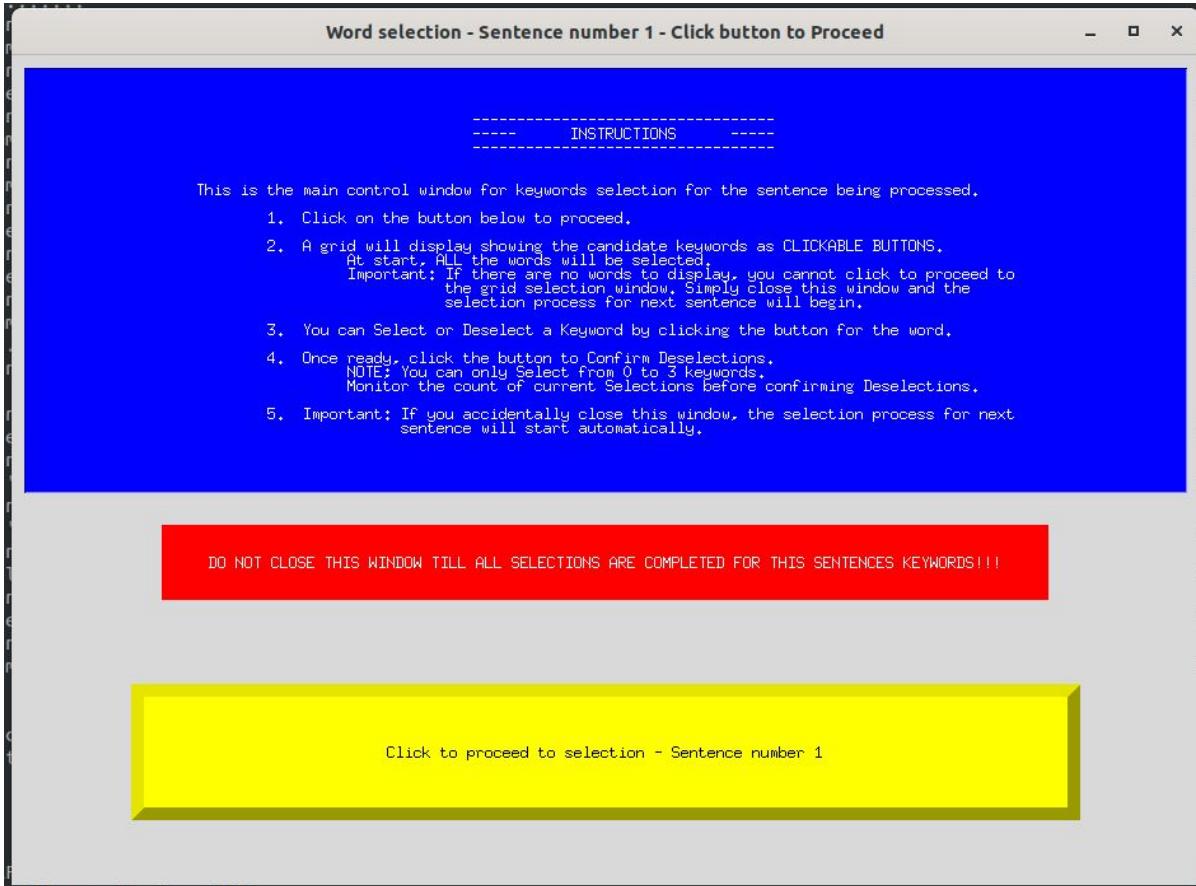
> Spacy: got one wrong  
got one right

> NLTK: got both wrong

```
In [9]: for entry in arr:
    result.append(sent_tokenize(entry))

In [10]: result
Out[10]: [['i want a story about a car on the road a child plays with a toy'],
           ['generate a story about persons walking on the street a truck is on the road']]
```

# Identify Key Elements - GUI flow



- Present candidate keywords via a GUI to user for final selection
- User to click on Yellow button to proceed
- If no words available for selection, button is disabled and user simply closes the window and moves to selection process for next sentence.
- From the candidate keywords of each of sentence, user must finally Select exactly 0 / 1 / 2 / 3 words.
- Note: logic already pre-checks the words and only presents matches against the current set of object class labels

# Identify Key Elements - GUI flow

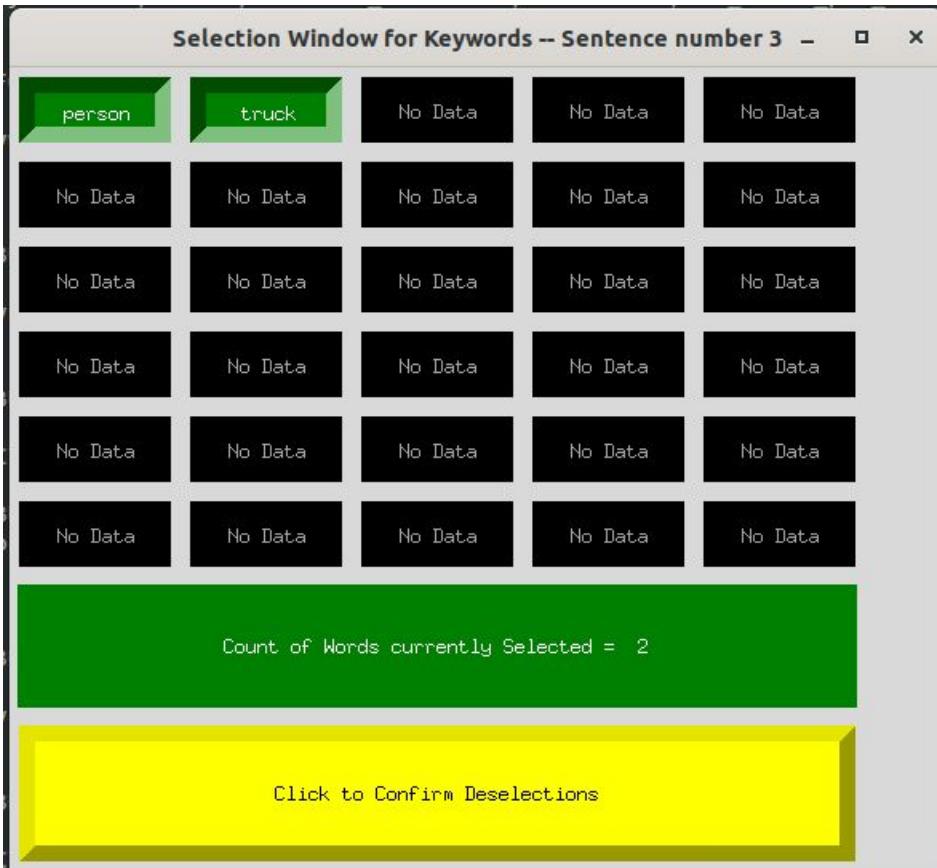
-----  
----- INSTRUCTIONS -----  
-----

This is the main control window for keywords selection for the sentence being processed.

1. Click on the button below to proceed.
2. A grid will display showing the candidate keywords as CLICKABLE BUTTONS.  
At start, ALL the words will be selected.  
Important: If there are no words to display, you cannot click to proceed to the grid selection window. Simply close this window and the selection process for next sentence will begin.
3. You can Select or Deselect a Keyword by clicking the button for the word.
4. Once ready, click the button to Confirm Deselections.  
NOTE: You can only Select from 0 to 3 keywords.  
Monitor the count of current Selections before confirming Deselections.
5. Important: If you accidentally close this window, the selection process for next sentence will start automatically.

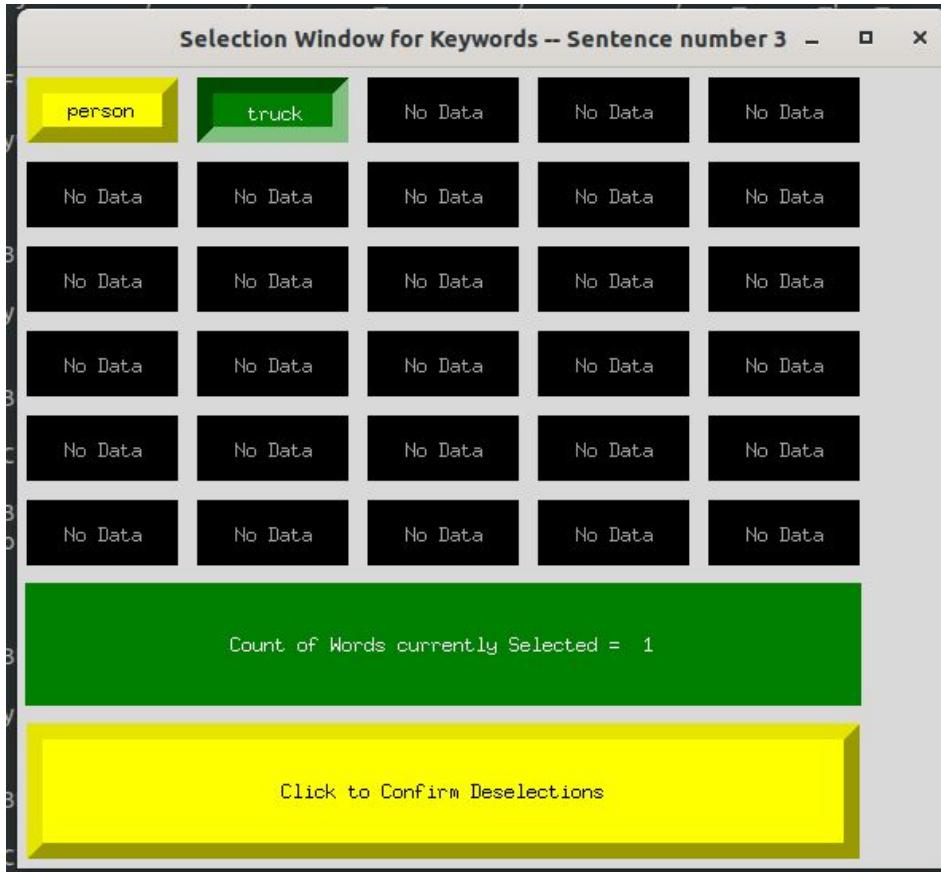
DO NOT CLOSE THIS WINDOW TILL ALL SELECTIONS ARE COMPLETED FOR THIS SENTENCES KEYWORDS!!!

# Identify Key Elements - GUI flow



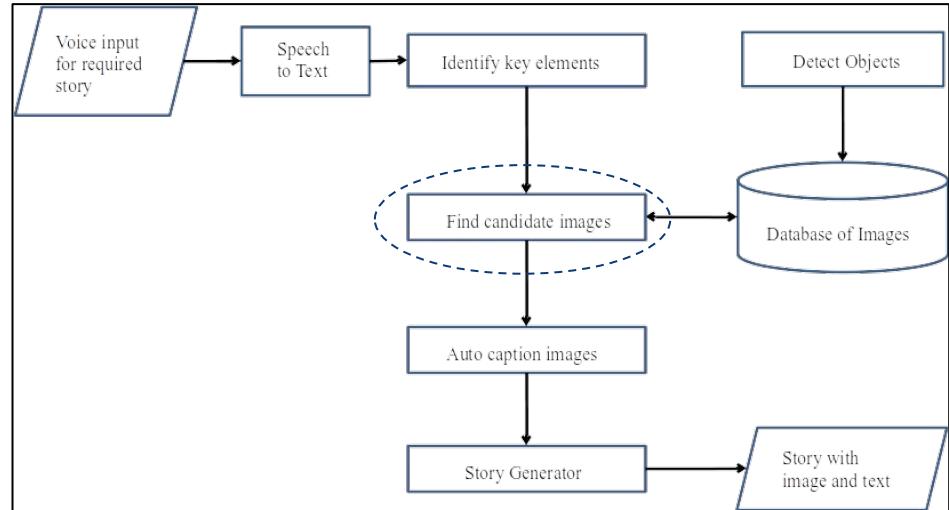
- Up to 30 candidate keywords possible
  - > non-stopwords of type noun which matched object class labels in Neo4j database
- Here 2 candidate keywords shown
- Remaining placeholders indicate No Data.
- All words Selected by default.
- Current count shown in button (value here = 2).
  - > If count > 3, then invalid and color would be Red.
  - > If color is Green, means valid count (0 to 3)
- Confirm Selections button is enabled

# Identify Key Elements - GUI flow



- User clicked word “person” to Deselect the word.
- Count is now 1
  - > Color remains Green
- User happy with the remaining selections, so clicks yellow button to confirm the current Deselections.
- Now only “truck” is passed on to Find Candidate Images block

