Master Thesis update

Voice input based story generation

30.09.2020 by:

Rohit Keshav Bewoor (11011831)

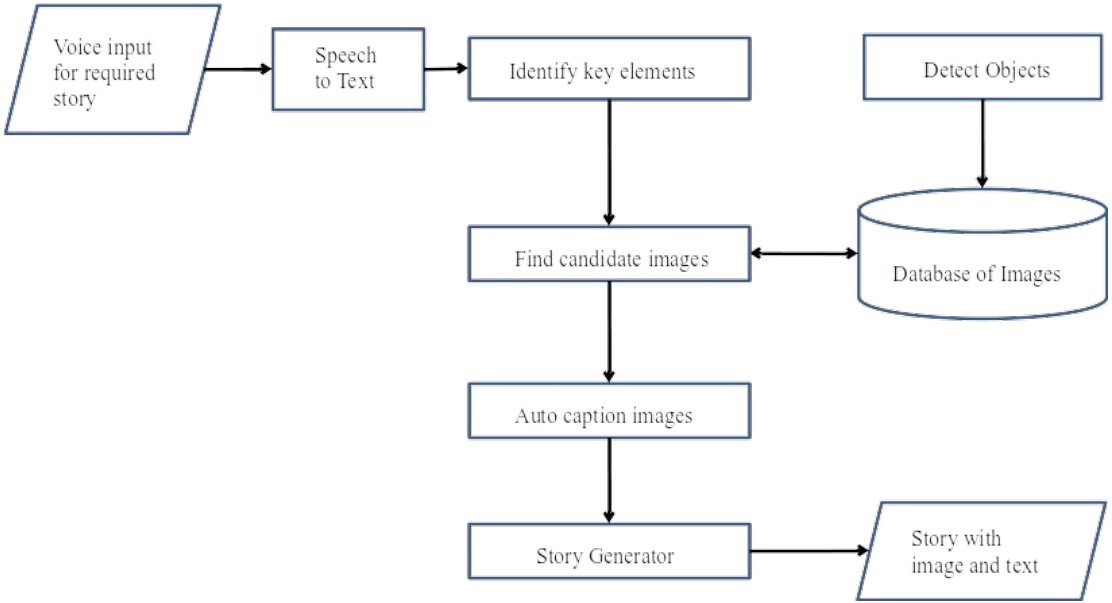
Big Data and Business Analytics 2018-20 batch
SRH Hochschule Heidelberg

SRH HOCHSCHULE HEIDELBERG
Intelligence in Learning

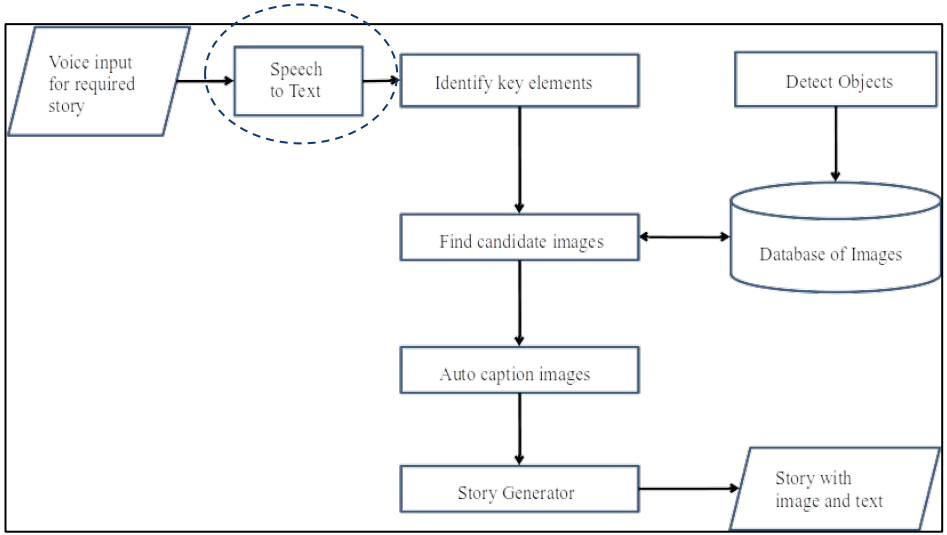
Block Diagram

STAATLICH ANERKANNTE HOCHSCHULE

- Objective: 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

STAATLICH
ANERKANNTE
HOCHSCHULE

- 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

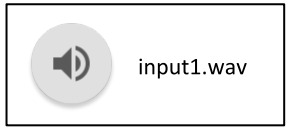
- Currently exploring open source STT implementation - Deepspeech v0.7.3 from Mozilla foundation
  - > Link: https://deepspeech.readthedocs.io/en/v0.7.3/DeepSpeech.html
  - > Using this pre-trained model for inference
  - > RNN model with 5 hidden layers
  - > Accepts audio file as input and outputs text transcript
    - > Input Audio format: wav file, 16kHz sampling, mono channel, 16-bit
  - > Output characters set {a,b,c,...,z,apostrophe}

Speech to Text

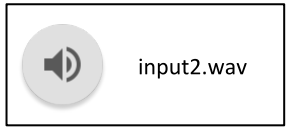
STAATLICH ANERKANNTE HOCHSCHULE

- Tested on my own voice with three audio files

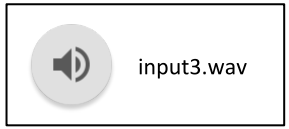
- input1.wav transcript: Errors in output transcript
  - > I said:
    Make me a story about persons sitting at a table. They are playing cards.
  - > Model Inference:
    me me a tory about persons sitting at table the blanchards



- 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



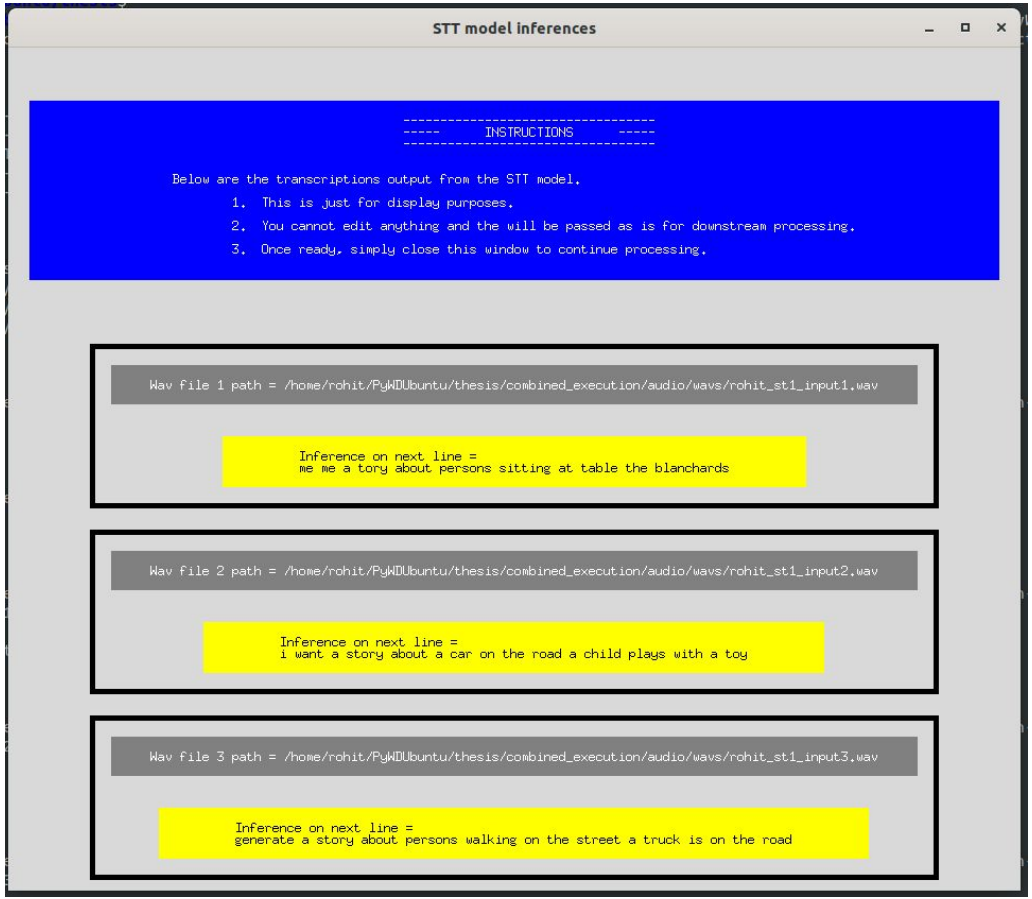
Speech to Text

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.
  - > **Doubt that this is foolproof and can work reliably - see example later!**

