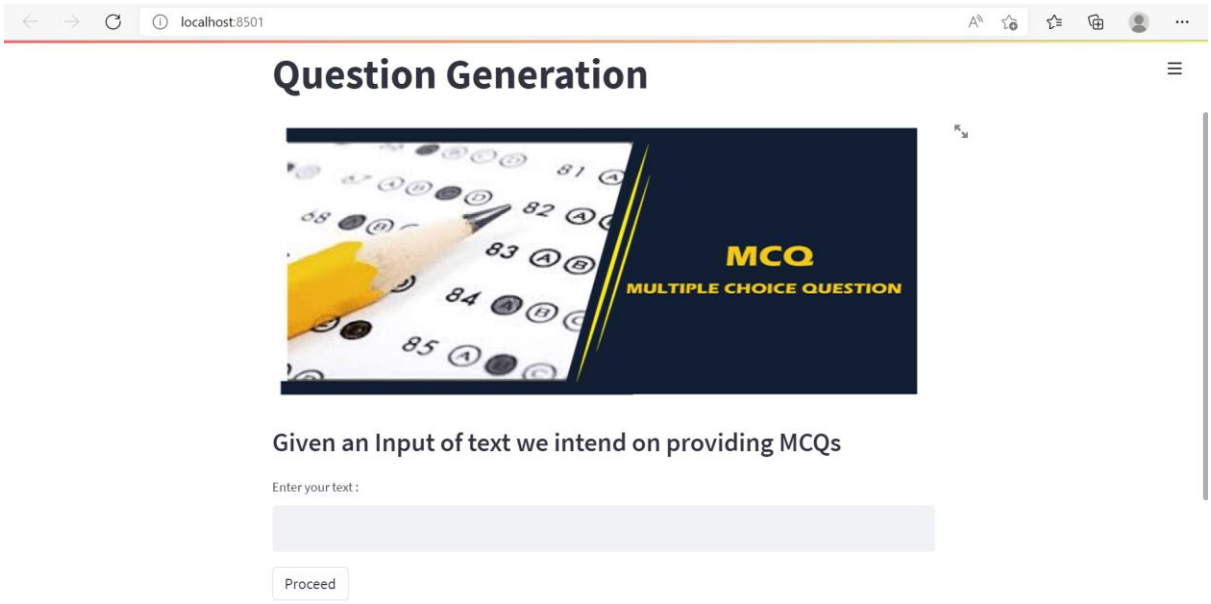
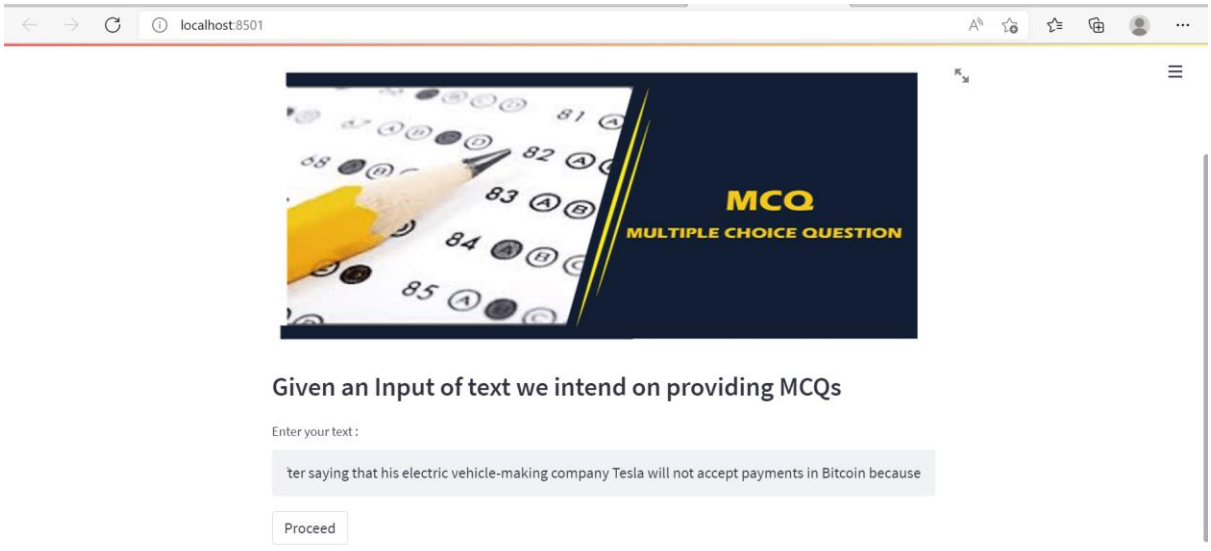
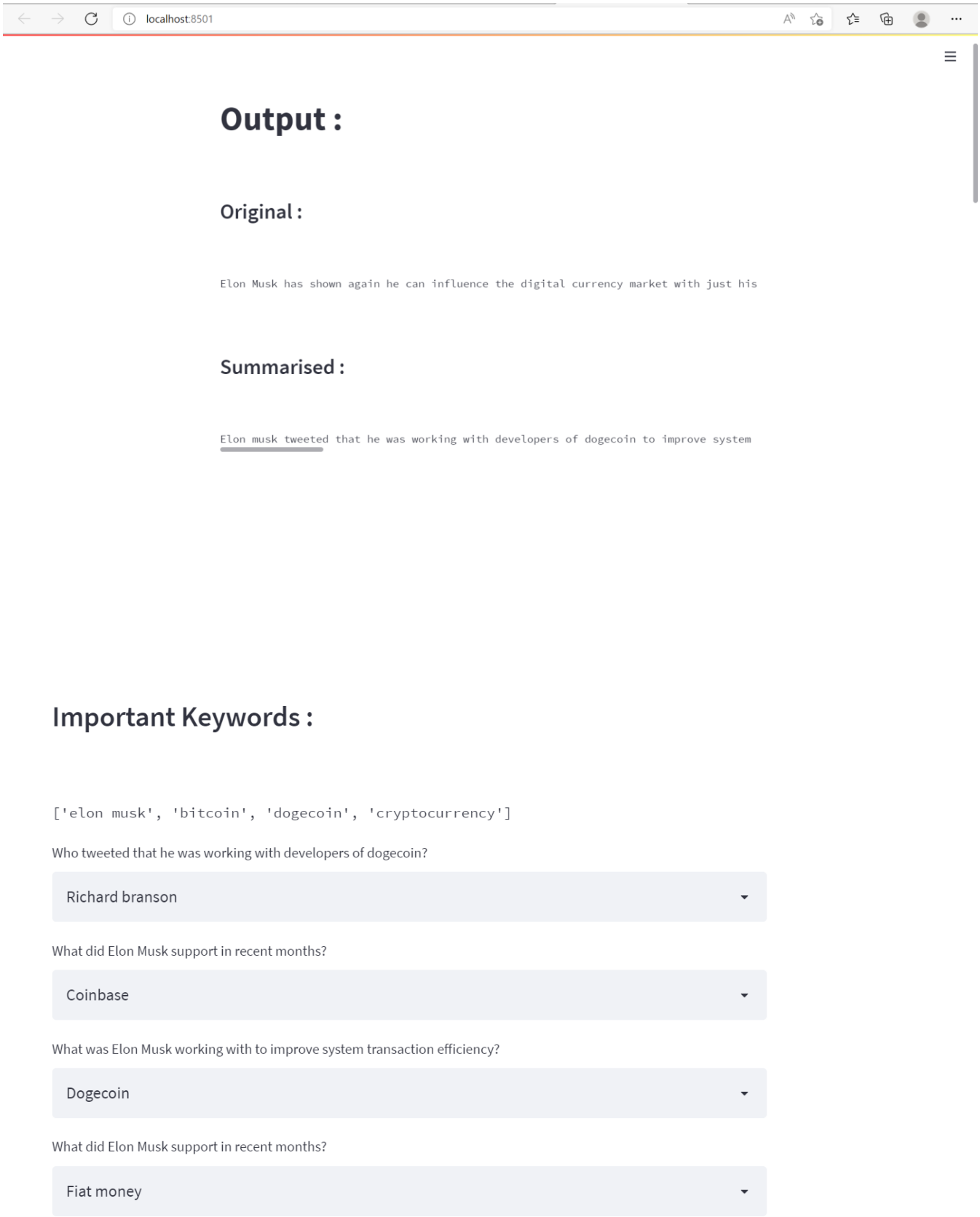
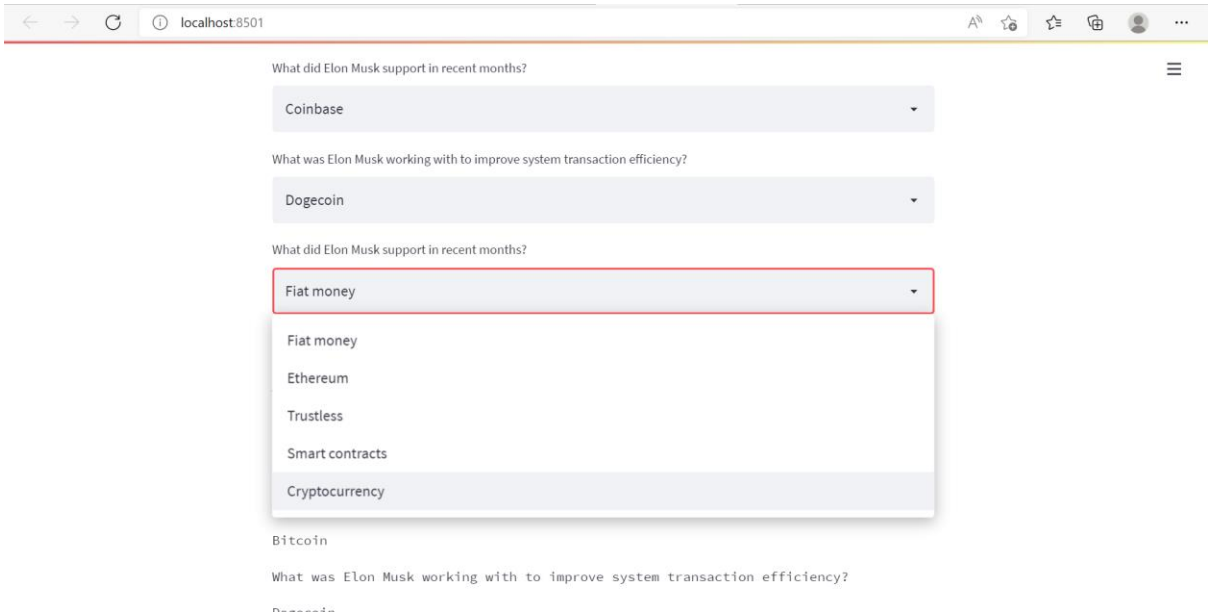
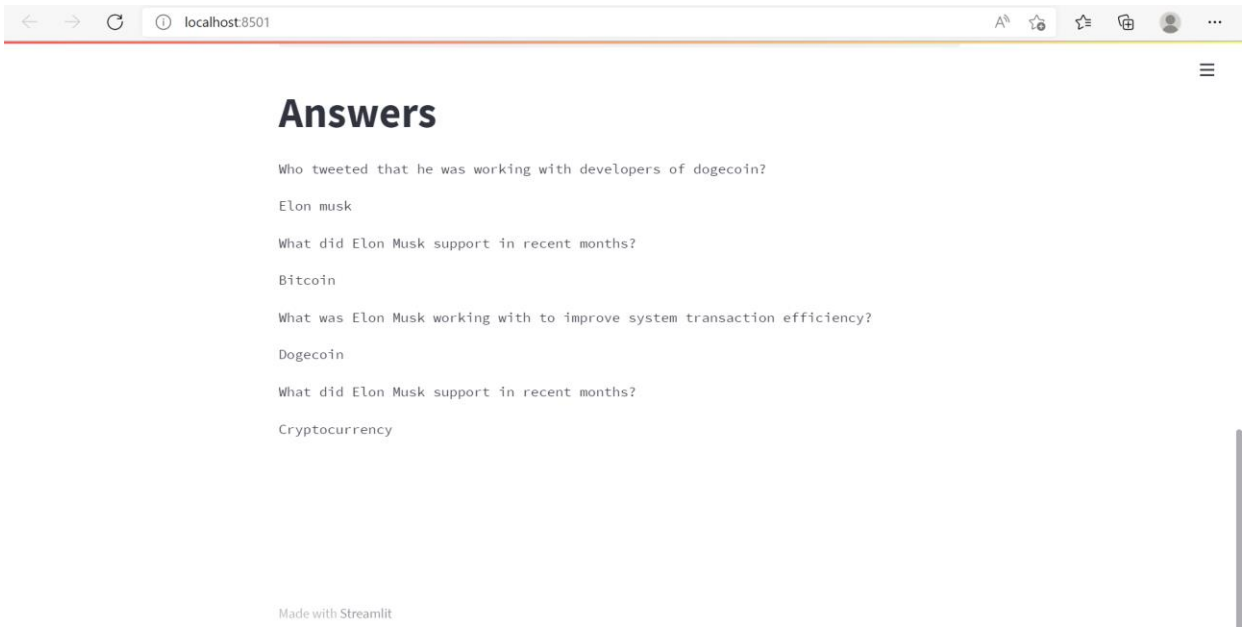