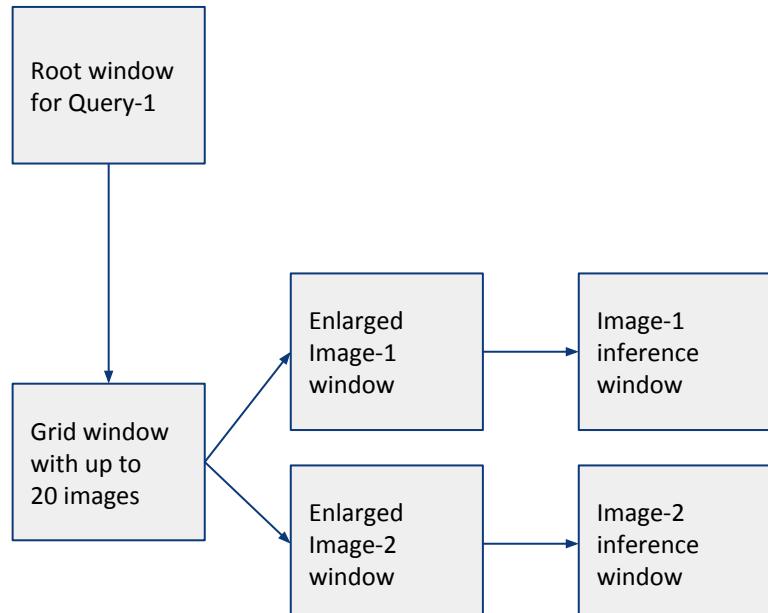
# Find Candidate Images Block



# Find Candidate Images

- **Goal:** With the user selected final keywords from previous stage, query the Neo4j database to retrieve images containing those objects of interest.
- Up to 20 images returned per query as “candidate images”.
  - > Only if image contains at least one object with a HAS relationship score > 90%
  - > Only from the COCO\_Test2017 and Flickr8k datasets
    - > As other datasets used for Yolo model training
- Allow user to view candidate images and Deselect any images:
  - > Misclassified images.
  - > Reduce the total Selections count to maximum 5 images.
- User selection via GUI interface:
  - > Images thumbnails shown.
    - > Clickable to “Select” or “Deselect”
  - > View Enlarged image for close inspection
  - > Perform one-off object detection inference
  - > Finally only keep the Selected images to pass on to the next stage i.e. Auto caption stage.
- GUI implemented using Tkinter python module

- Repeats for each of the three queries - once per input sentence
  - > If no images, does not allow grid window selection logic



# Find Candidate Images - wrong query results

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Example:
  - > Queries for objects “dog” and “cat”
  - > First image - wrong - has only dog
  - > Second image - correct - has both cat and dog
- Therefore allow user to deselect such wrong images
- Partial solution implemented::
  - > Used minimum threshold value of 90% during Neo4j query
  - > But is not foolproof

```
In [9]: for each_objects_list in input_list_for_query[:1]:
    print(f"\n\n{each_objects_list}:\n{query_neo4j_db(each_objects_list, 10)})")
```

```
['dog', 'cat']:
(0, [{'i.name': '000000183838.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000313767.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000341174.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000302015.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000101257.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000430186.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000310779.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000017269.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000210056.jpg', 'i.dataset': 'coco_test_2017'}])
```

```
In [12]: ## has only dog, no cat - wrong image retrieved by query
imgname = r'000000183838.jpg'
img = cv2.imread(cocoTest + imgname, cv2.IMREAD_COLOR)
#img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # cv2.IMREAD_GRAYSCALE
#img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) # cv2.IMREAD_COLOR
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) # cv2.COLOR_BGR2RGB
cv2.imshow('img', img)
plt.imshow(img)
```

```
Out[12]: <matplotlib.image.AxesImage at 0x7fa07c06d20>
```

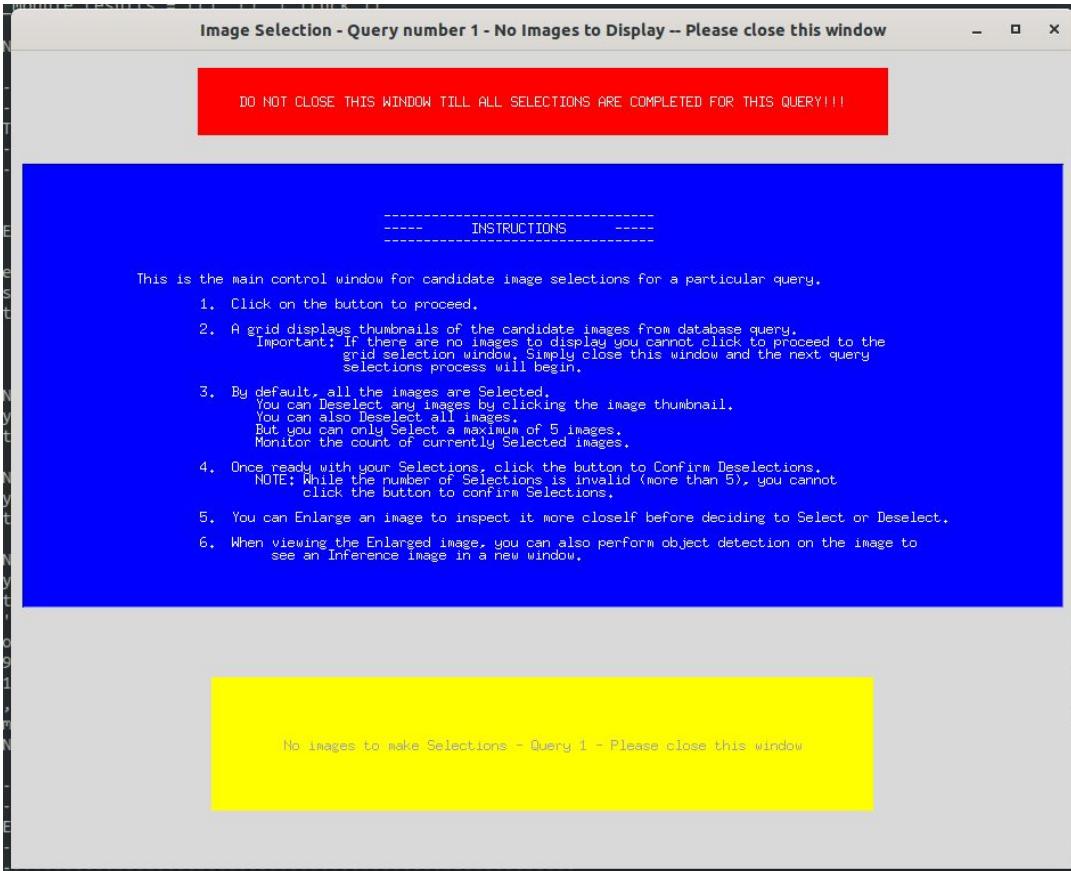


```
In [13]: ## has both dog and cat - correct image retrieved by query
imgname = r'000000313767.jpg'
img = cv2.imread(cocoTest + imgname, cv2.IMREAD_COLOR)
#img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # cv2.IMREAD_GRAYSCALE
#img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) # cv2.IMREAD_COLOR
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) # cv2.COLOR_BGR2RGB
cv2.imshow('img', img)
plt.imshow(img)
```

```
Out[13]: <matplotlib.image.AxesImage at 0x7fa05ab42310>
```



# Find Candidate Images - GUI flow



- Database query for this sentences' keywords returned no hits.
  - > Cannot click to proceed to grid selection.
  - > Close window and proceed to selection process for next query results

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

This is the main control window for candidate image selections for a particular query.

1. Click on the button to proceed.
2. A grid displays thumbnails of the candidate images from database query.  
*Important:* If there are no images to display you cannot click to proceed to the grid selection window. Simply close this window and the next query selections process will begin.
3. By default, all the images are Selected.  
You can Deselect any images by clicking the image thumbnail.  
You can also Deselect all images.  
But you can only Select a maximum of 5 images.  
Monitor the count of currently Selected images.
4. Once ready with your Selections, click the button to Confirm Deselections.  
*NOTE:* While the number of Selections is invalid (more than 5), you cannot click the button to confirm Selections.
5. You can Enlarge an image to inspect it more closely before deciding to Select or Deselect.
6. When viewing the Enlarged image, you can also perform object detection on the image to see an Inference image in a new window.

- User to simply close window and proceed to selection process for next query results

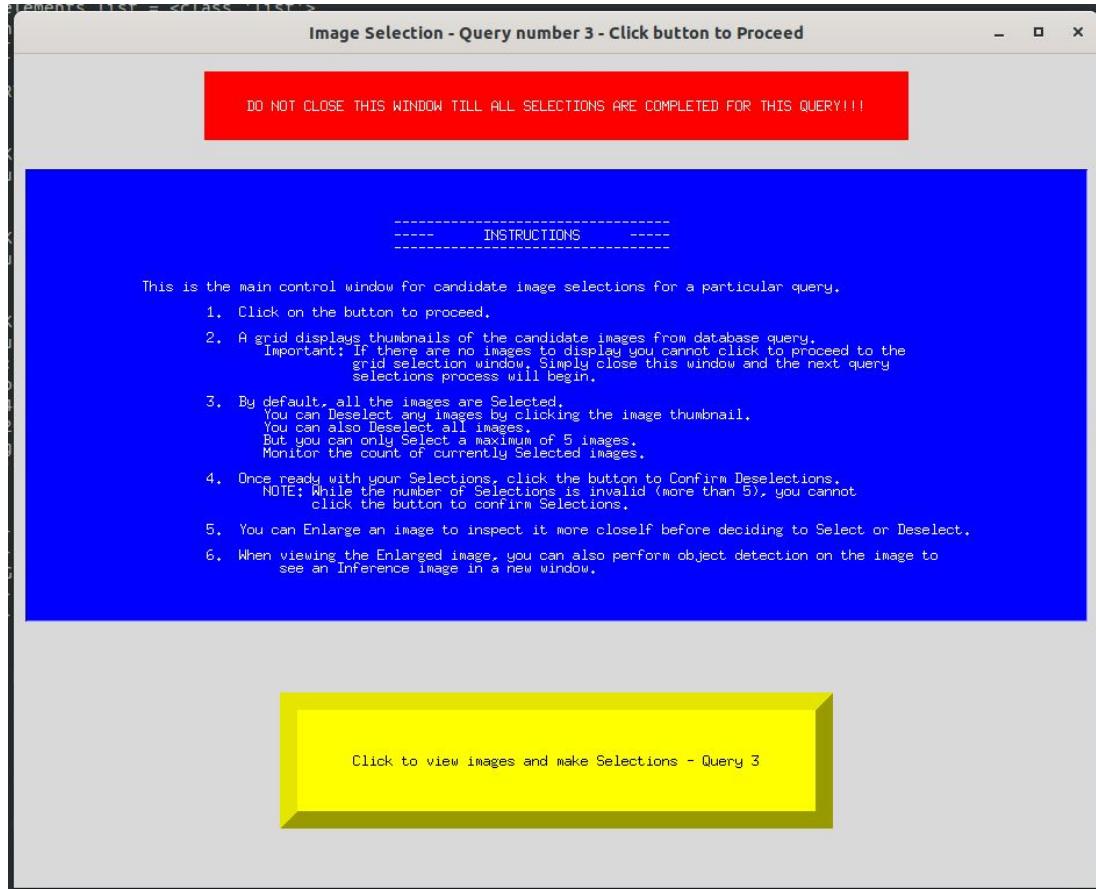
Image Selection - Query number 1 - No Images to Display -- Please close this window

DO NOT CLOSE THIS WINDOW TILL ALL SELECTIONS ARE COMPLETED FOR THIS QUERY!!!

No images to make Selections - Query 1 - Please close this window

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

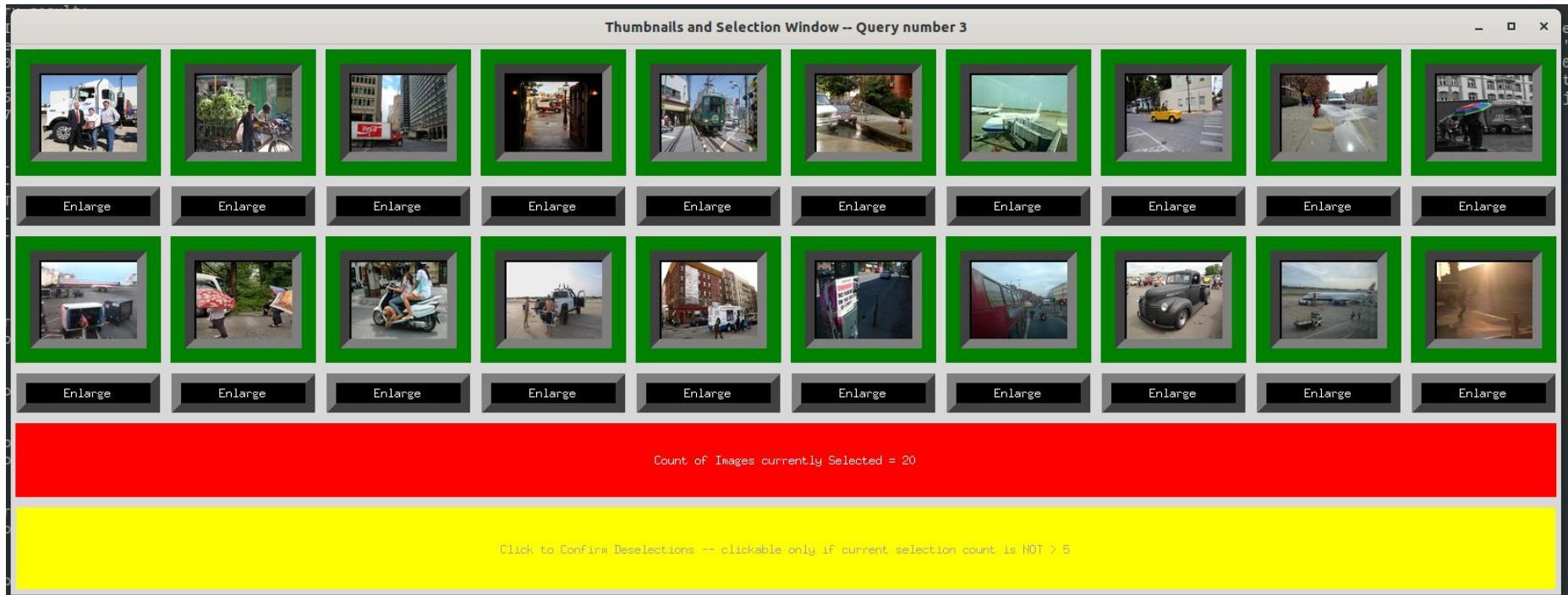


- Example of query with results:
  - > Many images found containing the object "Truck".
  - > Proceed to grid selection button is clickable