- 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

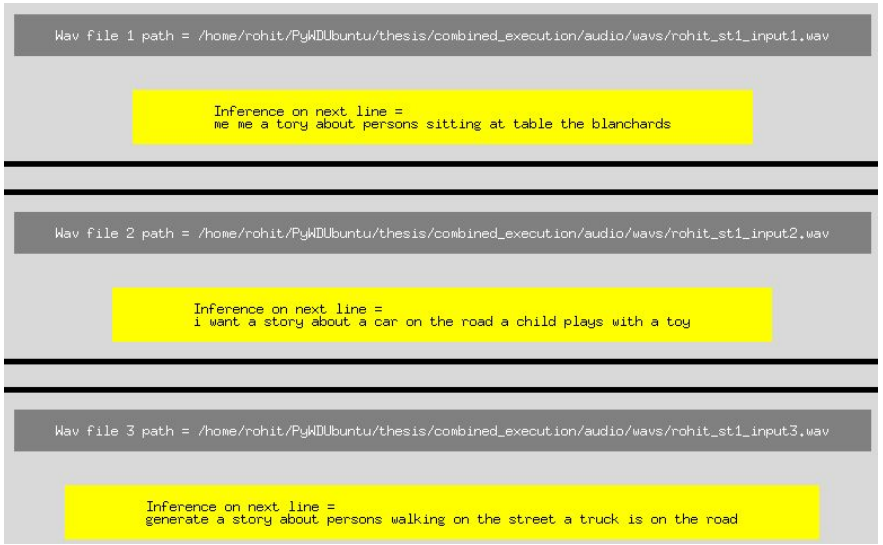


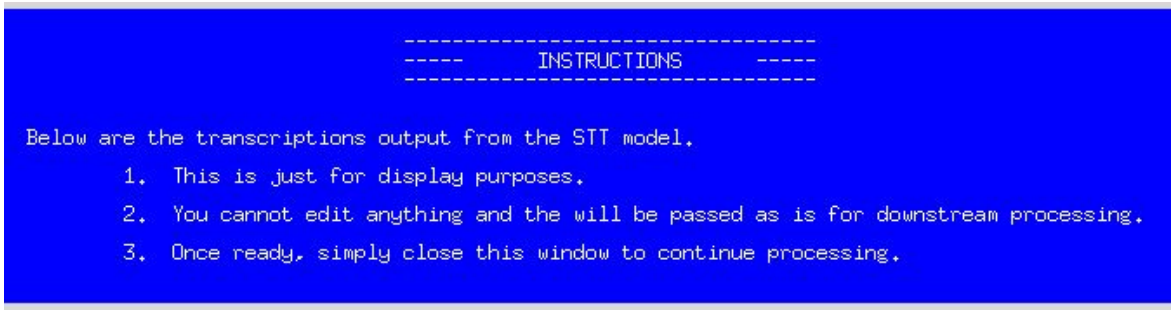
- GUI displays the output from STT
- Nothing editable
  - > Only for display purpose
  - > data is sent as is downstream

Speech to Text - GUI

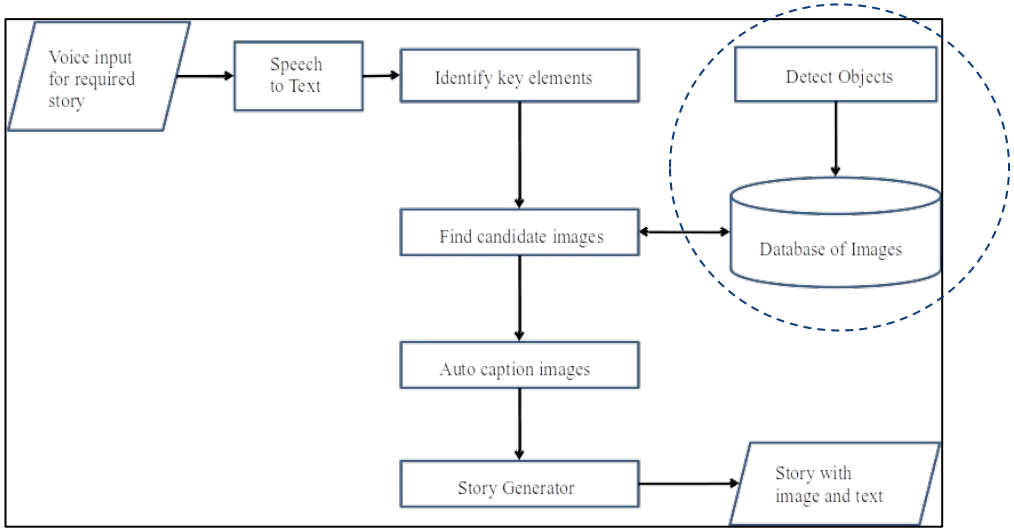
STAATLICH ANERKANNTE HOCHSCHULE

- Transcriptions and the instructions panels





Database creation using Object Detection Block



Objection Database creation

STAATLICH ANERKANNTE HOCHSCHULE

- Objective: 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

Neo4j database after population

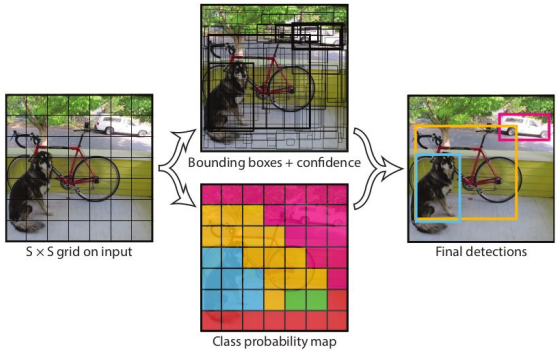
- Using You Only Look Once version 3 (YOLOv3): a single stage detector
  - > Implemented YOLOv3 detector using parallel processing
  - > On personal laptop takes 5 minutes to process 127 images
  - > Much faster with multi-core version

- Due to space constraints, the output inference image is discarded and not saved for later use

How YOLO works

STAATLICH
ANERKANNTE
HOCHSCHULE

- Divides image into a grid of size S x 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.

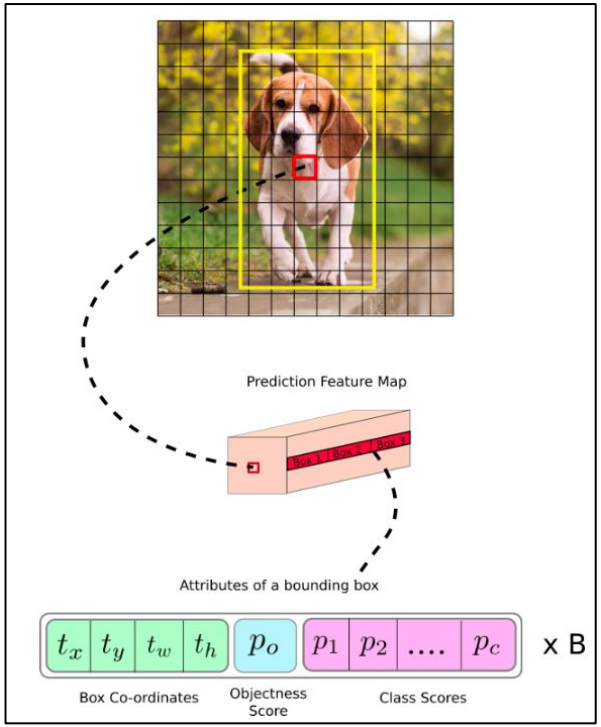


Source: YOLO v1 paper: https://arxiv.org/abs/1506.02640

How YOLO works

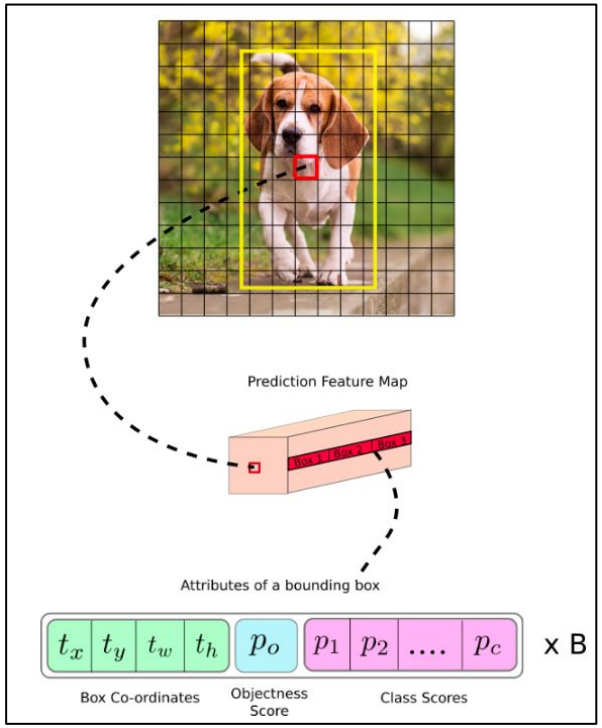
STAATLICH ANERKANNTE HOCHSCHULE

- Usually Non-max suppression used to remove redundant detections
- Total detections per image = (S x S) * (B * (5 + C))
  - Each bounding box has 5 + C attributes
- For example, suppose that:
  - image is divided into 13 x 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 x 13) * (3 * (5 + 80)) = 13 x 13 x 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



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

- Image divided into a grid of S x S grid-boxes
  - Predictions made at 3 scales where a 416 x 416 pixel input image is divided into 13x13, 26x26 and 52x52
- 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

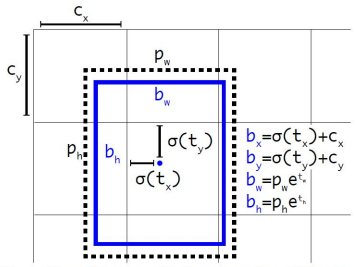


Figure 2. **Bounding boxes with dimension priors and location prediction.** We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized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

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

Use Case for YOLO

STAATLICH
ANERKANNTE
HOCHSCHULE

Objective:
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 has property called "data" to identify which dataset the image belongs to. E.g. coco_train_2017
- HAS relationship has a property called "score" to hold the confidence score

E.g. Image123.jpg HAS the objects:

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

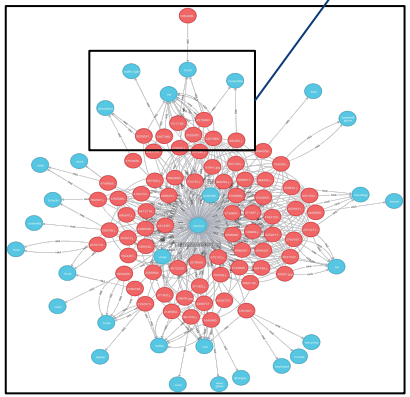
Note: Used a threshold score of 0.45

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

Neo4j representation

STAATLICH ANERKANNTE HOCHSCHULE

Neo4j db after inserts



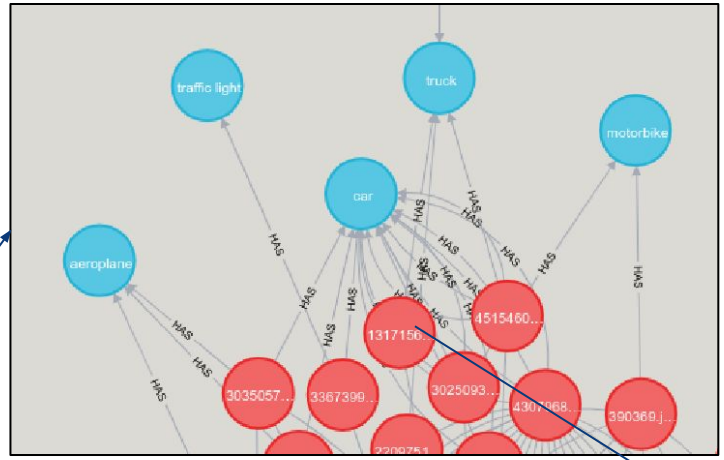


Image (red node) - HAS (line) -> Object (green node)

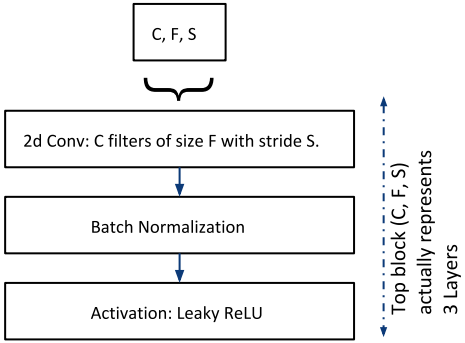
Objects found in images: Traffic light, truck, motorbike, car, etc.

Many to many relationship could exist.

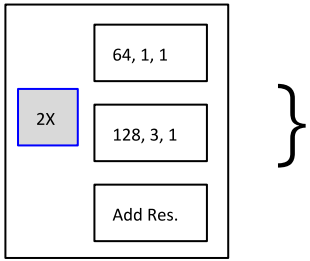
Detection output: 1317156_det.jpg

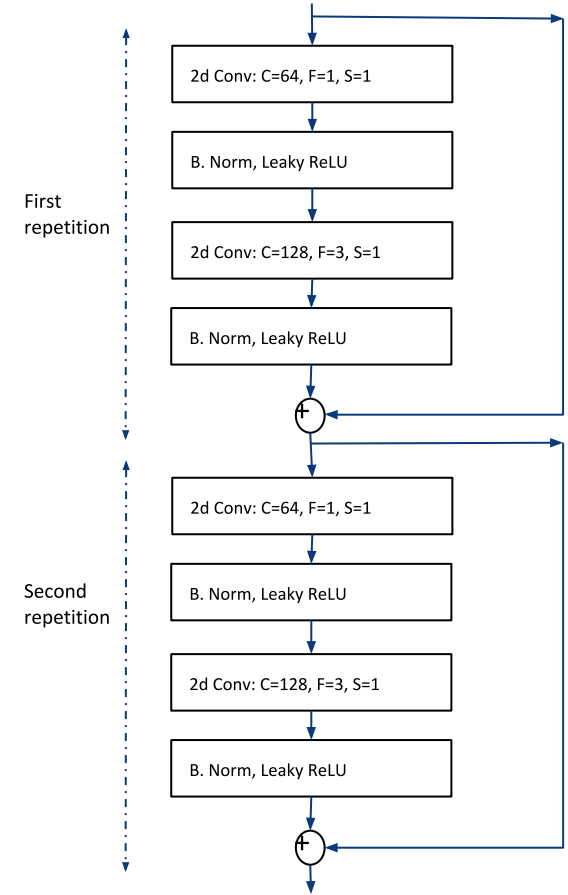
Model description - elementary blocks

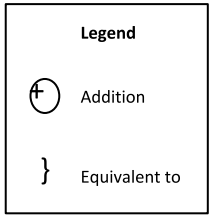
- Series of Convolution, then Batch Normalization, then Leaky ReLU. This is repeated many times.
- Convolution type:
  - Stride = 1, then “same”
  - Stride = 2, then “valid”
- Convolution parameters:
  - C: #Filters
  - F: Filter size
  - S: Stride