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

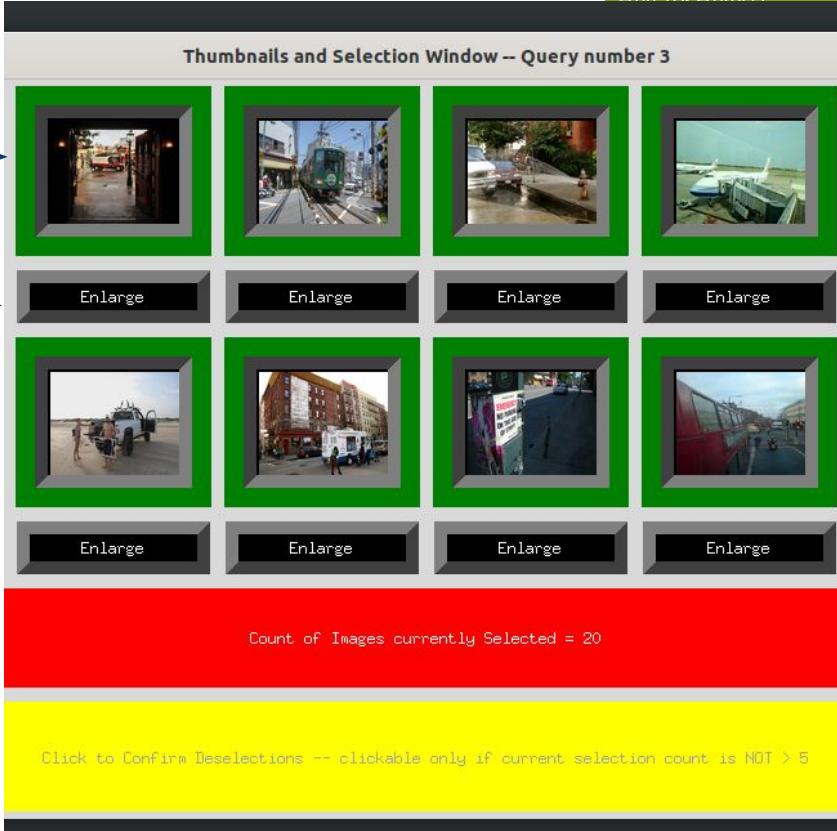
- Grid selection window displays up to 20 images.



# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE

- All images Selected by default - the border is Green



- User clicks Enlarge button
  - > Then new window displays enlarged image



- But count is 20 (greater than maximum limit of 5)  
=> User MUST Deselect images



- Confirmation button is disabled



# Find Candidate Images - GUI flow

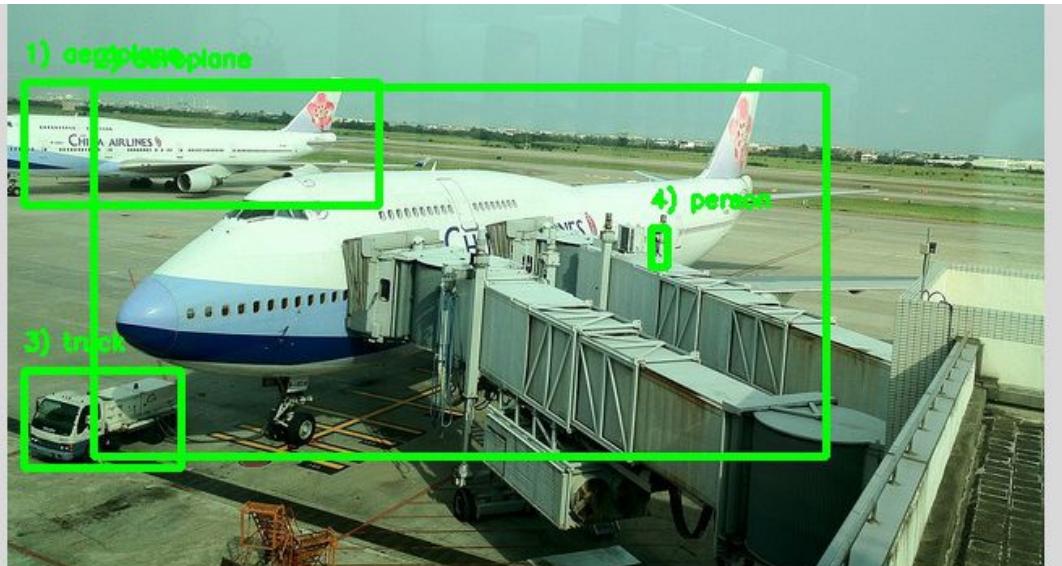
STAATLICH  
ANERKANNTE  
HOCHSCHULE



- User clicked Enlarge button to see full size image
- Image path displayed
- Option to click button to perform one-off inference with same detector used to populate database.

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

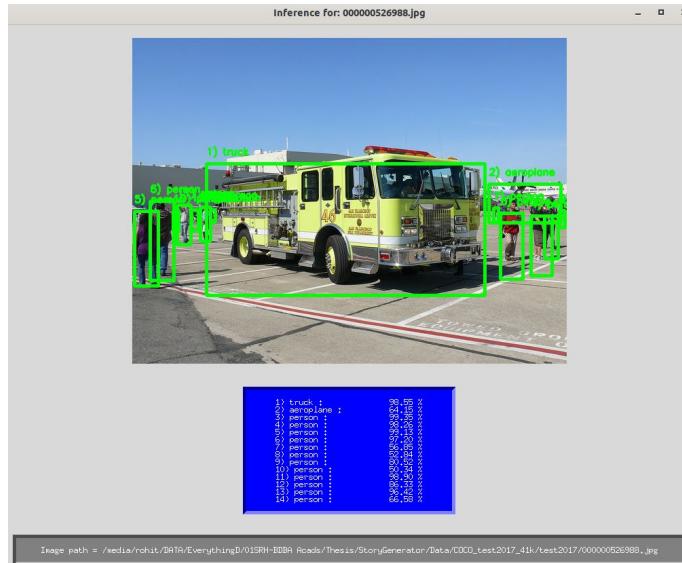


1) aeroplane :	98.07 %
2) aeroplane :	95.12 %
3) truck :	87.70 %
4) person :	56.92 %

- User clicked button to perform object detection inference and can study output image in new window
- Bounding boxes shown with serial number matching blue box information
- Earlier logic snapshot:
  - > Without min. HAS relationship score check. Thus could return image with no check of the all keyword object scores
  - > Note: Truck = 87% and Person = 56%

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE



Updated logic snapshot:

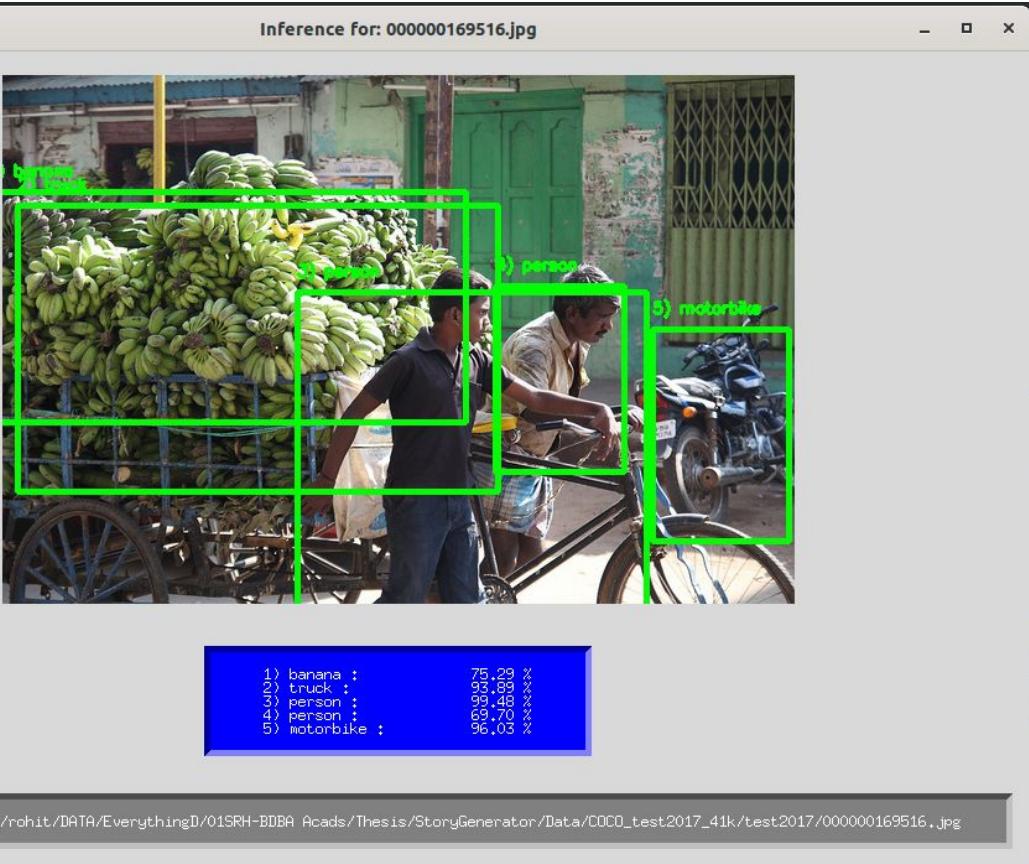
1) truck :	98.55 %
2) aeroplane :	64.15 %
3) person :	99.35 %
4) person :	98.26 %
5) person :	99.13 %
6) person :	97.20 %
7) person :	56.85 %
8) person :	52.84 %
9) person :	80.52 %
10) person :	50.34 %
11) person :	98.90 %
12) person :	86.33 %
13) person :	96.42 %
14) person :	66.58 %

## Updated logic snapshot:

Checks HAS relationship score > 0.90 for the objects used in query.

Note: At least one object of "Truck" and "Person" have scores > 90%

# Find Candidate Images - GUI flow

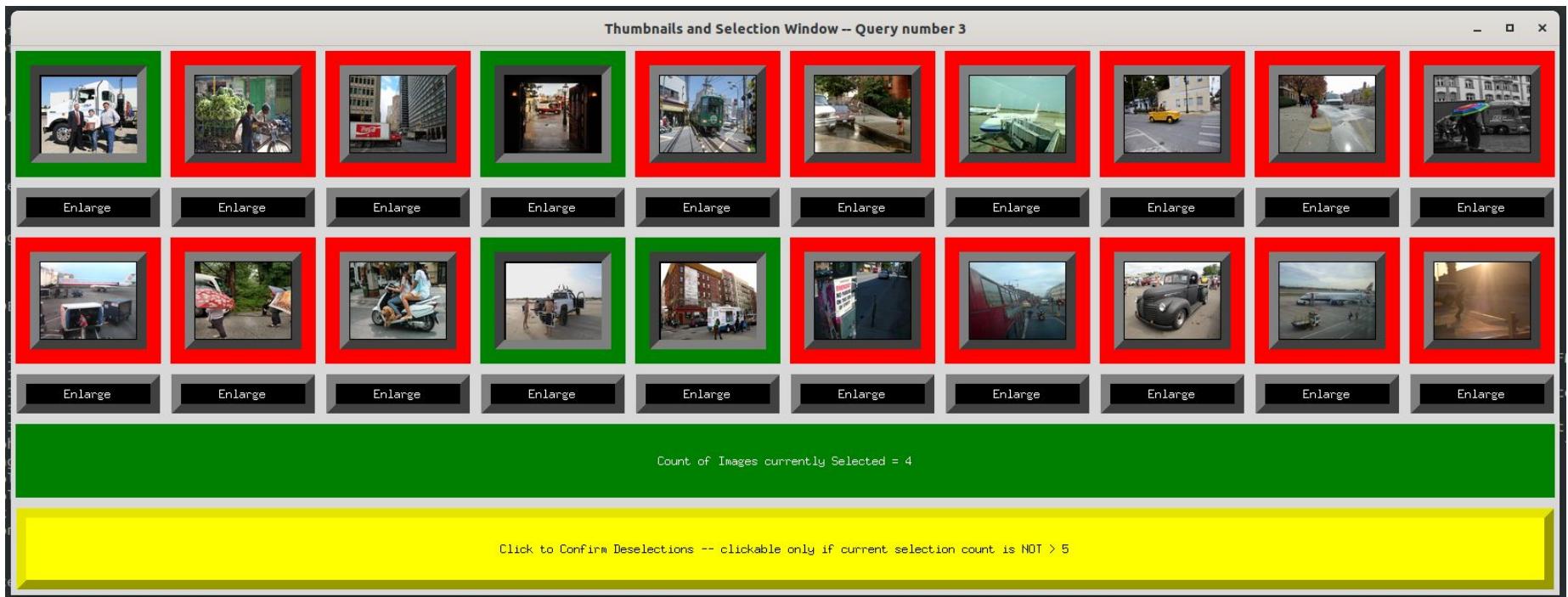


- **Example of completely wrong image**
  - > There is no “truck”.
  - > Model scores 93% confidence for “truck”.
- Despite setting minimum HAS score = 90% during Neo4j query, such cases will not be tackled.
- **GUI selection by user is crucial!**

# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Final Selections for 4 images
- Now the Confirm Deselections button is clickable



# Find Candidate Images - GUI flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Data structure before and after selection process:
  - > User deselected 16 of 20 images. So 4 retained and passed to Auto-captions Block