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

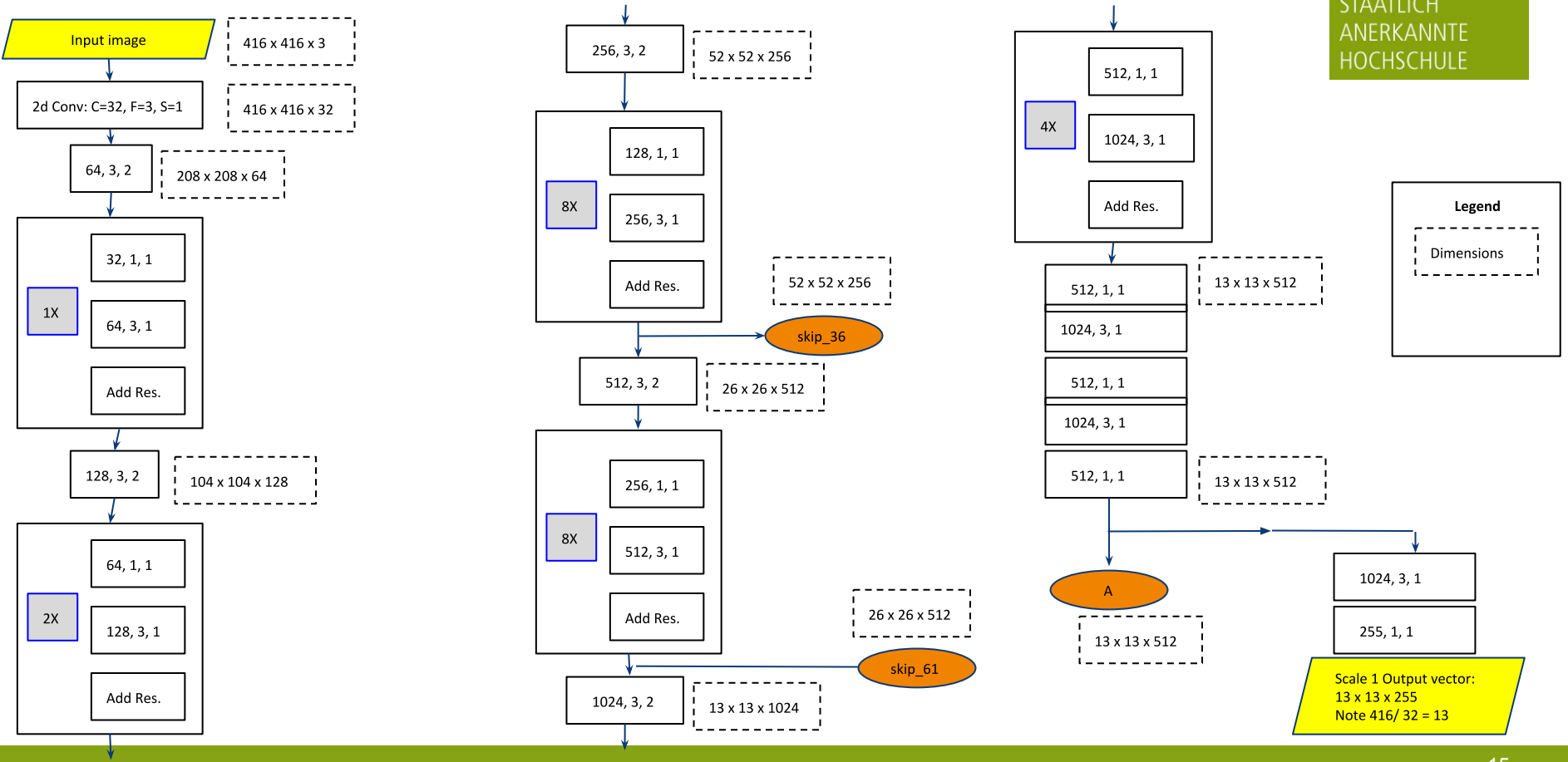






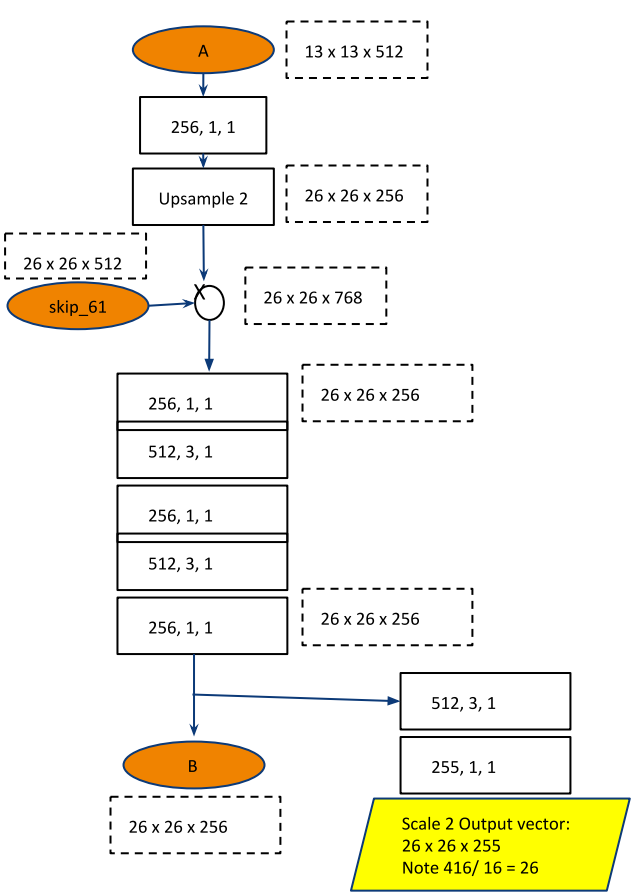
STAATLICH ANERKANNTE HOCHSCHULE

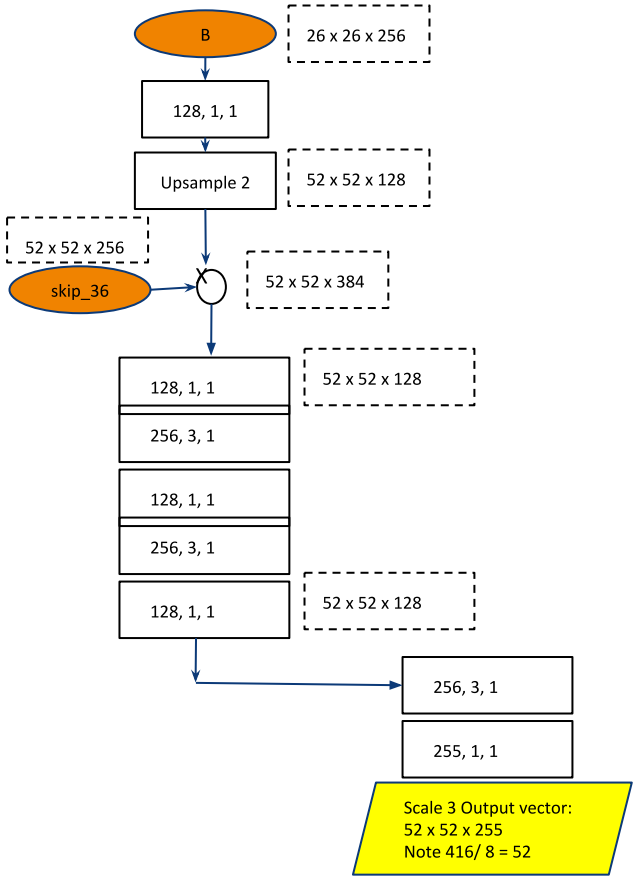
Model description - i/p o/p feature sizes

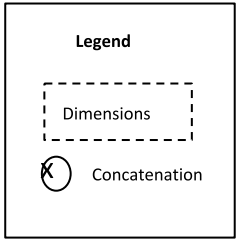


STAATLICH ANERKANNTE HOCHSCHULE

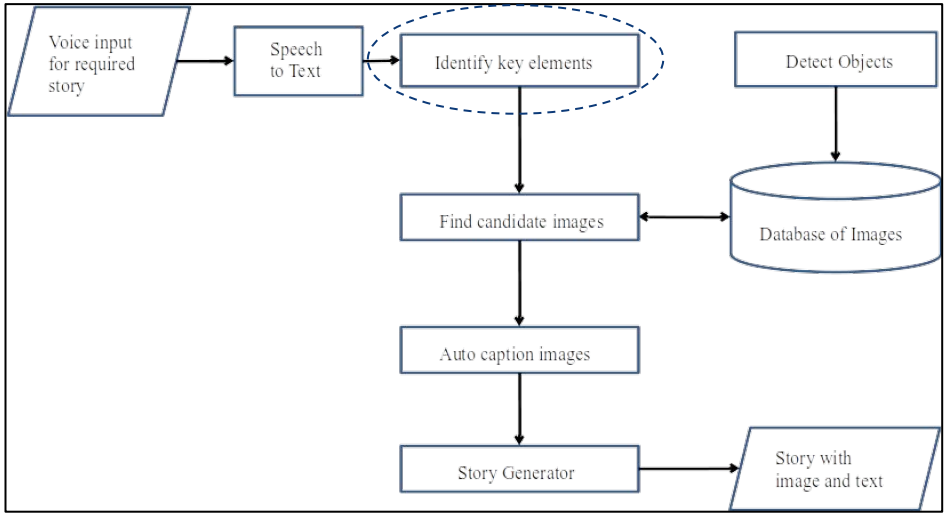
Model description - i/p o/p feature sizes







Identify Key Elements Block



Identify Key Elements

STAATLICH ANERKANNTE HOCHSCHULE

- 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

- Logic performed using Spacy:
  - > Word tokenization Remove all stop-words POS tagging
  - > Keeping only Noun type words of two 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

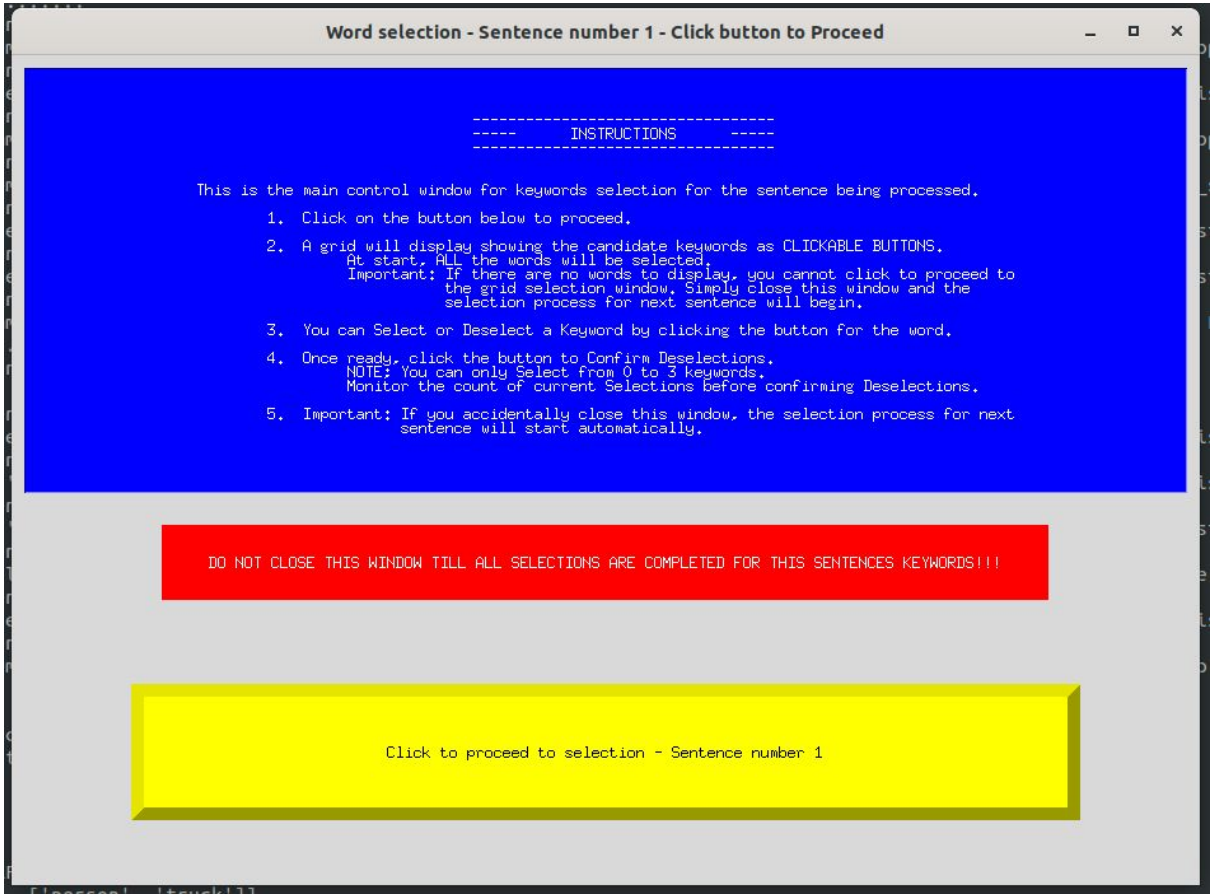
STAATLICH ANERKANNTE HOCHSCHULE

- Sentence tokenization is not reliable.
  - > Input had two sentences from the output of STT block
  - > Spacy: got one wrong got one right
  - > NLTK: got both wrong

```
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
```

```
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']]
```

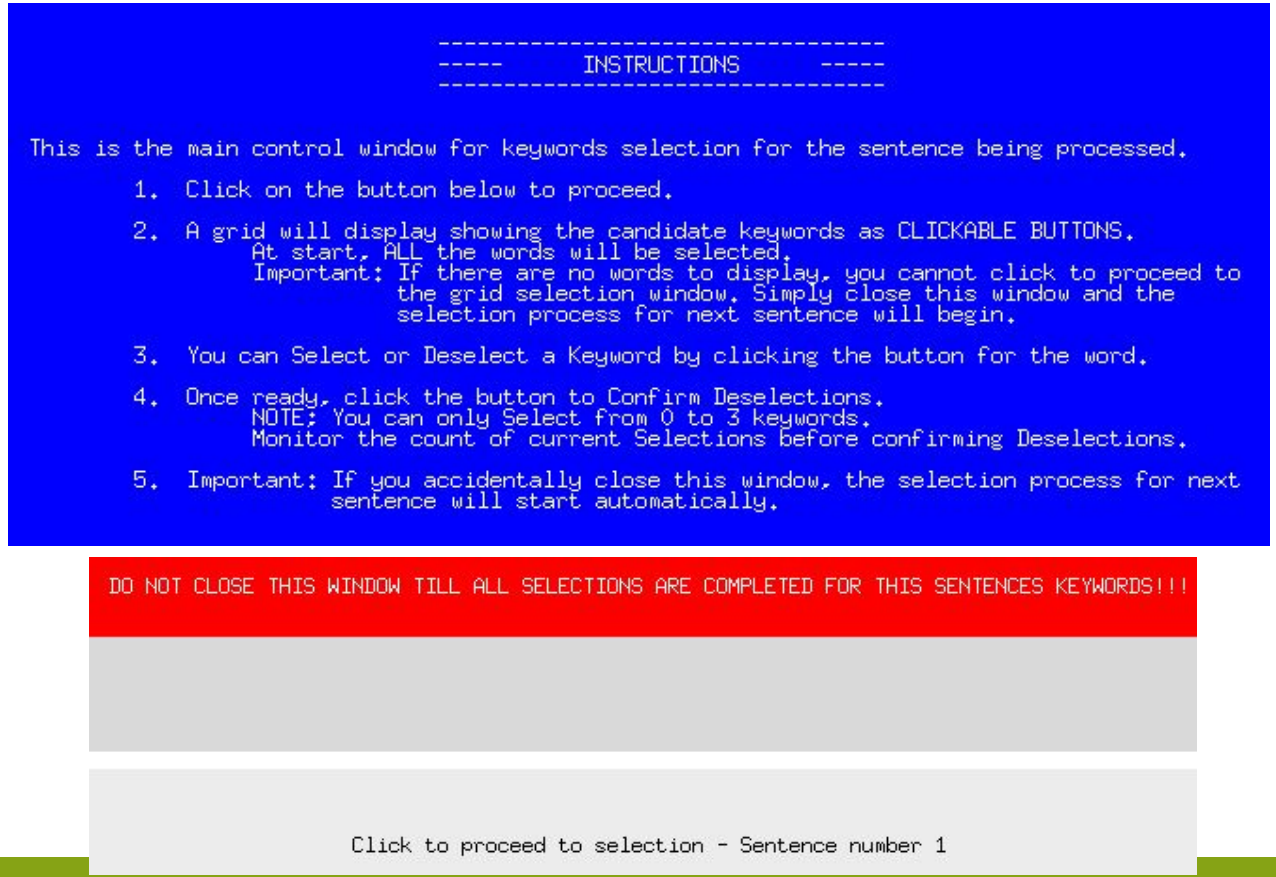
Identify Key Elements - GUI flow



- 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

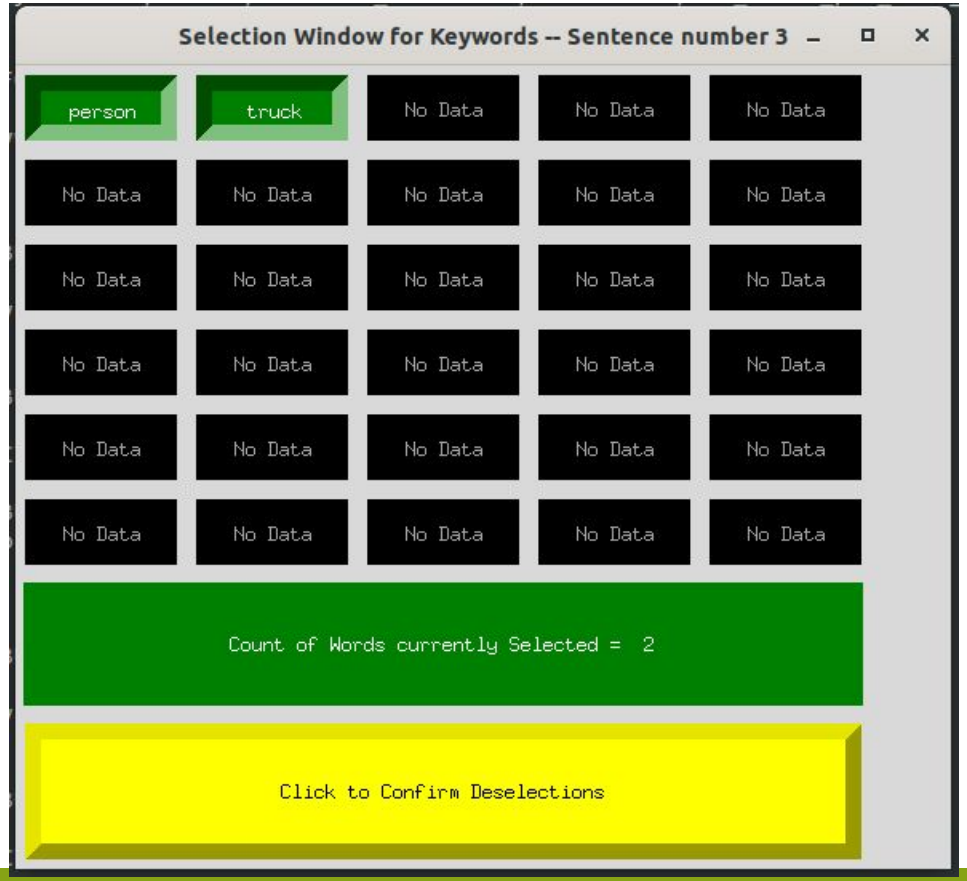
Identify Key Elements - GUI flow

STAATLICH
ANERKANNTE
HOCHSCHULE



Identify Key Elements - GUI flow

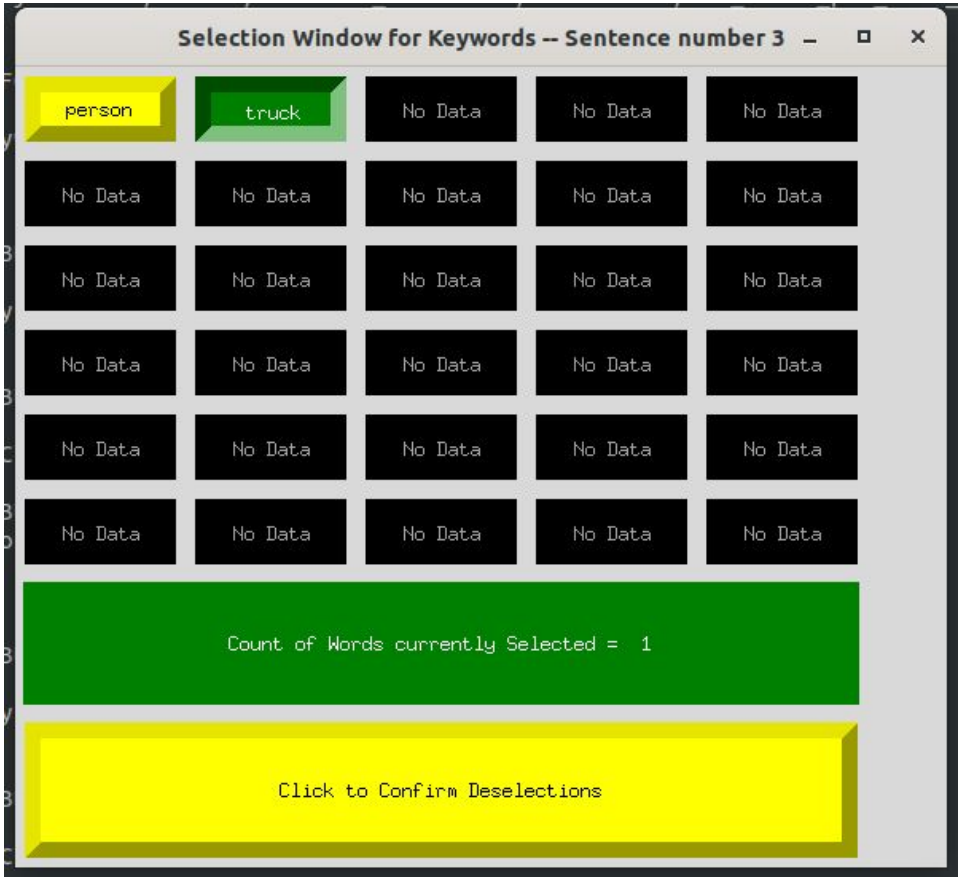
STAATLICH ANERKANNTE HOCHSCHULE



- Can handle up to 30 candidate keywords
- Here 2 candidate keywords shown, remaining placeholders indicate No Data. These are the non-stopwords of type noun which matched the existing names of objects stored in database.
- All are Selected by default.
- Current count shows value 2.
  - > Invalid option for Selections count (0 to 3 only allowed) indicated by color
  - > Color is Green (not Red)
- Confirm Selections button is enabled

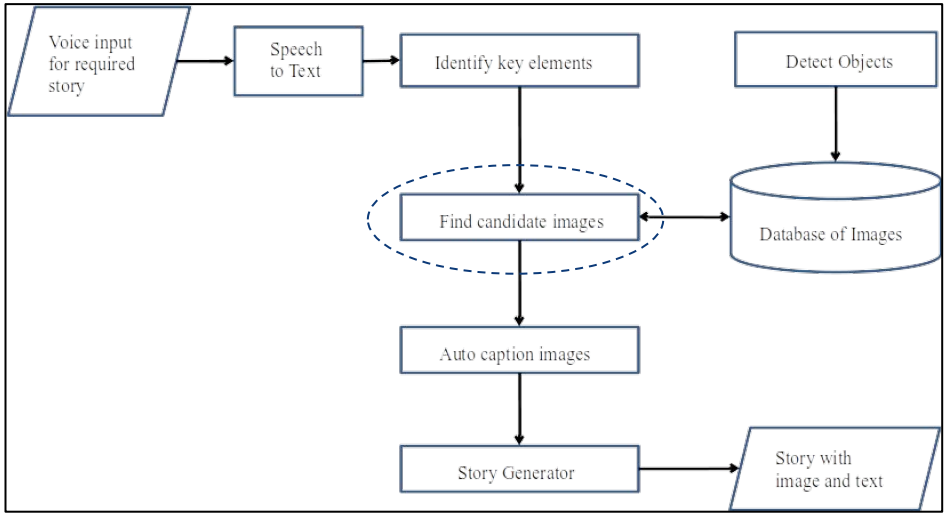
Identify Key Elements - GUI flow

STAATLICH ANERKANNTE HOCHSCHULE



- 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

STAATLICH ANERKANNTE HOCHSCHULE

- Goal: Using the user selected final keywords from previous stage, query the database to retrieve images containing the objects of interest.