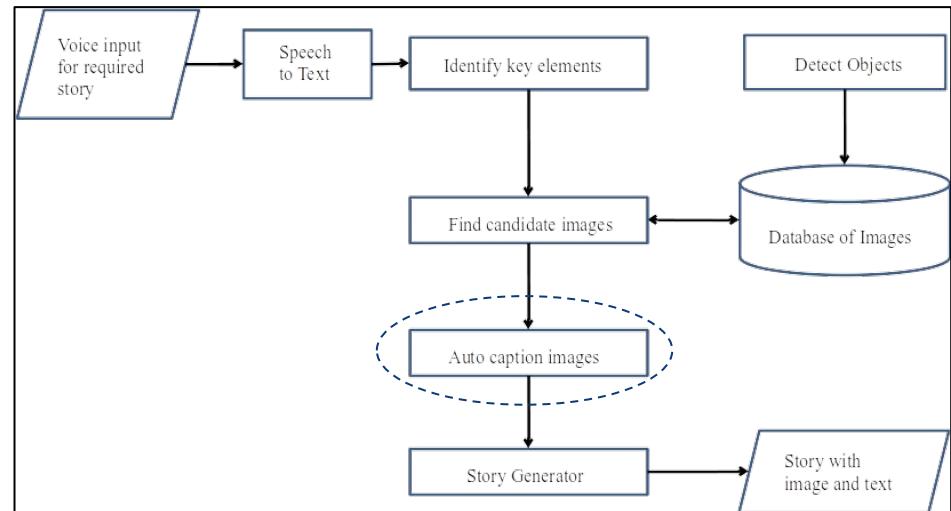
```
For Query 3
Number of candidate images before selection = 20
Number of Deselections done = 16
Number of images remaining after Deselections = 4

      ----- Query images info BEFORE::
[{'Image': '000000169542.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000169516.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000292186.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000146747.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000313777.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000449668.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000509771.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000012149.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000168815.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000168743.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000518174.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000017467.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000581864.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000225580.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000265504.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000361201.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000304424.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000225081.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000225051.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000499699.jpg', 'Source': 'coco_test_2017'}]

      ----- Positions removed::
[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19]

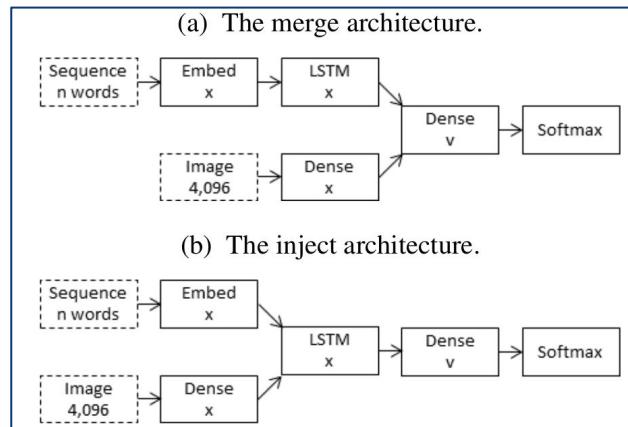
      ----- Query images info AFTER::
[{'Image': '000000169542.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000146747.jpg', 'Source': 'coco_test_2017'},
{'Image': '000000225580.jpg', 'Source': 'coco_test_2017'}, {'Image': '000000265504.jpg', 'Source': 'coco_test_2017'}]
```

# Auto caption Images Block



# Image Captioning - Introduction

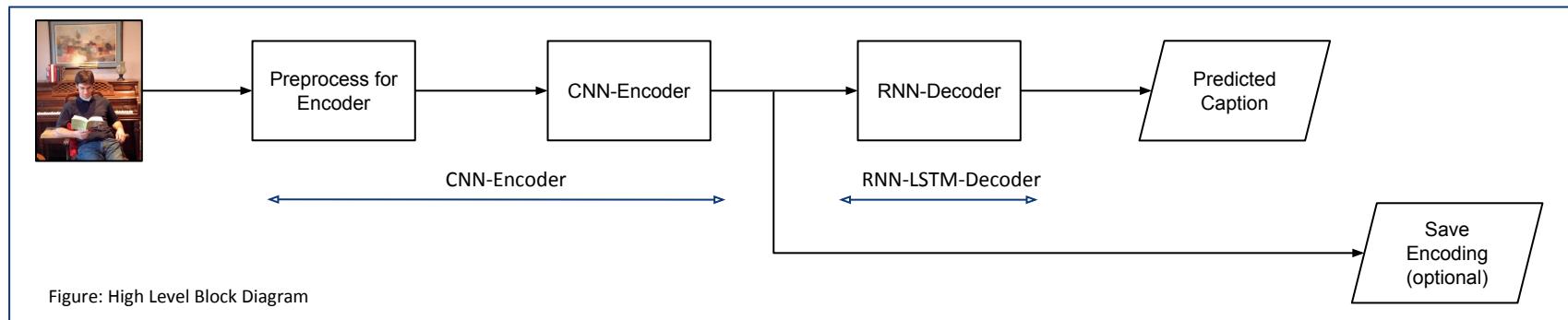
- **Goal:** Describe an image with a natural language sentence
- Three broad methods: Paper “A survey of deep neural network-based image captioning” : <https://doi.org/10.1007/s00371-018-1566-y>
  - > **Retrieval** aka Ranking based
    - > Not used as outdated method.
    - > Essentially find images in repository that are “close to” input image. Use the captions of the “close” images to create prediction. Heavily dependant on the breadth of repository
  - > **Template** based
    - > Two tasks carried out: Object detection to find bag of likely words. Then, create templates with the likely words and uses an LM to generate the prediction.
    - > Limitations: does not really understand overall image and only tries to combine objects detected in meaningful way based on preset templates.
  - > **End-to-End** Learning based:
    - > Use a Deep-CNN to create feature vector to represent image.
    - > Use this vector as during Decoding stage that are based on RNNs to output predicted caption
    - > Two methods to combine the description and image: *Inject vs Merge*
    - > “*Attention Mechanism*”: Focuses attention on specific part of image at different stages of prediction timesteps by learning “Context mapping”
- **Model used:** End-to-End type, Merge architecture, without Attention mechanism



Source: From paper “What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator” by Tanti et al. 2017 (<https://arxiv.org/abs/1708.02043>)

# Image Captioning - High level architecture

- CNN-Encoder: Google's Inception-v3 pre-trained on Imagenet data
- RNN-Decoder: Single layer 256 cell LSTM model - explicitly trained on use case data



- Figure: High Level Block Diagram
- Input image: Resized to 299 x 299 pixels due to encoder requirement
  - Preprocess for Encoder: Done via Keras built-in method - change scaling etc. per encoder requirements
  - Encoder: Outputs 2048 feature vector for each image aka "Image encoding". Tapped the last layer before final softmax output.
  - Decoder:
    - > *After training:* Process the Image encoding to output the Predicted caption
    - > *During training:* Process the image encoding + input description to learn correct predictions
  - Followed this link: <https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8>

# Image Captioning - Decoder Detailed architecture

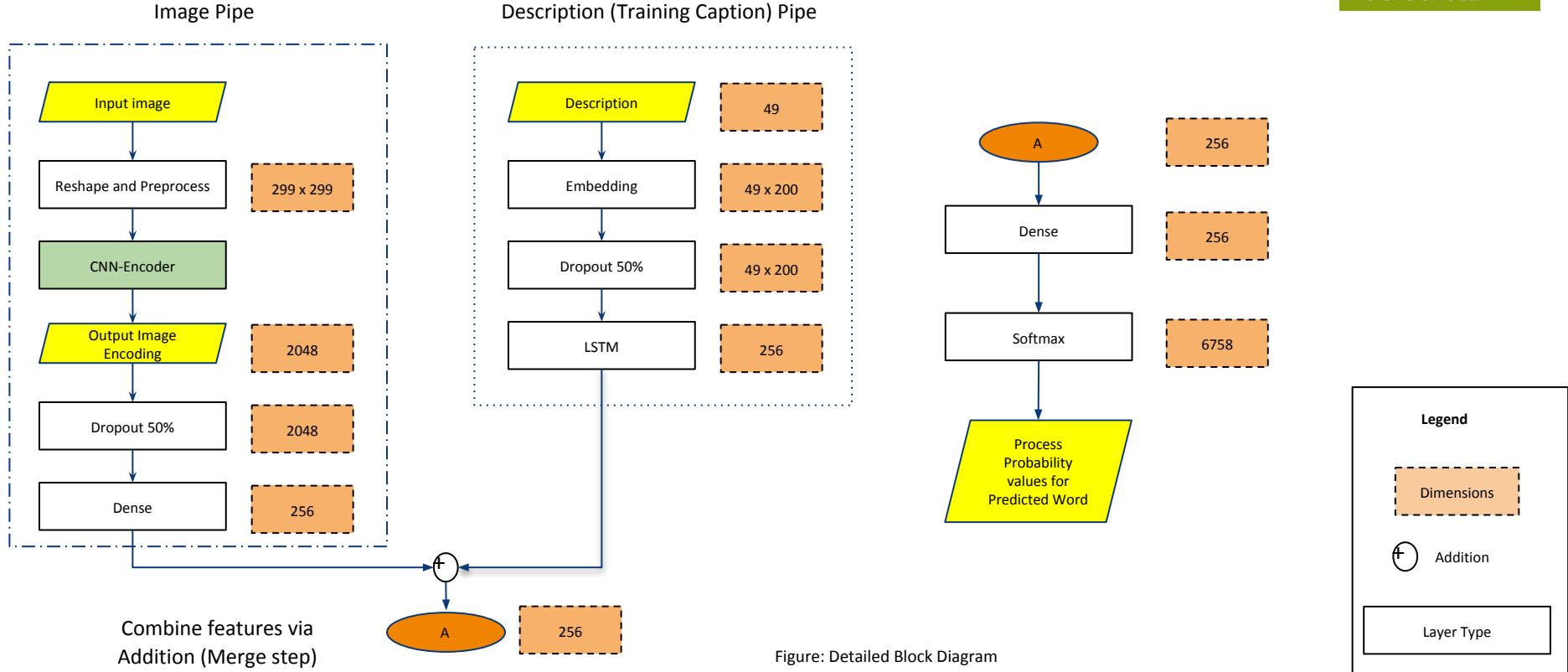


Figure: Detailed Block Diagram

# Image Captioning - Data for model training

STAATLICH

- Inputs = Image + Description:
  - > Image = vector of 2048 values from Encoder
  - > Description = cleaned ground truth sentence with insertion of special tokens for start and end
- During preprocessing:
  - > build the “Wordtoix” and “IxtoWord” to map between word tokens and unique integer representation
  - > Find the maximum length of ground truth description and decide the value for MAX\_LENGTH\_CAPTION
- Each image feature and its description gets mapped to series of actual inputs as the input vector (ImgFeat + Xt) and the expected output Yt
  - > 0 padding at end to make all data points of equal length = MAX\_LENGTH\_CAPTION
  - > Yt is the unique integer representation of the predicted word

Image 1	Description 1:		“A dog is barking outside.”	
	Cleaned + special tokens:		“startseq dog is barking outside endseq”	
	Length = 6			
Image 2	Description 2:		“That is a nice sweater.”	
	Cleaned + special tokens:		“startseq that is nice sweater endseq”	
	Length = 6			
Max Length of Caption: Parameter value chosen = 8				
SN	LogicalData	ImgFeat	Xt	Yt
1	1	imgF1	startseq	dog
2	1	imgF1	startseq dog	is
3	1	imgF1	startseq dog is	barking
4	1	imgF1	startseq dog is barking	outside
5	1	imgF1	startseq dog is barking outside	endseq
6	2	imgF2	startseq	that
7	2	imgF2	startseq that	is
8	2	imgF2	startseq that is	nice
9	2	imgF2	startseq that is nice	sweater
10	2	imgF2	startseq that is nice sweater	endseq
Above word inputs get mapped using the wordtoix data structure and 0 padding				
1	1	imgF1	1 0 0 0 0 0 0 0	11
2	1	imgF1	1 1 1 0 0 0 0 0 0	12
3	1	imgF1	1 1 1 1 2 0 0 0 0 0	13
4	1	imgF1	1 1 1 1 1 2 1 3 0 0 0 0	14
5	1	imgF1	1 1 1 1 1 2 1 3 1 4 0 0 0	9
6	2	imgF2	1 0 0 0 0 0 0 0	21
7	2	imgF2	1 2 1 0 0 0 0 0 0	12
8	2	imgF2	1 2 1 1 2 0 0 0 0 0	22
9	2	imgF2	1 2 1 1 1 2 2 0 0 0 0	23
10	2	imgF2	1 2 1 1 1 2 2 2 0 0 0 0	9

Figure: Data prep for decoder training

# Image Captioning - Preprocessing

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Load descriptions from the “annotations json file” and cleanup sentences:
  - > Lowercase
  - > Remove punctuations and special characters (including the period at end of descriptions)
  - > Remove words with length = 1
    - > Prevent orphan letters causing a mess. E.g. punctuation removal from “there’s a great” -> “there s a great” -> “there great”
    - > unfortunately legitimate “a” will also be dropped
  - > Drop non-alphabet words. E.g. gr8
- Insert the special tokens for start and end of descriptions:
  - > “startseq” and “endseq” - required only for the training data
- Use this data to calculate:
  - > Create the full Vocabulary i.e. find all the unique words from the descriptions
  - > Based on chosen threshold for “high frequency words”, cull the full Vocabulary to the one to actually use for model
  - > Using the Culled Vocabulary, calculate the Maximum Length of Caption
- Create the data structures for “wordtoix” and “ixtoword” - mapping string tokens to unique integer representations
- Create embeddings matrix based on culled vocabulary:
  - > using GloVe-200 consisting of 400,000 words