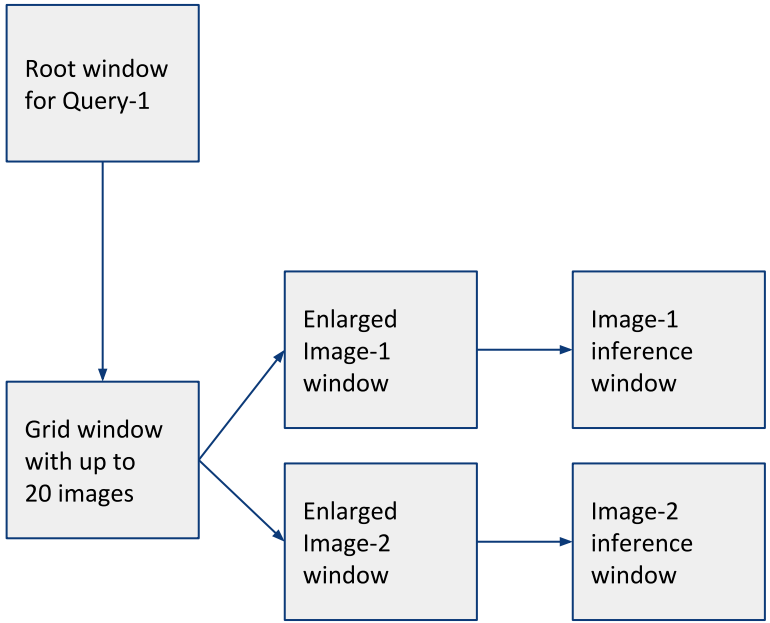
- Database query returns up to 20 images per query as candidate images.
  - > Only objects with a HAS relationship score > 90%
  - > Only from the COCO_Test2017 and Flicker8k datasets
    - > As other two were used for Yolo model training

- Allow user to view candidate images and Deselect any images:
  - > Because the inference module wrongly “found” object of interest.
  - > Reduce the total Selections count to maximum 5 images.

- User selection via GUI interface:
  - > Show thumbnails of the images.
  - > View Enlarged image for close inspection
  - > Perform one-off inference to see objects detected
  - > Image thumbnails clickable to “Select” or “Deselect”
  - > 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

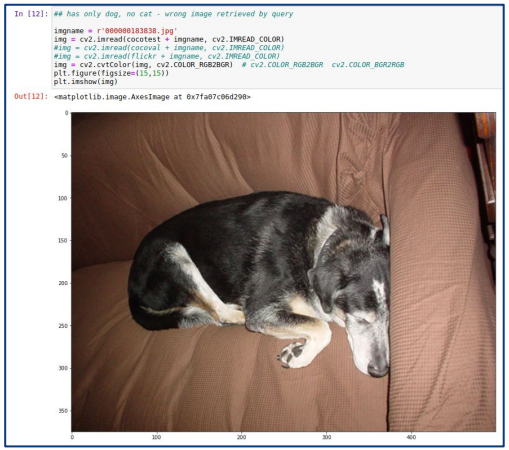


Find Candidate Images - wrong query results

STAATLICH ANERKANNTE HOCHSCHULE

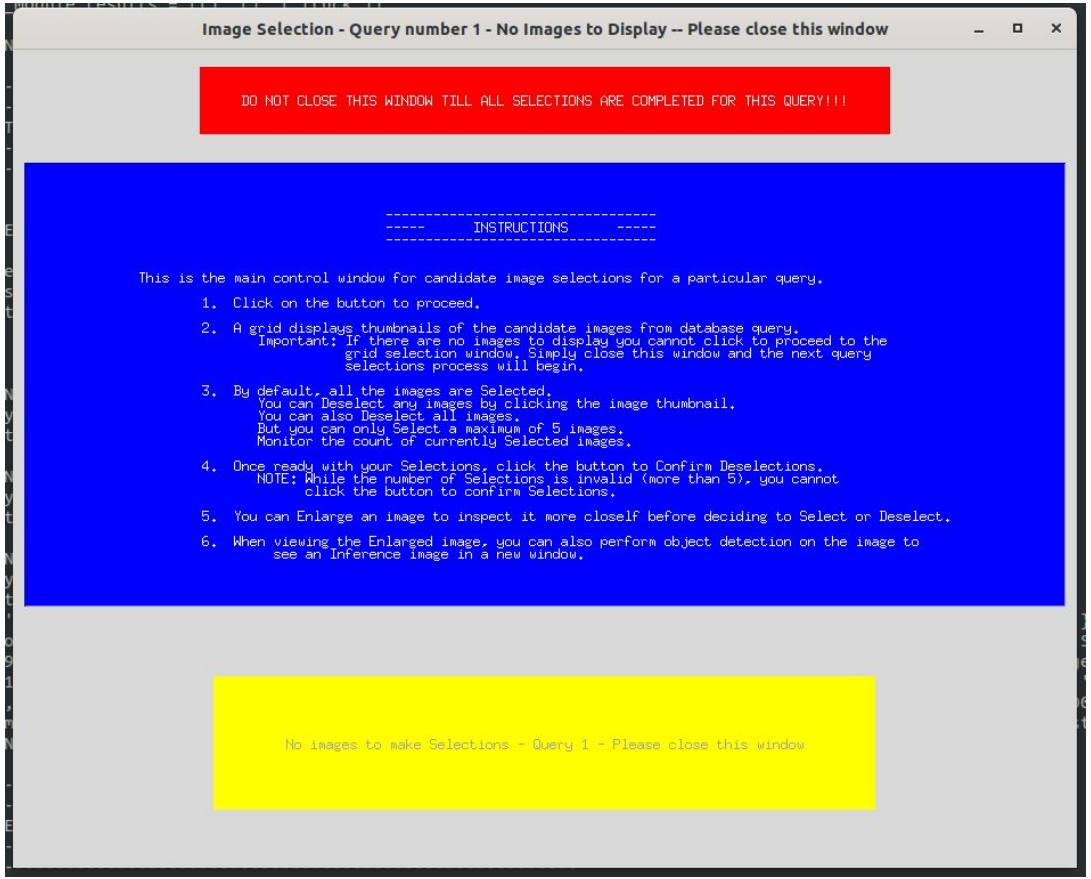
- Sometimes image results do not actually have the objects of interest.
- Example:
  - > Queries for object dog and cat
  - > First image is wrong - has only dog
  - > Second image is correct - has both cat and dog
- Therefore allow user to deselect such images
- Possible solution to explore:
  - > Play around with the threshold used in object detection stage (currently used 0.45)
  - > Implemented: Used min. threshold value of 0.90 during Neo4j query

```
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': 'co
co_test_2017'}, {'i.name': '000000341174.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000302015.jpg', 'i.da
taset': 'coco_test_2017'}, {'i.name': '000000101257.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000209673.
jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000430186.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '00
0000310779.jpg', 'i.dataset': 'coco_test_2017'}, {'i.name': '000000017269.jpg', 'i.dataset': 'coco_test_2017'},
{'i.name': '000000210056.jpg', 'i.dataset': 'coco_test_2017'}])
```



Find Candidate Images - GUI flow

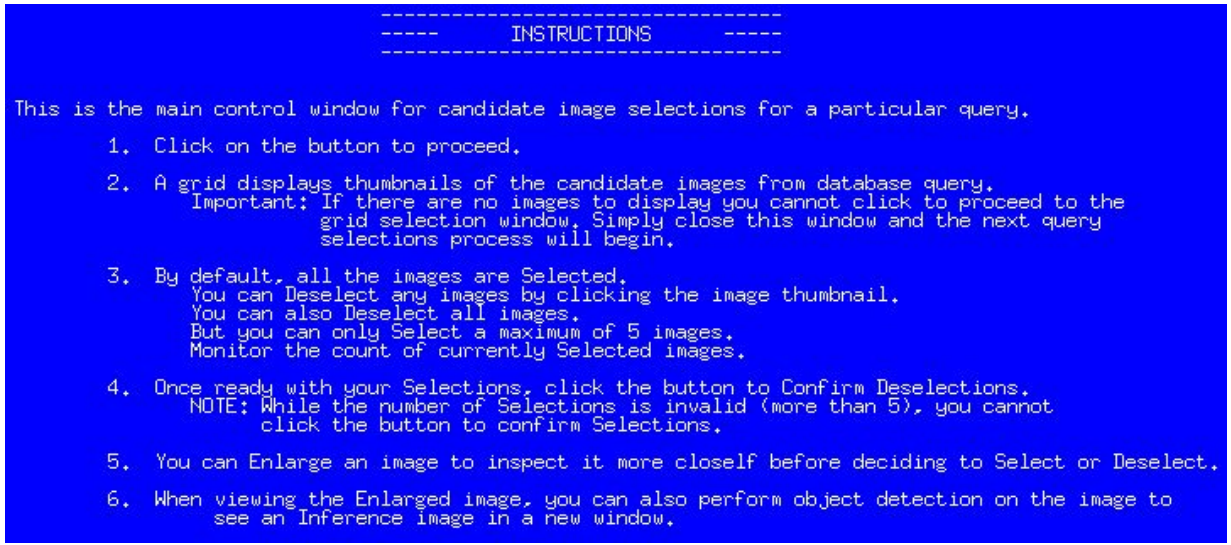
STAATLICH ANERKANNTE HOCHSCHULE



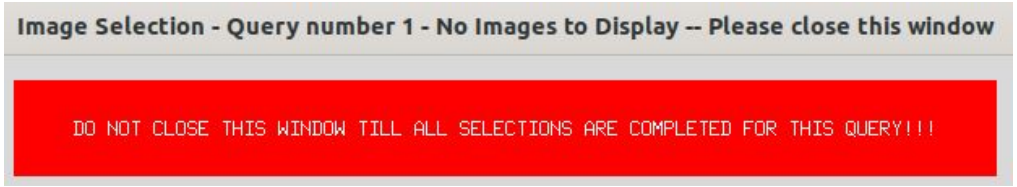
- Database query for this sentences' keywords returned no hits.
  - > Cannot click to proceed to grid selection.
  - > User to simply close window and proceed to selection process for next query results

Find Candidate Images - GUI flow

STAATLICH ANERKANNTE HOCHSCHULE



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



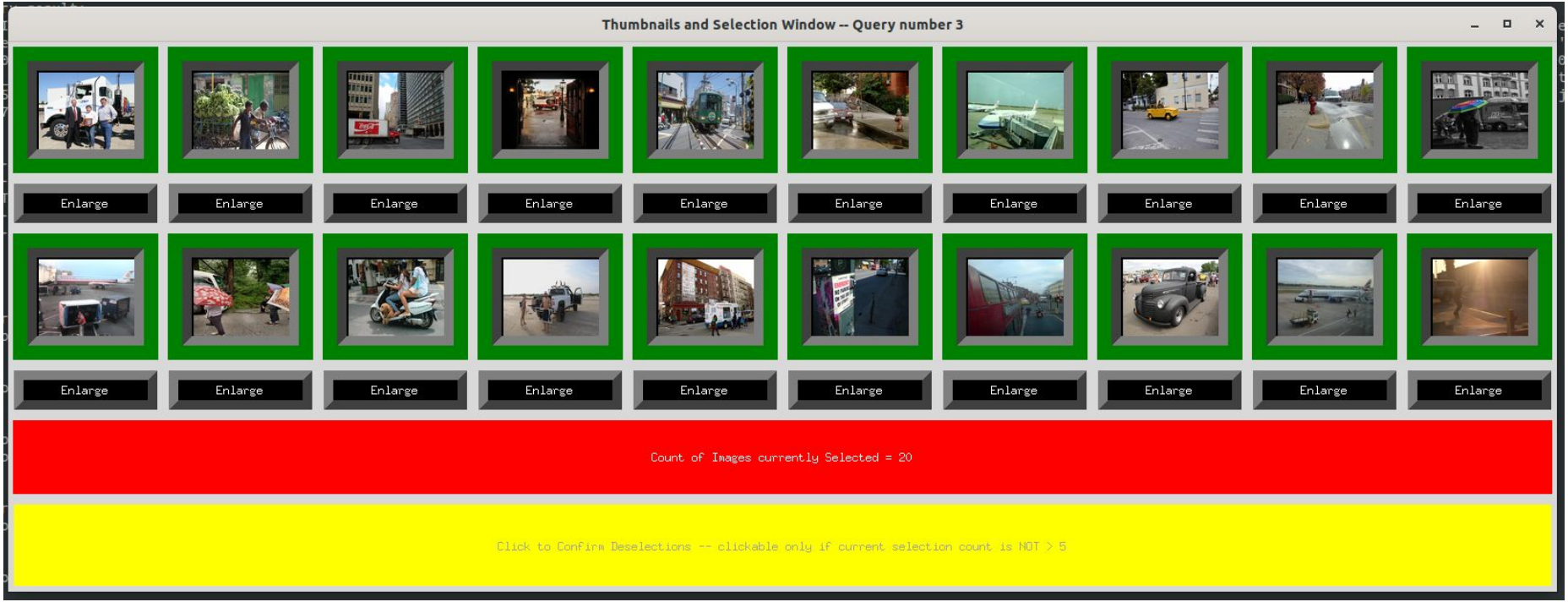
Find Candidate Images - GUI flow

- Third query has 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 the up to 20 images.



Find Candidate Images - GUI flow

- 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

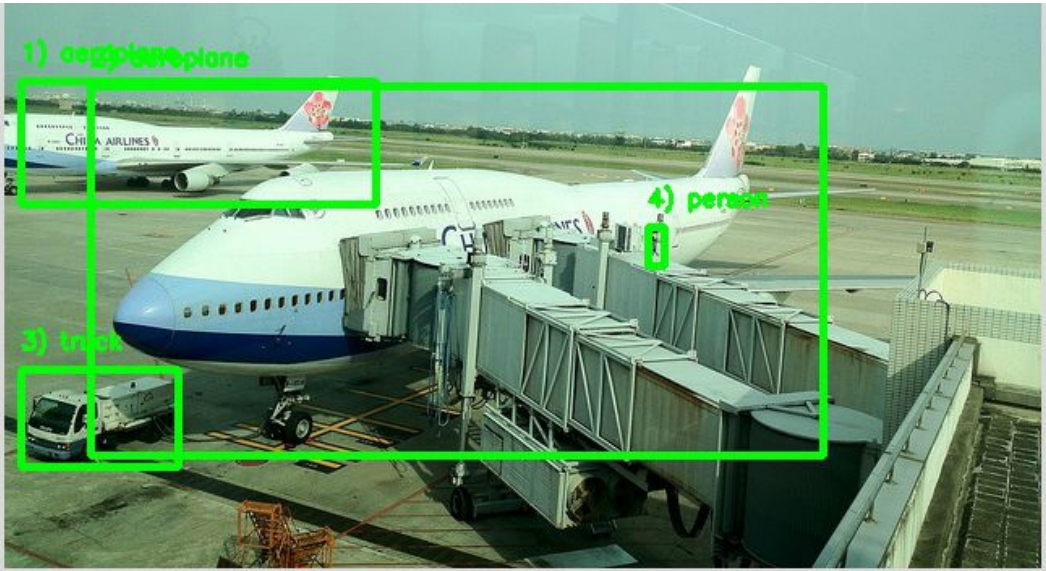
Find Candidate Images - GUI flow

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

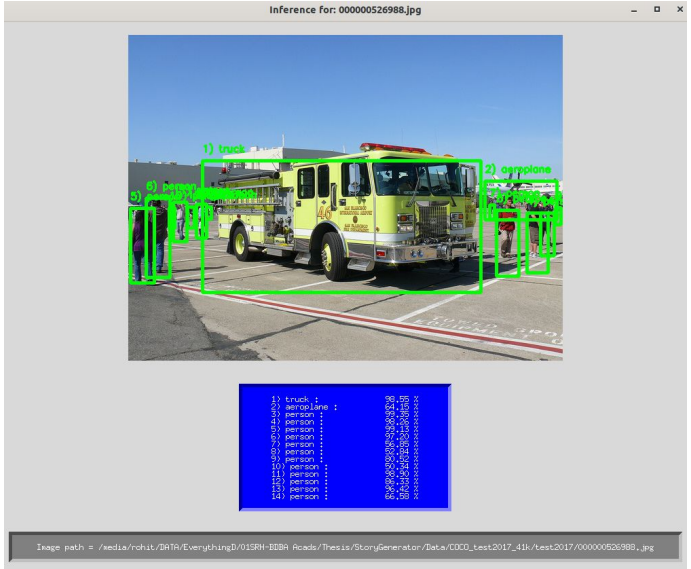
STAATLICH ANERKANNTE HOCHSCHULE

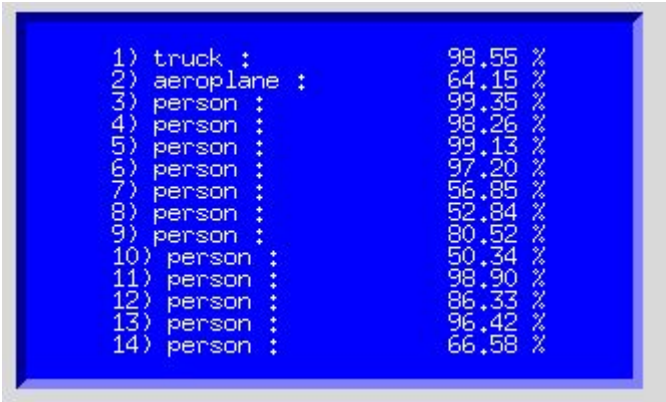


- User clicked button to perform object detection inference and can study output image in new window
- Textual information corresponding to bounding boxes superimposed in image
- User closes window and decides to Deselect image
- **Earlier logic snapshot:**
  - > **Without min. HAS relationship score check. Thus could return image with the min. score of 0.45**
  - > **Note: Truck = 87% and Person = 56%**

Find Candidate Images - GUI flow

STAATLICH
ANERKANNTE
HOCHSCHULE





Updated logic snapshot:

Check for HAS relationship score check > 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

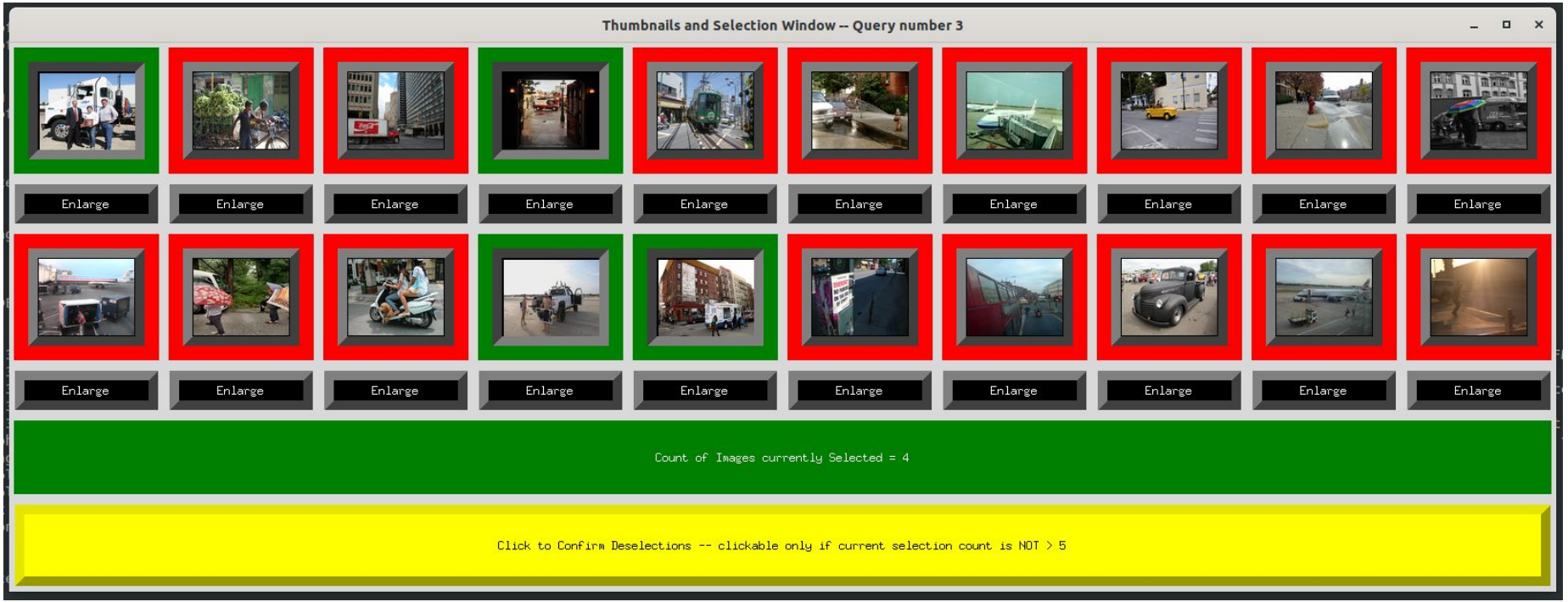
STAATLICH ANERKANNTE HOCHSCHULE

- Example of completely wrong image
- There is no “truck”!
- But model is 93% confident of “truck” object presence.
- Despite setting minimum HAS score = 90% during Neo4j query, such cases will not be tackled.
- **That is why 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:
  - > 16 of 20 images Deselected, 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