# Image Captioning - Preprocessing

STAATLICH

Clean up  
descriptions for a  
random image

Note the change  
before and after

```
In [24]: ## example of caption with accidental newline \n in the caption
descriptions_test['000000482917']
```

```
Out[24]: ['A dog sitting between its masters feet on a footstool watching tv\n',
'A dog between the feet of a person looking at a TV.',
'A dog and a person are watching television together.',
'A person is sitting with their dog watching tv.',
'A man relaxing at home, watching television with his dog.]
```

```
In [25]: # prepare translation table for removing punctuation
## string.punctuation gives '!#$%&|()>*,-.:;=>?@[]{}^`{|}~` and will take care of all these characters being made in
to a space
tran_table = str.maketrans(string.punctuation, ' ' * len(string.punctuation))
for key, desc_list in descriptions.items():
    for idx in range(len(desc_list)):
        desc = desc_list[idx]
        # replace all punctuation with space in description before tokenizing
        desc = desc.translate(tran_table)
        # tokenize
        desc = desc.split()
        # convert to lower case
        desc = [word.lower() for word in desc]
        # remove hanging 's' and 'a'
        desc = [word for word in desc if len(word)>1]
        # remove any non-alphabetic tokens
        desc = [word for word in desc if word.isalpha()]
        # overwrite with cleaned description
        desc_list[idx] = ' '.join(desc)
```

```
In [26]: ## example of caption with accidental newline \n in the caption -- POST CLEANUP
descriptions_test['000000482917']
```

```
Out[26]: ['dog sitting between its masters feet on footstool watching tv',
'dog between the feet of person looking at tv',
'dog and person are watching television together',
'person is sitting with their dog watching tv',
'man relaxing at home watching television with his dog']
```

# Image Captioning - Model Parameters

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- RNN-Decoder model defined based on the chosen training images (97k images) and their cleaned descriptions

Model Parameters: Common for both models 2 and 3		
Parameter	Value	Description
VOCAB_SIZE	6758	Number of words model can output - includes special tokens ("startseq", "endseq", 0)
EMBEDDING_DIMS	200	Size of the vector to represent each word as per chosen embeddings approach
EMBEDDING_MATRIX_SHAPE	(6758, 200)	Shape of the embedding matrix = (VOCAB_SIZE, EMBEDDING_DIMS)
MAX_LENGTH_CAPTION	49	Maximum length of ground truth description with "startseq" and "endseq"
UNIQUE WORDS IN VOCAB	24323	Total unique words in all the descriptions
UNIQUE WORDS (HIGH FREQ.)	6757	Total unique words in descriptions occurring > High frequency threshold
HIGH FREQUENCY THRESHOLD	10	Threshold value used to cull all unique words in vocab

# Image Captioning - Model Parameters

## Code Snippets

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Sizes of the descriptions and the Encodings for Train and Val data = 97k and 3k respectively

```
[15]: print(f"Encodings data:\nlen(img_encodings_train) = {len(img_encodings_train)}\nt\tnlen(img_encodings_val) = {len(img_encodings_val)}")  
print(f"Descriptions data:\nlen(descriptions_train) = {len(descriptions_train)}\nt\tnlen(descriptions_val) = {len(descriptions_val)}")  
print(f"\nCHECK : reloaded values = 97k for Train , 3k for Validation")
```

```
Encodings data:  
len(img_encodings_train) = 97000           len(img_encodings_val) = 3000  
Descriptions data:  
len(descriptions_train) = 97000           len(descriptions_val) = 3000  
  
CHECK : reloaded values = 97k for Train , 3k for Validation
```

Calculate the Total Unique words in vocabulary based on descriptions = 24323

```
[16]: ## at this stage the descriptions_train already has the start and end tokens added to it  
vocabulary = set()  
for key in descriptions_train.keys():  
    [vocabulary.update(d.split()) for d in descriptions_train[key]]  
print(f"Original Vocabulary Size with all words = {len(vocabulary)}")  
print(f"\nCHECK : reloaded value = 24323")
```

```
Original Vocabulary Size with all words = 24323  
  
CHECK : reloaded value = 24323
```

# Image Captioning - Model Parameters

## Code Snippets

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Calculate the High Frequency words = 6757

[17]:

```
# Create a list of all the training captions, find the freq and retain words where the freq > threshold chosen

all_desc_in_training_samples = []
for key, val in descriptions_train.items():
    for cap in val:
        all_desc_in_training_samples.append(cap)

MIN_WORD_COUNT_THRESHOLD = 10
word_counts = {}
nsents = 0
for each_desc in all_desc_in_training_samples:
    nsents += 1
    for w in each_desc.split(' '):
        word_counts[w] = word_counts.get(w, 0) + 1

vocab_threshold = [w for w in word_counts if word_counts[w] >= MIN_WORD_COUNT_THRESHOLD]

print(f"Culled vocabulary to only retain words occurring more than threshold = {MIN_WORD_COUNT_THRESHOLD} times.\nNew vocab size , len(vocab_threshold) = {len(vocab_threshold)}")
print(f"\nCHECK : reloaded value = 6757")
```

```
Culled vocabulary to only retain words occurring more than threshold = 10 times.
New vocab size , len(vocab_threshold) = 6757

CHECK : reloaded value = 6757
```

# Image Captioning - Model Parameters

## Code Snippets

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Calculate Max. caption length = 49

```
[18]: ## determine the maximum sequence length - parameter MAX_LENGTH_CAPTION used during the RNN decoder model setup

## convert a dictionary of clean descriptions to a list of descriptions
def extract_each_desc(_descriptions):
    all_desc = list()
    for key in _descriptions.keys():
        [all_desc.append(d) for d in _descriptions[key]]
    return all_desc

## find the longest description length
def find_max_length_desc(_descriptions):
    desc_sentences = extract_each_desc(_descriptions)
    return max(len(d.split()) for d in desc_sentences)

MAX_LENGTH_CAPTION = find_max_length_desc(descriptions_train) ## will be used directly later while defining Decoder model
print(f"Max Description Length: {MAX_LENGTH_CAPTION}")
print(f"\nCHECK : reloaded value = 49")
```

```
Max Description Length: 49
```

```
CHECK : reloaded value = 49
```

# Image Captioning - Model Parameters

## Code Snippets

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Set VOCAB\_SIZE using the “wordtoix” or “ixtoword”  
data length + 1

Additional 1 for the 0 pad token

```
[19]: ## the value now, as it will be used as:: VOCAB_SIZE = len(wordtoix) + 1
print(f"\nlen(wordtoix) = {len(wordtoix)}")
print(f"\nCHECK : reloaded value = 6757")

VOCAB_SIZE = len(wordtoix) + 1
print(f"\n\nSet the VOCAB_SIZE = len(wordtoix) + 1 = {VOCAB_SIZE}")

EMBEDDING_DIMS = 200
print(f"\n\nSet the EMBEDDING_DIMS = {EMBEDDING_DIMS}")
```

```
len(wordtoix) = 6757

CHECK : reloaded value = 6757

Set the VOCAB_SIZE = len(wordtoix) + 1 = 6758

Set the EMBEDDING_DIMS = 200
```

See the indices in “wordtoix” for special tokens and  
some random word

```
[20]: ## see the index output by wordtoix for the start and end sequence tokens as well as some random one word
print( wordtoix.get('startseq') , wordtoix.get('endseq') , wordtoix.get('cat') )
```

```
1 9 526
```

# Image Captioning - Datasets Used

- MS COCO 2017 has images with 5 descriptions per image in the Annotations file of the dataset.

## Original dataset information

Original MS COCO 2017 Dataset		
Dataset	No. of Images	Descriptions?
Train	118k	Yes
Val	5k	Yes
Test	41k	No

## Usage in the thesis work:

- MS\_Coco2017\_Train : Used 100k as my Train and Val datasets
- MS\_Coco2017\_Val : Used all 5k as my Test dataset

Data splits for thesis-work			
Original Source	Dataset	No. of Images	Descriptions?
Coco2017_Train	Train	97k	Yes
Coco2017_Train	Validation	3k	Yes
Coco2017_Val	Test	5k	Yes

# Image Captioning - Model Comparison

STAATLICH  
ANERKANNTE  
HOCHSCHULE

## Training Hyperparameters

Model 2			
#Ep	Ep (From-To)	LR	BS
2	1-2	0.0005	128
5	3-7	0.0002	128
3	8-10	0.0001	64

Model 3			
#Ep	Ep (From-To)	LR	BS
13	1-13	0.001	64
2	14-15	0.001	128
3	16-18	0.0005	32

- Models trained with different hyperparameters
- Common Parameters:
  - > Optimizer = Adam
  - > Loss functions = Categorical cross-entropy

Legend	
Ep	Epoch
#Ep	Number of Epochs
LR	Learning Rate
BS	Batch Size

# Image Captioning - Model Comparison

Training and Validation Set Losses for Model 2 v/s Model 3 : **Chosen Model 3 to proceed**

Train dataset = 97k images, Validation dataset = 3k images

Ep	Model 2 Losses		Model 3 Losses	
	Train	Val	Train	Val
2	3.4819	3.52842	3.1985	3.27492
4	3.2286	3.36663	3.0032	3.17518
6	3.1432	3.29084	2.9315	3.14292
8	3.0858	3.24664	2.8673	3.12330
10	<b>3.0448</b>	<b>3.21695</b>	<b>2.8443</b>	<b>3.11437</b>
12	-	-	2.8397	3.11331
14	-	-	2.8013	<b>3.08822</b>
16	-	-	2.7977	3.09369
18	-	-	<b>2.7704</b>	<b>3.08865</b>

What we see:

Train and Val losses: Consistent fall with epochs - no overfitting

After 10 epochs, Model 3 better than Model 2.

About Batch sizes:

Training Losses: BS varies

Val Losses: BS = 64 always

Inconsistent Value: Wrong weights file

Common Parameters:

Optimizer = Adam

Loss function = Categorical cross-entropy

Training Losses: BS varies

Val Losses: BS = 64

Legend	
Ep	Epoch
LR	Learning Rate
BS	Batch Size

# Image Captioning - Model Evaluation

- Bleu scorer of NLTK used:
  - > import nltk.translate.bleu\_score as nltk\_bleu
  - > Bleu score = {nltk\_bleu.sentence\_bleu( [list of GT descriptions] , "the predicted caption from model" )}
- Some random lowest and highest scores from both models below

Model 2 - After 10 epochs

In [12]:	dfbs.head()			
Out[12]:		img	infcap	bsnltk
2294	000000484760	clock tower with clock on it		8.580523e-155
2299	000000485130	bed with two beds and two beds		3.831503e-78
819	000000197870	bird perched on the ground next to bird		4.351978e-78
1959	000000015517	train traveling down bridge next to bridge		4.815777e-78
2461	000000439522	man in black jacket and black jacket and black...		5.385075e-02

In [13]:	dfbs.tail()			
Out[13]:		img	infcap	bsnltk
1702	000000149770	man riding surfboard on top of wave		1.0
3635	000000199442	man riding wave on top of surfboard		1.0
1663	000000450303	group of people sitting around table with laptops		1.0
611	000000383606	bathroom with sink and mirror		1.0
3581	000000320696	man riding wave on top of surfboard		1.0

Model 3 - After 10 epochs

In [14]:	dfbs.head()			
Out[14]:		img	infcap	bsnltk
493	000000484760	clock tower with clock on it		8.580523e-155
2270	000000112626	room with two beds and chair		9.746048e-155
1762	000000507667	an old model model model fighter jet		2.845685e-78
4200	000000468233	an old model model model cell phone		3.516915e-78
2084	000000453860	an open suitcase with an open door open		3.775819e-78

In [15]:	dfbs.tail()			
Out[15]:		img	infcap	bsnltk
1412	000000373705	red fire hydrant sitting on the side of road		1.0
3401	000000325347	man holding tennis racquet on tennis court		1.0
3410	000000223959	man holding tennis racquet on tennis court		1.0
689	000000466835	bunch of bananas hanging from tree		1.0
4813	000000462031	baseball player holding bat on top of field		1.0

# Image Captioning - Model Comparison

- Bleu Scores after 10 epochs:
  - > only Greedy Search used
  - > Test dataset = 5k images
- Model 3 Scores **slightly higher**: Average by 0.12% and Median by 0.06%
- Standard deviation and relative frequencies almost same
- Overall Model 3 is slightly better than Model 2 : [continued training Model 3 for more epochs, abandoned Model 2](#)

Bleu Scores after 10 epochs: Model 2 v/s 3		
Bleu Scores	Model 2	Model 3
Max	1.0	1.0
Min	0.0	0.0
Median	0.6372	0.6376
Average	0.6327	0.6335
Std. Dev.	0.1643	0.1648

Bleu Scores by Bins – frequency comparison – Total data points = 5k data points						
Greedy Search used overall						
	Model 2			Model 3		
Score Bin	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq
0.0 – 0.1	9	0.00	0.00	6	0.00	0.00
0.1 – 0.2	26	0.01	0.01	24	0.00	0.01
0.2 – 0.3	82	0.01	0.02	90	0.02	0.02
0.3 – 0.4	294	0.06	0.08	271	0.05	0.08
0.4 – 0.5	664	0.13	0.21	676	0.14	0.21
0.5 – 0.6	977	0.20	0.41	978	0.20	0.41
0.6 – 0.7	1175	0.24	0.65	1190	0.24	0.65
0.7 – 0.8	965	0.19	0.84	948	0.19	0.84
0.8 – 0.9	565	0.12	0.95	572	0.12	0.95
0.9 – 1.0	217	0.05	1.00	203	0.04	1.00
Total	4974	-	-	4958	-	-

# Image Captioning - Model Comparison

- Bleu Scores for Model 3 after after 10 vs 18 epochs:
  - > only Greedy Search used
  - > Test dataset = 5k images
- 18 Epoch Scores **higher**: Average by 1.44% and Median by 1.87%
- Standard deviations and relative frequencies similar - but improvements in 0.6 to 0.8 range (see Count)
- Overall Model 3 after 18 epochs is better : **Model 3 with 18 epochs used**

Bleu Scores Model 3: After 10 v/s 18 epochs		
Bleu Scores	10 Ep	18 Ep
Max	1.0	1.0
Min	0.0	0.0
Median	0.6376	0.6495
Average	0.6335	0.6426
Std. Dev.	0.1648	0.1629

Bleu Scores by Bins – frequency comparison – Total data points = 5k data points						
Greedy Search used overall						
	Model 3 – 10 Epochs			Model 3 – 18 Epochs		
Score Bin	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq
0.0 – 0.1	6	0.00	0.00	2	0.00	0.00
0.1 – 0.2	24	0.00	0.01	12	0.00	0.00
0.2 – 0.3	90	0.02	0.02	81	0.01	0.02
0.3 – 0.4	271	0.05	0.08	169	0.05	0.07
0.4 – 0.5	676	0.14	0.21	609	0.12	0.19
0.5 – 0.6	978	0.20	0.41	988	0.20	0.39
0.6 – 0.7	1190	0.24	0.65	959	0.23	0.62
0.7 – 0.8	948	0.19	0.84	1028	0.20	0.83
0.8 – 0.9	572	0.12	0.95	570	0.12	0.94
0.9 – 1.0	203	0.04	1.00	230	0.05	1.00
Total	4958	-	-	4648	-	-

# Image Captioning - Choosing Inference Method

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Evaluated Greedy Search v/s Beam Search
  - > Tried with Beam Widths = 3 and 5
- Comparing Model 3 after 18 epochs with Bleu Scores
  - > Test dataset = 5k images
- Scores for Greedy consistently better.
- Scores worst for Beam search with Width = 5
  - > Also inference time substantially longer
- Will go with Greedy or Beam (3)

Bleu Scores Model 3: Greedy v/s Beam search			
Bleu Scores	Greedy	Beam (3)	Beam (5)
Max	1.0	1.0	1.0
Min	0.0	0.0	0.0
Median	0.6495	0.6431	0.6270
Average	0.6426	0.6385	0.6235
Std. Dev.	0.1629	0.1600	0.1617

# Image Captioning - Choosing Inference Method

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Bleu scores for Model 3 after 18 epochs:
  - > Test dataset = 5k images
- NLTK able to score 300 more inferences with Beam search:
  - > Width=3 results: mostly with 0.6-0.7 scores
- Overall similar results
- Implemented Beam search with Width = 3 in thesis work

Bleu Scores by Bins – frequency comparison – Total data points = 5k data points

Model 3 – 18 Epochs – Comparison Greedy / Beam Search

Score Bin	Greedy			Beam Search (Width = 3)			Beam Search (Width = 5)		
	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq	Count	Rel. Freq	Cum. Freq
0.0 – 0.1	2	0.00	0.00	4	0.00	0.00	6	0.00	0.00
0.1 – 0.2	12	0.00	0.00	9	0.00	0.00	14	0.00	0.00
0.2 – 0.3	81	0.01	0.02	91	0.02	0.02	112	0.03	0.03
0.3 – 0.4	169	0.05	0.07	272	0.05	0.07	323	0.06	0.09
0.4 – 0.5	609	0.12	0.19	590	0.12	0.19	661	0.13	0.22
0.5 – 0.6	988	0.20	0.39	999	0.20	0.39	1031	0.21	0.43
0.6 – 0.7	959	0.23	0.62	1235	0.25	0.64	1201	0.24	0.67
0.7 – 0.8	1028	0.20	0.83	1008	0.21	0.85	964	0.19	0.86
0.8 – 0.9	570	0.12	0.94	565	0.11	0.96	495	0.10	0.96
0.9 – 1.0	230	0.05	1.00	189	0.04	1.00	159	0.04	1.00
Total	4648	-	-	4962	-	-	4966	-	-

# Image Captioning - Greedy v/s Beam (3) outputs

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Inferring image 1 of 100:  
/media/rohit/DATA/EverythingD/01SRH-BDBA\_Acads/Thesis/StoryGenerator/D

Cleaned descriptions:

Orig Desc 1 :: woman stands in the dining area at the table  
Orig Desc 2 :: room with chairs table and woman in it  
Orig Desc 3 :: woman standing in kitchen by window  
Orig Desc 4 :: person standing at table in room  
Orig Desc 5 :: living area with television and table



Greedy caption :: living room with couch and television  
Beam3 caption :: living room filled with furniture and flat screen tv

Inferring image 36 of 100:  
/media/rohit/DATA/EverythingD/01SRH-BDBA\_Acads/Thesis/StoryGenerator/D

Cleaned descriptions:

Orig Desc 1 :: bread and fruit are on table with knife and fork  
Orig Desc 2 :: some bread ham fruit jam and glass of orange juice  
Orig Desc 3 :: an assortment of meats and cheeses with bread  
Orig Desc 4 :: variety of different types of food on table  
Orig Desc 5 :: selection of breads meat spreads and cheeses



Greedy caption :: plate of food with some fruit and vegetables  
Beam3 caption :: table topped with plates of food and drink

Inferring image 2 of 100:  
/media/rohit/DATA/EverythingD/01SRH-BDBA\_Acads/Thesis/StoryGenerator/Data/C

Cleaned descriptions:

Orig Desc 1 :: big burly grizzly bear is show with grass in the background  
Orig Desc 2 :: the large brown bear has black nose  
Orig Desc 3 :: closeup of brown bear sitting in grassy area  
Orig Desc 4 :: large bear that is sitting on grass  
Orig Desc 5 :: close up picture of brown bear face



Greedy caption :: brown bear walking across lush green field  
Beam3 caption :: brown bear standing on top of grass covered field

Inferring image 4 of 100:  
/media/rohit/DATA/EverythingD/01SRH-BDBA\_Acads/Thesis/StoryGenerator/Data/COCO

Cleaned descriptions:

Orig Desc 1 :: stop sign is mounted upside down on it post  
Orig Desc 2 :: stop sign that is hanging upside down  
Orig Desc 3 :: an upside down stop sign by the road  
Orig Desc 4 :: stop sign put upside down on metal pole  
Orig Desc 5 :: stop sign installed upside down on street corner



Greedy caption :: stop sign is shown with graffiti on it  
Beam3 caption :: red stop sign sitting on the side of the road

From random outputs for 100 images of Test dataset:

[https://github.com/rbewoor/thesis/blob/master/ImgCap/BleuscoreAccuracy/demo\\_ImgCap\\_Reload\\_Lappy\\_Infer\\_Greedy\\_Beam3\\_1.ipynb](https://github.com/rbewoor/thesis/blob/master/ImgCap/BleuscoreAccuracy/demo_ImgCap_Reload_Lappy_Infer_Greedy_Beam3_1.ipynb)

# Image Captioning - Greedy v/s Beam (3) outputs

STAATLICH  
ANERKANNTE  
HOCHSCHULE

Inferring image 37 of 100:  
[/media/rohit/DATA/EverythingD/01SRH-BDBA\\_Acads/Thesis/StoryGenerator/Data](/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data)

Cleaned descriptions:

Orig Desc 1 :: clock is situated atop colorful tower  
Orig Desc 2 :: tall green clock tower sitting in the middle of sidewalk  
Orig Desc 3 :: colourful building has clock at the top  
Orig Desc 4 :: tall green and yellow building with clock at the top  
Orig Desc 5 :: large green building with clock at the top of it



Greedy caption :: red and white clock tower with clock on it  
Beam3 caption :: red fire hydrant on the side of the road

Inferring image 57 of 100:  
[/media/rohit/DATA/EverythingD/01SRH-BDBA\\_Acads/Thesis/StoryGenerator/Data/COCO\\_val2017](/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_val2017)

Cleaned descriptions:

Orig Desc 1 :: grey colored jet plane flying over snowy mountain range  
Orig Desc 2 :: silver jet is flying high in the sky above the clouds  
Orig Desc 3 :: jet fighter flying through the sky above snow covered mountain  
Orig Desc 4 :: an airplane with the swiss flag symbol is flying through the mountains  
Orig Desc 5 :: an airplane flying solo above blue terrain



Greedy caption :: fighter jet flying through the air with smoke coming out of it  
Beam3 caption :: fighter jet is flying through the sky

Inferring image 6 of 100:  
[/media/rohit/DATA/EverythingD/01SRH-BDBA\\_Acads/Thesis/StoryGenerator/Data](/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data)

Cleaned descriptions:

Orig Desc 1 :: woman posing for the camera standing on skis  
Orig Desc 2 :: woman standing on skis while posing for the camera  
Orig Desc 3 :: woman in red jacket skiing down slope  
Orig Desc 4 :: young woman is skiing down the mountain slope  
Orig Desc 5 :: person on skis makes her way through the snow



Greedy caption :: man in red jacket skiing down hill  
Beam3 caption :: man riding skis on top of snow covered slope

Inferring image 24 of 100:  
[/media/rohit/DATA/EverythingD/01SRH-BDBA\\_Acads/Thesis/StoryGenerator/Data/COCO\\_val2017\\_5k](/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_val2017_5k)

Cleaned descriptions:

Orig Desc 1 :: group of three chefs preparing food in kitchen  
Orig Desc 2 :: man making pizza in kitchen  
Orig Desc 3 :: some people in kitchen preparing food on counter  
Orig Desc 4 :: couple of guys cooking in restaurant kitchen that is open to the restaurant  
Orig Desc 5 :: cooks are gathered preparing meal in the kitchen



Greedy caption :: man is cutting cake with knife  
Beam3 caption :: man and woman preparing food in kitchen

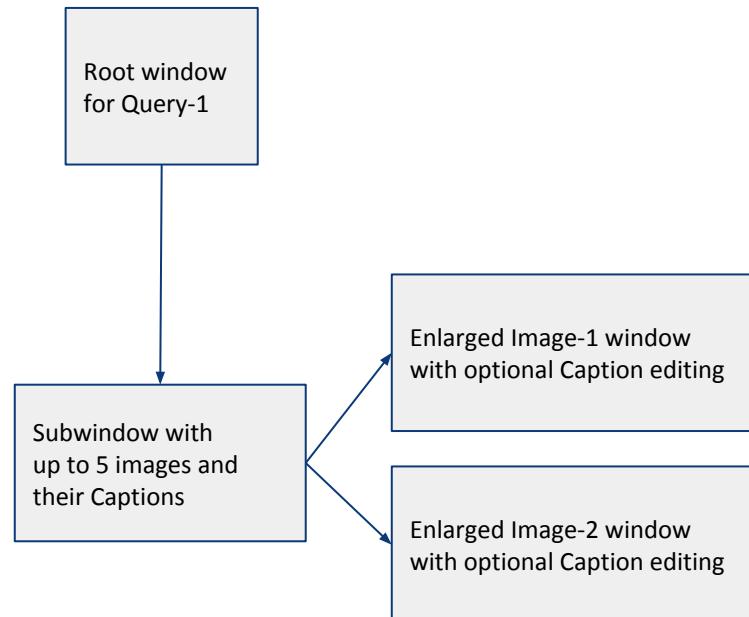
From random outputs for 100 images of Test dataset:

[https://github.com/rbewoor/thesis/blob/master/ImgCap/BleuscoreAccuracy/demo\\_ImgCap\\_Reload\\_Lappy\\_Infer\\_Greedy\\_Beam3\\_1.ipynb](https://github.com/rbewoor/thesis/blob/master/ImgCap/BleuscoreAccuracy/demo_ImgCap_Reload_Lappy_Infer_Greedy_Beam3_1.ipynb)

# Image Captioning - GUI Flow

- **Goal:** Use the data from the Image Caption inference stage; present the images with their captions to the user for inspection.
  - > Allow user to deselect images
    - > these images and their caption will NOT be sent to the Story Generator block.
  - > Allow editing of the captions for corrections before passing data to Story Generation block

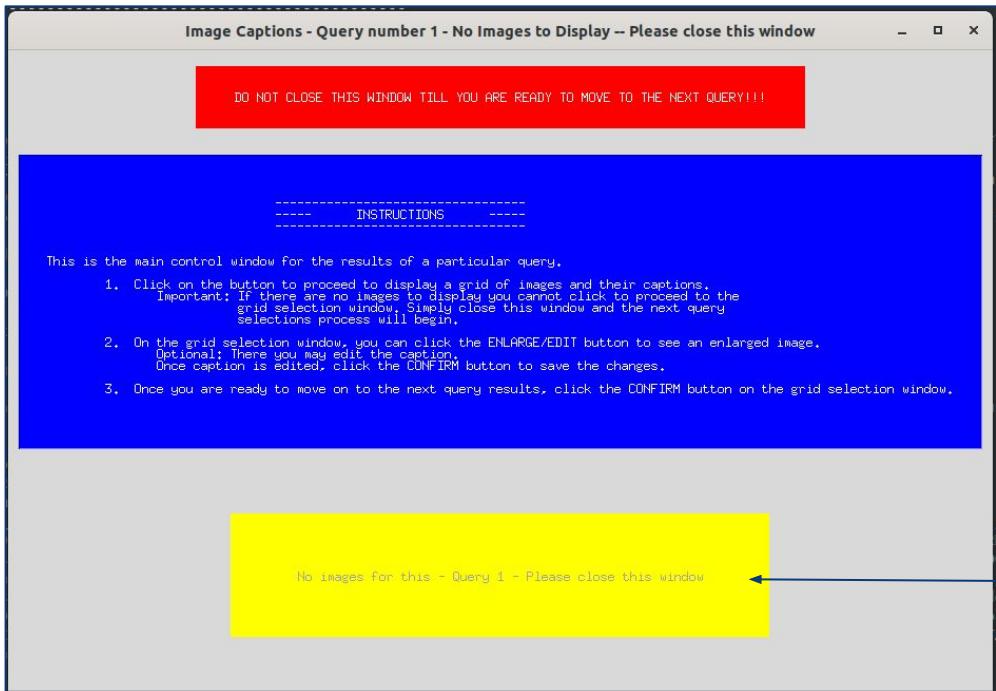
- Repeats for each of the three queries - once per input sentence
  - > If no images, does not allow grid window selection logic