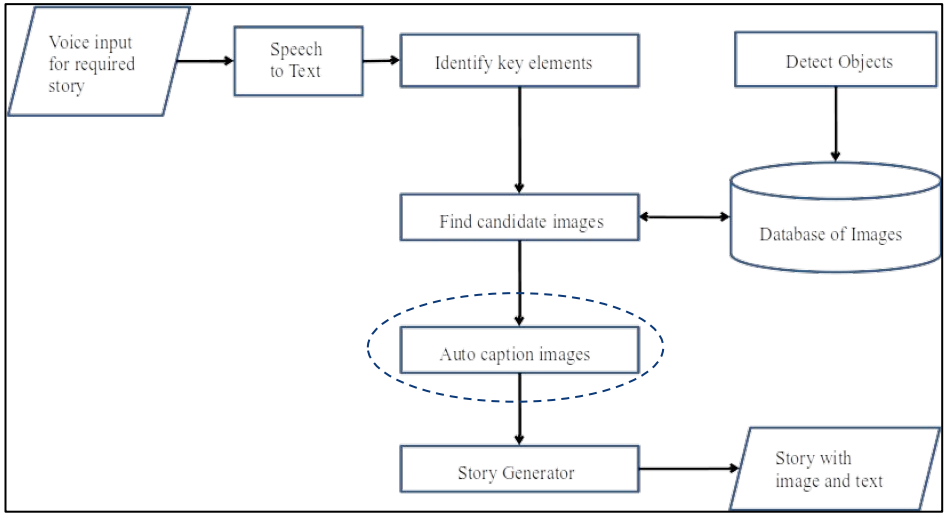


Image Captioning - Introduction

STAATLICH
ANERKANNTE
HOCHSCHULE

- WIP.
-

Image Captioning - Model architecture

STAATLICH ANERKANNTE HOCHSCHULE

- CNN-Encoder: Google Inception v3 pre-trained on Imagenet data (used by Keras built in methods)
- RNN-Decoder: Single layer 256 cell LSTM RNN model - explicitly trained on use case data

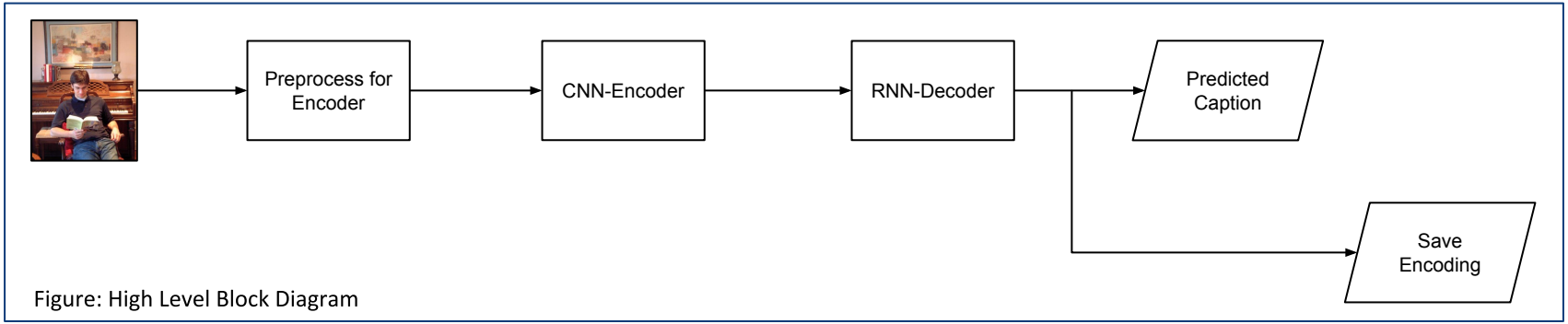


Figure: High Level Block Diagram

- 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: Output 2048 feature vector for each image aka Image encoding. Tapping the last layer before final softmax output.
- Decoder:
  - > After training: Process the Image encodings and output the Predicted caption
  - > During training: Process the image encodings + input descriptions to learn correct predictions

Image Captioning - Model architecture

STAATLICH
ANERKANNTE
HOCHSCHULE

- WIP

Figure: Decoder Detailed Architecture

Image Captioning - Data for model training

- Inputs are combination of Image and the Description:
  - > Image is vector of 2048 values tapped from CNN-Encoder Inception-v3
  - > Description is the cleaned ground truth sentence with insertion of special tokens for start and end

- During preprocessing we:
  - > build the Wordtoix and IxtoWord data structures to map between work 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 point Xt of equal length = MAX_LENGTH_CAPTION

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 11 0 0 0 0 0 0	12
3	1	imgF1	1 11 12 0 0 0 0 0	13
4	1	imgF1	1 11 12 13 0 0 0 0	14
5	1	imgF1	1 11 12 13 14 0 0 0	9
6	2	imgF2	1 0 0 0 0 0 0 0	21
7	2	imgF2	1 21 0 0 0 0 0 0	12
8	2	imgF2	1 21 12 0 0 0 0 0	22
9	2	imgF2	1 21 12 22 0 0 0 0	23
10	2	imgF2	1 21 12 22 23 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 all words
  - > Remove punctuations and special characters (including the period at end of descriptions)
  - > Drop all words with length = 1
    - > done as orphan letters will mess it up. E.g. punctuation removal from “there’s a great” -> “there s a great” -> “there great”
    - > unfortunately legitimate “a” will be dropped also, but no workaround for this
  - > Drop words which are not purely alphabetic. 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 words in the descriptions
  - > Maximum length of caption
  - > Based on chosen threshold for “high frequency words”, cull the full Vocabulary to the one to actually use for model
- Create the data structures for “wordtoix” and “ixtoword” - to convert from string tokens to unique numbers for that token
- Create embeddings matrix using chosen word representation method - Using GloVe-200 here consisting of 400,000 words

Image Captioning - Preprocessing

Example of clean up on the 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_test.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 data = 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 GT 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 occuring > 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)}\t\tlen(img_encodings_val) = {len(img_encodings_val)}")
print(f"Descriptions data:\nlen(descriptions_train) = {len(descriptions_train)}\t\tlen(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 deocder 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

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

[19]:

```
## the value now, as it will be used as:: VOCAB_SIZE = len(wordtoix) + 1
print(f"len(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
```

[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 - Data Used

STAATLICH ANERKANNTE HOCHSCHULE

- MS COCO 2017 has images with 5 descriptions per image.
- Descriptions are used as “training captions”

Original dataset information:

- Each image has 5 descriptions to used as “training captions”

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	No

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	LR	BS
13	1-13	0.001	64
2	14-15	0.001	128
5	16-20	0.0005	32

Training Losses

Epoch	Model 2	Model 3
2	3.4819	3.1985
4	3.2286	3.0032
6	3.1432	2.9315
8	3.0858	2.8673
10	3.0448	2.8443
12	-	2.8397
14	-	2.8013
16	-	2.7977
18	-	
20	-	

Training Parameters (both):

Optimizer = Adam

Loss function = Categorical cross-entropy

Key	
Ep	Epoch
LR	Learning Rate
BS	Batch Size

Image Captioning - Model Evaluation

STAATLICH ANERKANNTE HOCHSCHULE

- Used built-in Bleu scorer of NLTK calculated in Python as:
  - > import nltk.translate.bleu_score as nltk_bleu
  - > Bleu score = {nltk_bleu.sentence_bleu([list of GT descriptions], “the predicted caption from model”)
- Below are some random lowest and highest scores from both models

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 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 Evaluation

STAATLICH ANERKANNTE HOCHSCHULE

- Computed on all the 5k Test dataset images: Both models - comparison after 10 Epochs
- Average value for Model 3 very slightly higher.
- Std. Dev. for Model 3 slightly larger
- Overall Model 3 is slightly better than Model 2 - continued training Model 3 for more epochs

Bleu Scores	Model 2	Model 3
Maximum	1.0	1.0
Minimum	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						
	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 - Scope for improvements

STAATLICH
ANERKANNTE
HOCHSCHULE

- Will try to implement these if time permits:
  - > Currently it is Greedy search aka Max. Likelihood search
  - > Train a new model with different hyperparameters during training - if possible as time is short
  - > Present the Image and its predicted caption using GUI interface like in earlier stages
    - > Possibly then allow user to edit the caption for any corrections

Changes to Expose

STAATLICH
ANERKANNTE
HOCHSCHULE

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

STAATLICH
ANERKANNTE
HOCHSCHULE

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

STAATLICH
ANERKANNTE
HOCHSCHULE

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

STAATLICH
ANERKANNTE
HOCHSCHULE

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 guage 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

STAATLICH
ANERKANNTE
HOCHSCHULE

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. Image Captioning with Keras, https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8 as on 10.09.2020
11. Automatic Image Captioning, https://gist.github.com/nttuan8/a621aa6700995db6db71b0a768a8552f as on 20.09.2020