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Main window - no images for this query
  - > User must simply close window and proceed to next query



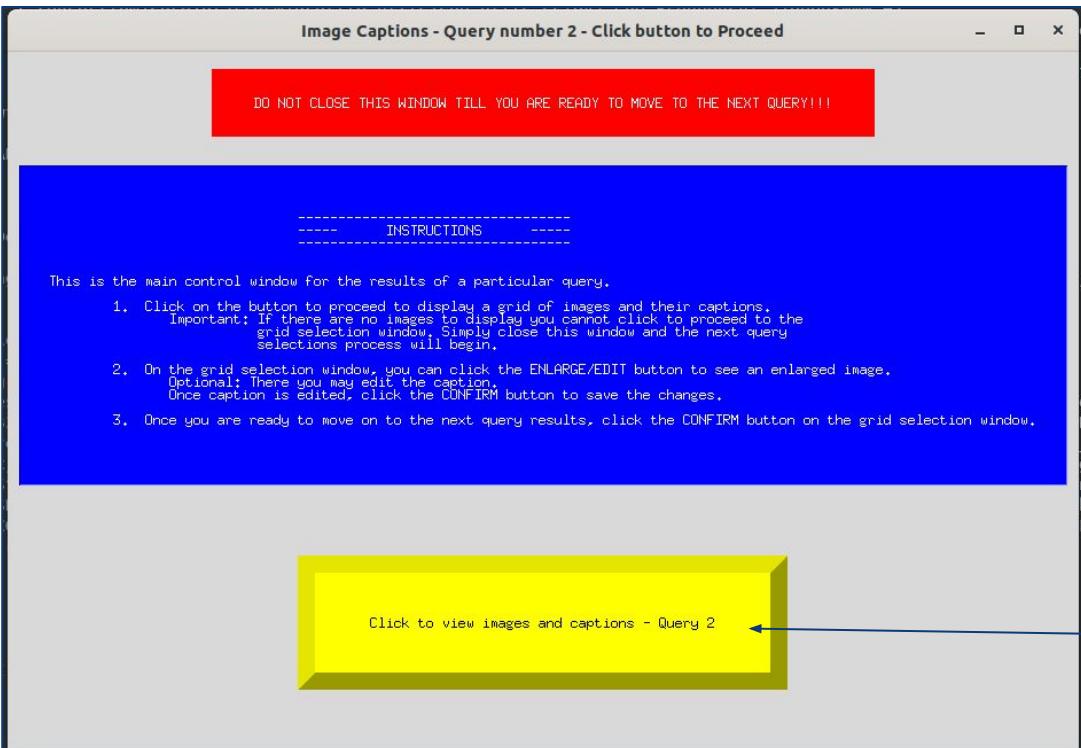
Disabled button



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Main window - query with hits for images



Clickable button - to go to next subwindow



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Subwindow for images with original captions - maximum possible images displayed for this query



Image thumbnails on far left: Clickable to deselect with border color change

Image absolute path

Original Caption for Image (from earlier inference stage)

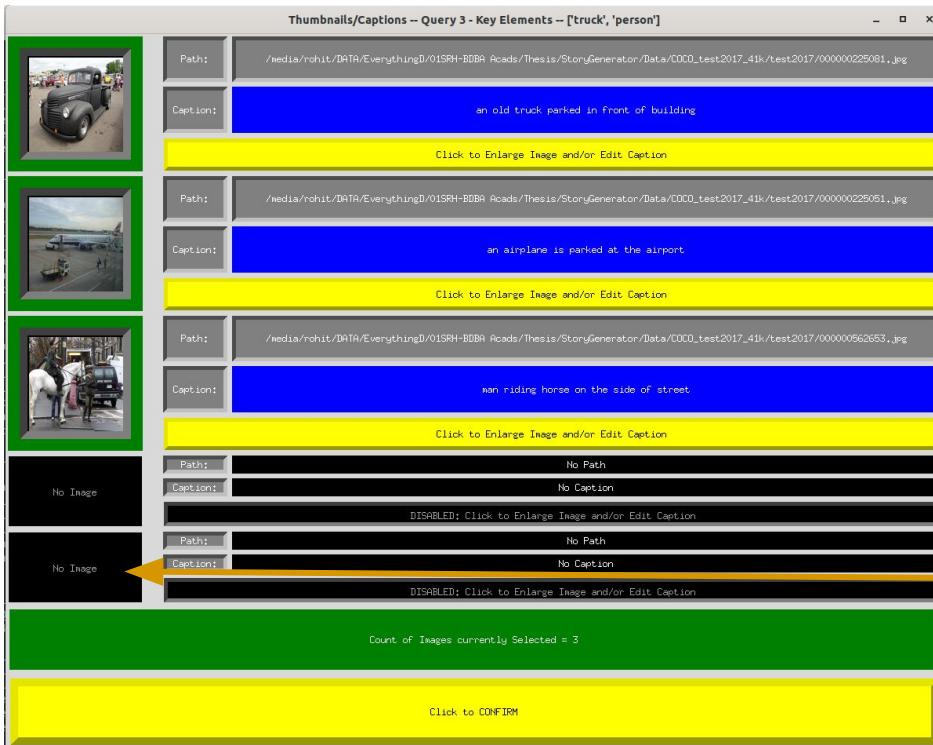
Clickable button to Enlarge Image + Edit Caption

Count of currently selected Images (defaults to ALL)

Final Confirm button

# Image Captioning - GUI Flow

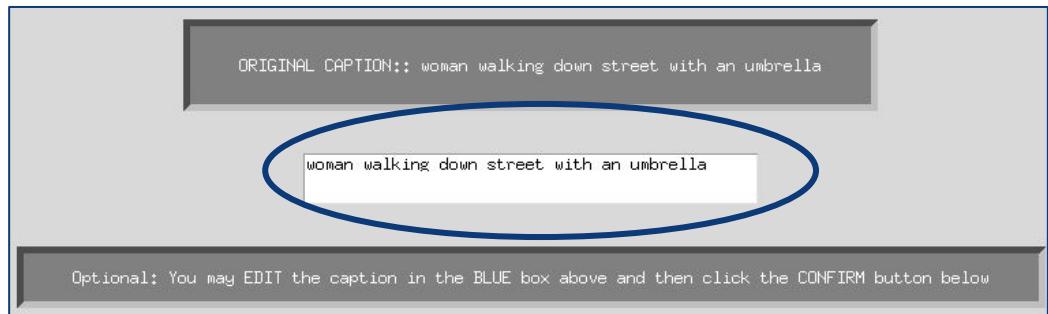
- Subwindow for images with original captions - Only 3 images selected in this query



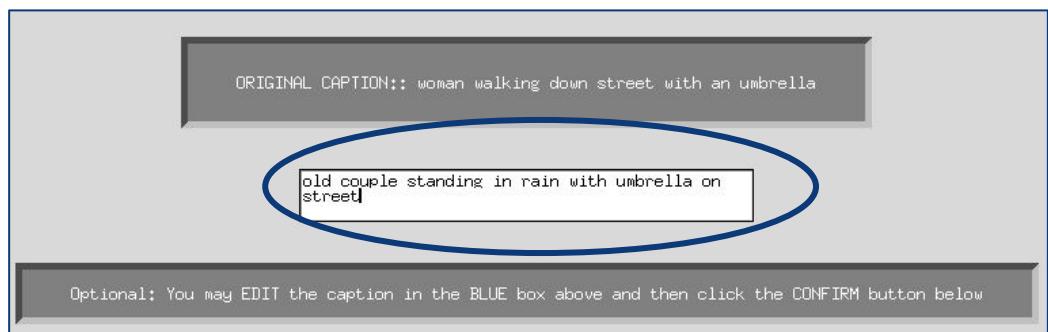
- Last two placeholders not populated
- All relevant buttons disabled

# Image Captioning - GUI Flow

- New window: Enlarged image + original caption + editable caption (for optional correction)



User edited the caption in the white box; Then clicks yellow button to CONFIRM



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- User edited caption of second image and returned to this window

Thumbnails/Captions – Query 2 - Key Elements – [car]

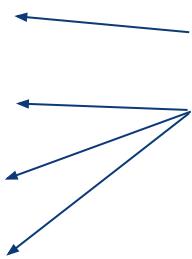
Path:	Caption:
/media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000003825.jpg	group of people standing around an open car accident. Click to Enlarge Image and/or Edit Caption
/media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000146856.jpg	old couple standing in rain with umbrella on street. Click to Enlarge Image and/or Edit Caption
/media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000312777.jpg	train is travelling down the tracks in the city. Click to Enlarge Image and/or Edit Caption
/media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000320027.jpg	red stop sign sitting on the side of road. Click to Enlarge Image and/or Edit Caption
/media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000009717.jpg	group of people riding motorcycles down street. Click to Enlarge Image and/or Edit Caption

Count of Images currently Selected = 5

Click to CONFIRM



- Old caption replaced with new one
- Color changed from Blue to Yellow
- Remaining captions unchanged with Blue backgrounds



Path: /media/rohit/DATA/EverythingD/01SRH-BDBA Roads/Thesis/StoryGenerator/Data/COCO\_test2017\_41k/test2017/00000146856.jpg

Caption: old couple standing in rain with umbrella on street

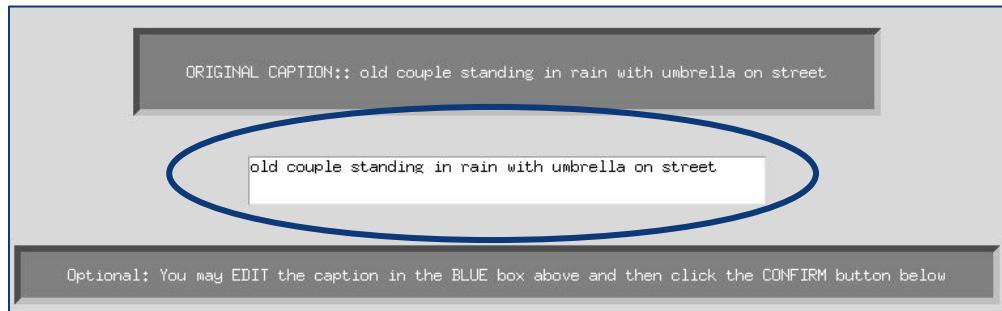
Click to Enlarge Image and/or Edit Caption

# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE



- Already edited caption reflects in window if re-entered



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

thumbnails/Captions – Query 2 - Key Elements - [car]

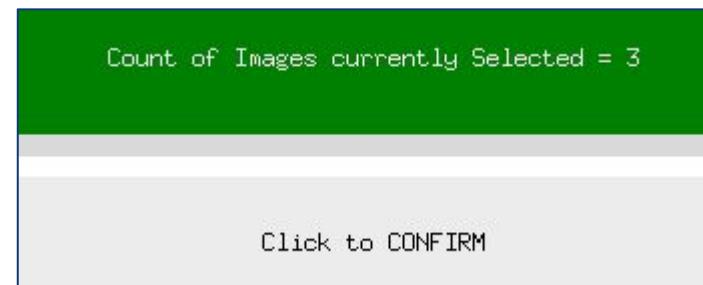
Path:	Caption:
/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg	group of people standing around an open air market
Click to Enlarge Image and/or Edit Caption	
/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000146856.jpg	old couple standing in rain with umbrellas
Click to Enlarge Image and/or Edit Caption	
/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000313777.jpg	tram is traveling down the tracks in the city center
Click to Enlarge Image and/or Edit Caption	
/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000320027.jpg	red stop sign sitting on the side of road
Click to Enlarge Image and/or Edit Caption	
/media/rohit/DATA/EverythingD/01SRH-BDBA_Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000509717.jpg	group of people riding motorcycles down street
Click to Enlarge Image and/or Edit Caption	

Count of Images currently Selected = 3

Click to CONFIRM

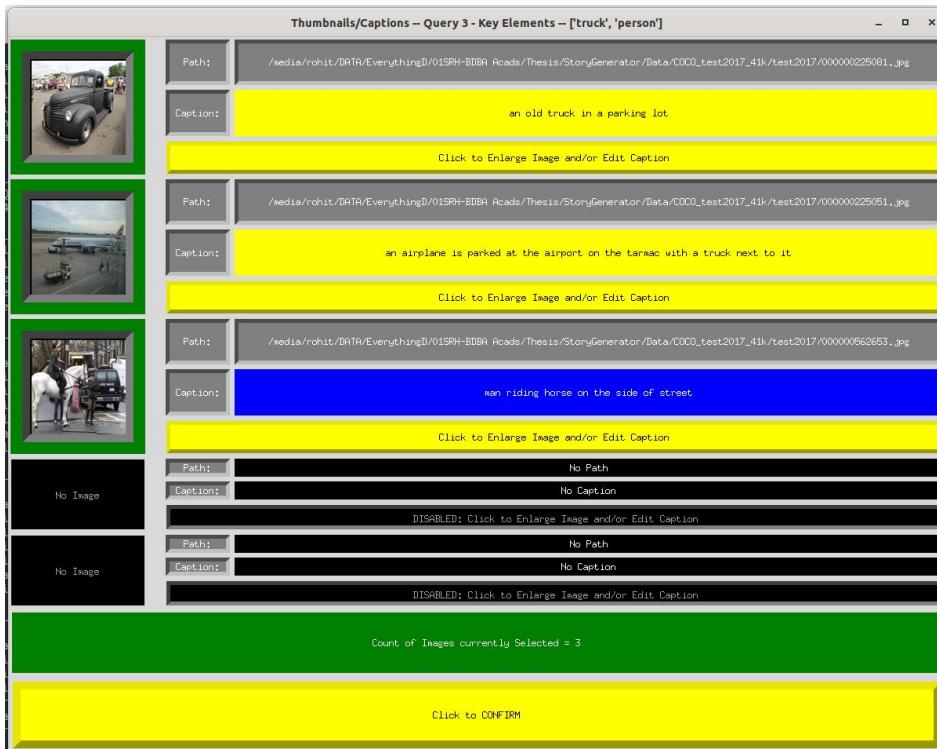
- For Query 2:

- > User edited two captions (for Image 2 and 3)
- > User Deselected last two images



# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE



- For Query 3:
  - > User edited two captions (for images 1 and 2)
  - > User did not Deselect any image, so all images remained selected

# Image Captioning - GUI Flow

STAATLICH  
ANERKANNTE  
HOCHSCHULE

- Query 1: No changes
- Query 2: 3 of 5 images selected (with 2 caption edits)
- Query 3: All images selected (with 2 caption edits)

```
*****
***** SUMMARY for Image Captioning GUI - all queries *****
***** AFTER removing the deselected images *****
*****
Data structure BEFORE =
[{'key_elements': [], 'selected_images': []}, {'key_elements': ['car'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg', 'group of people standing around an open air market'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000146856.jpg', 'woman walking down street with an umbrella'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000313777.jpg', 'train is traveling down the tracks in the city'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000320027.jpg', 'red stop sign sitting on the side of road'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000509717.jpg', 'group of people riding motorcycles down street']]], {'key_elements': ['truck', 'person'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225081.jpg', 'an old truck parked in front of building'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225051.jpg', 'an airplane is parked at the airport'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000562653.jpg', 'man riding horse on the side of street']]}}
Deselected image positions all queries = [[], [3, 4], []]
Data structure AFTER GUI; changed captions and removing Deselected images =
[{'key_elements': [], 'selected_images': []}, {'key_elements': ['car'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg', 'group of people standing around an open air market'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000146856.jpg', 'old couple standing in rain with umbrella'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000313777.jpg', 'tram is traveling down the tracks in the city center']]], {'key_elements': ['truck', 'person'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225081.jpg', 'an old truck in a parking lot'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225051.jpg', 'an airplane is parked at the airport on the tarmac with a truck next to it'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000562653.jpg', 'man riding horse on the side of street']]}}
LOG_LEVEL INFO :: 
After GUI TO DISPLAY IMAGE CAPTIONING RESULTS logic execution:
gui_img_cap_show_results_logic_RC = 0
gui_img_cap_show_results_logic_msg =
None
gui_img_cap_module_results =
[{'key_elements': [], 'selected_images': []}, {'key_elements': ['car'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg', 'group of people standing around an open air market'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000146856.jpg', 'old couple standing in rain with umbrella'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000313777.jpg', 'tram is traveling down the tracks in the city center']]], {'key_elements': ['truck', 'person'], 'selected_images': [['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225081.jpg', 'an old truck in a parking lot'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225051.jpg', 'an airplane is parked at the airport on the tarmac with a truck next to it'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000562653.jpg', 'man riding horse on the side of street']]}}
LOG_LEVEL INFO :: 

Normal exit from program.
```

Data structures before and after this stage - Console output

# Image Captioning - GUI Flow

- Query 1: No changes
- Query 2: 3 of 5 images selected (with 2 caption edits)
- Query 3: All images selected (with 2 caption edits)

```
*****
***** SUMMARY for Image Captioning GUI - all queries *****
***** AFTER removing the deselected images *****
*****

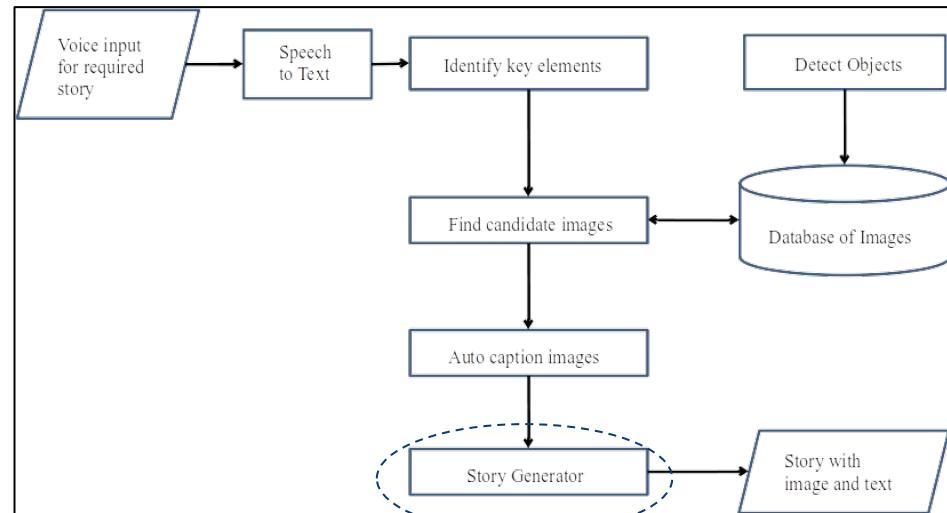
Data structure BEFORE =
[{'key_elements': [], 'selected_images': []}, {'key_elements': ['car'], 'selected_images': ["/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg", 'group of people standing around an open air market'],
['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000146856.jpg', 'woman walking down street with an
umbrella'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000313777.jpg', 'train is traveling down the tracks
in the city'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000320027.jpg', 'red stop sign sitting on the side
of road'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000509717.jpg', 'group of people riding motorcycles
down street']}], {'key_elements': ['truck', 'person'], 'selected_images': ["/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225081.jpg", 'an old truck parked in front of building'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225051.jpg', 'an airplane is parked at the airport'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000562653.jpg', 'man riding horse on the side of street']]}

Deselected image positions all queries = [[], [3, 4], []]

Data structure AFTER GUI; changed captions and removing Deselected images =
[{'key_elements': [], 'selected_images': []}, {'key_elements': ['car'], 'selected_images': ["/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/00000033825.jpg", 'group of people standing around an open air market'],
['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000146856.jpg', 'old couple standing in rain with umbrella'],
['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000313777.jpg', 'tram is traveling down the tracks in the city
center']]}, {'key_elements': ['truck', 'person'], 'selected_images': ["/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225081.jpg", 'an old truck in a parking lot'], ['/media/rohit/DATA/EverythingD/01SRH-BDBA
Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000225051.jpg', 'an airplane is parked at the airport on the tarmac with a truck next to it'],
['/media/rohit/DATA/EverythingD/01SRH-BDBA Acads/Thesis/StoryGenerator/Data/COCO_test2017_41k/test2017/000000562653.jpg', 'man riding horse on the side of street']]}
```

Data structures before and after this stage - text extracted for readability

# Story Generator Block - PENDING



# Changes to Expose

1. Limit the names of objects found from images in database:
  - a. Only the labels of the object detection model can be “found” in the images.
  - b. Implication: User must speak using exactly these labels while structuring the input sentences.
  - c. Currently: 80 labels part of the detection database
    - i. labels = ['aeroplane', 'apple', 'backpack', 'banana', 'baseball bat', 'baseball glove', \  
'bear', 'bed', 'bench', 'bicycle', 'bird', 'boat', 'book', 'bottle', 'bowl', \  
'broccoli', 'bus', 'cake', 'car', 'carrot', 'cat', 'cell phone', 'chair', \  
'clock', 'cow', 'cup', 'diningtable', 'dog', 'donut', 'elephant', 'fire hydrant', \  
'fork', 'frisbee', 'giraffe', 'hair drier', 'handbag', 'horse', 'hot dog', \  
'keyboard', 'kite', 'knife', 'laptop', 'microwave', 'motorbike', 'mouse', \  
'orange', 'oven', 'parking meter', 'person', 'pizza', 'pottedplant', \  
'refrigerator', 'remote', 'sandwich', 'scissors', 'sheep', 'sink', 'skateboard', 'skis', \  
'snowboard', 'sofa', 'spoon', 'sports ball', 'stop sign', 'suitcase', 'surfboard', \  
'teddy bear', 'tennis racket', 'tie', 'toaster', 'toilet', 'toothbrush', 'traffic light', \  
'train', 'truck', 'tvmonitor', 'umbrella', 'vase', 'wine glass', 'zebra']
  - d. Will call out this limitation out in proposal
2. Voice input simulated using wav files to represent each of the three sentences. These need to be recorded separately and then presented to the system.
  - a. Implication: User needs to record their input externally (e.g. using Audacity, etc) to create a wav file in required format (16kHz sampling, 16-bit, Mono).
  - b. Only if time permits, will try to automate this aspect to directly record users speech and include wav file creation as part of pipeline.

# Changes to Expose

3. While recording the audio files, the user must speak in “active” and not “passive” voice:
  - a. E.g. “person is walking his dog”, NOT “dog is being walked by a person”
    - i. Shorter and cleaner sentence for model.
    - ii. Generally natural speech is in Active voice, so language models will have more training on such sentences.
    - iii. Less chance of transcription errors.
  - b. Only if time permits will relax this criterion later on.
4. User will be presented opportunities at various stages in pipeline to edit the data being processed:
  - a. Keywords identification - select exactly 1/ 2/ 3 words per input wav file.
  - b. Images extracted from database - view them and deselect images if required.
    - i. Maximum limit of selection = 5 images.
    - ii. Remove an image due to false detection of object
5. Limiting the types of words and number of words processed as keywords for the “Identify key elements” block:
  - a. Irrespective of what all was said by user in the input wav file, only the Noun words in the transcription of each wav file will be processed.
  - b. Thus increase chance of finding a suitable image with all the objects present.

# Changes to Expose

6. Only English language is allowed.
7. Images passed to the “Auto caption images” block will be limited to 5 images per input sentence.
8. Motivation: what exactly to add?
  - a. Easy way to provide short stories on the fly for children
  - b. User (usually parent) controls what the story says. If the child slightly older (and speaks clearly) they can ask for stories that interest them.
  - c. No paper, no delivery of the book, unlimited stories
9. Inputs from Prof. Sprick during 23.07 meeting:
  - a. Motivation structure above okayed.
  - b. Make overall story process even more interactive:
    - i. Store sounds from typical keywords. For example:
      - > Dog: “dog barking”
      - > Truck: “truck honking”, or “truck going past on road”, “truck engine starting”
      - > Spoon: “sounds of cutlery being used”
    - ii. Once story is ready, use a Text-to-speech (TTS) block to “speak the story”. Superimpose suitable sounds in background when appropriate word is being spoken.
    - iii. Accepted idea as it is excellent. But will only be attempted if time permits.

# Changes to Expose

6. inputs from Mr. Frank Schulz during 31.07 meeting:
  - a. Once overall models working together, use the pipeline parameters as part of research question to gauge efficacy of the use-case implementation.
  - b. For example:
    - i. how many images should be sent to auto-caption
    - ii. How many words per input audio file should be kept as the candidate key elements
7. Each audio input file (or user input via mic if possible) to be exactly one sentence. Thus model expects exactly three input sentences as the start point user input.

# References

1. How to Perform Object Detection With YOLOv3 in Keras.  
<https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/>
2. Github Code <https://github.com/rbewoor/keras-yolo3> ( forked from <https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/> on 05.06.2020)
3. YOLOv3 pre-trained weights. <https://pjreddie.com/media/files/yolov3.weights>
4. J. Redmon et al. You Only Look Once: Unified, Real-Time Object Detection. 09-05-2016. <https://arxiv.org/abs/1506.02640>
5. J. Redmon et al. YOLO9000: Better, Faster, Stronger. 25-12-2016. <https://arxiv.org/abs/1612.08242>
6. J. Redmon et al. YOLOv3: An Incremental Improvement. 08-04-2018. <https://arxiv.org/abs/1804.02767>
7. How to Visualize a Deep Learning Neural Network Model in Keras.  
<https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/>
8. All About YOLO Object Detection and its 3 versions (Paper Summary and Codes).  
<https://medium.com/data-science-in-your-pocket/all-about-yolo-object-detection-and-its-3-versions-paper-summary-and-codes-2742d24f56e>
9. YOLO v3 theory explained, <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193> as on 10.06.2020
10. M. Tanti et al. What is the Role of RNNs in an Image Caption Generator?. 07.08.2017. <https://arxiv.org/abs/1708.02043>
11. X. Liu et al. A survey of deep neural network-based image captioning. 09.06.2018. <https://doi.org/10.1007/s00371-018-1566-y>
12. Image Captioning with Keras,  
<https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8> as on 10.09.2020
13. Automatic Image Captioning, <https://gist.github.com/nttuan8/a621aa6700995db6db71b0a768a8552f> as on 20.09.2020