## Working Example :



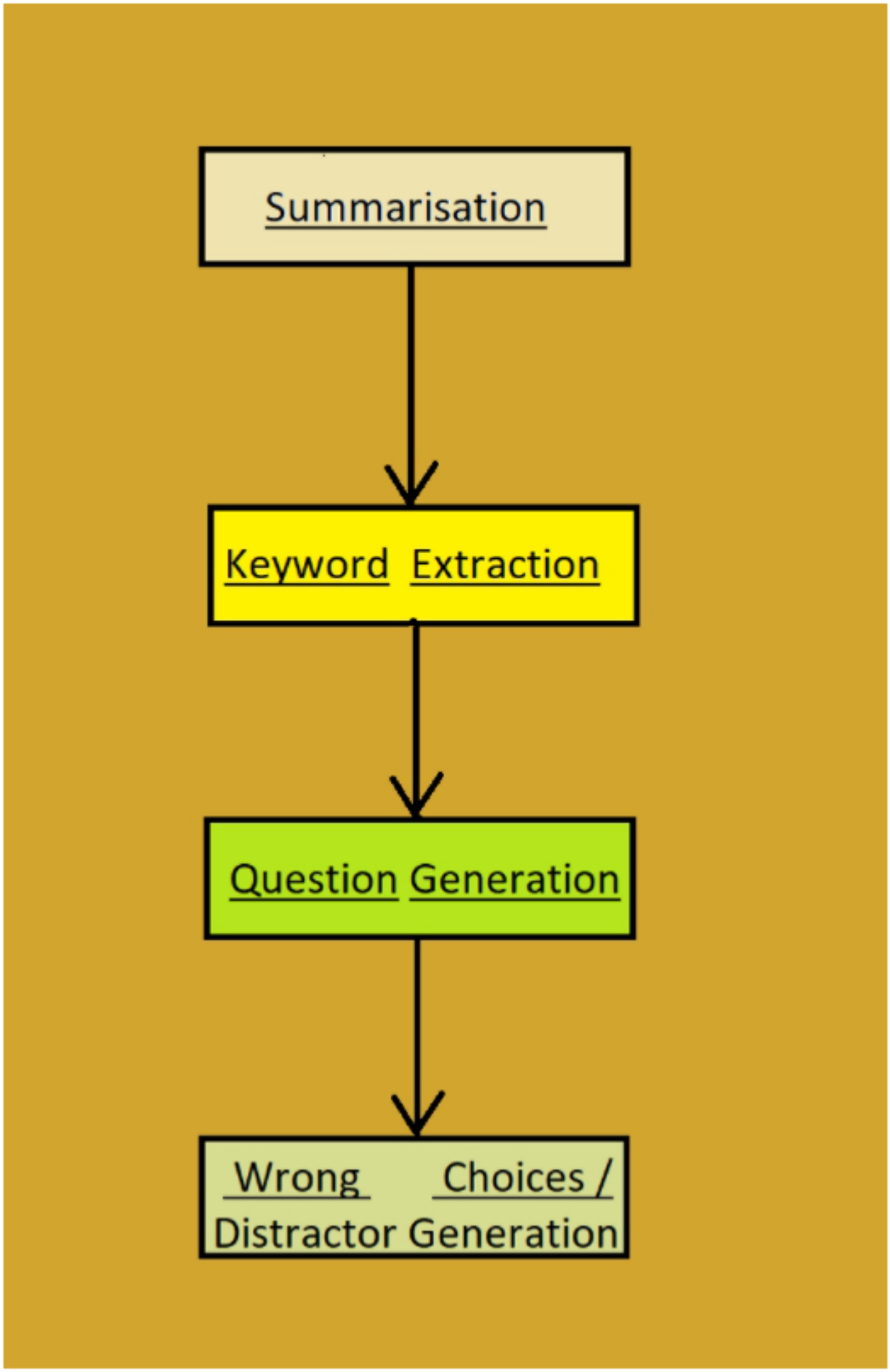

localhost:8501
Output :
Original :
Elon Musk has shown again he can influence the digital currency market with just his
Summarised :
Elon musk tweeted that he was working with developers of dogecoin to improve system
Important Keywords :
['elon musk', 'bitcoin', 'dogecoin', 'cryptocurrency']
Who tweeted that he was working with developers of dogecoin?
Richard branson
What did Elon Musk support in recent months?
Coinbase
What was Elon Musk working with to improve system transaction efficiency?
Dogecoin
What did Elon Musk support in recent months?
Fiat money


localhost:8501
What did Elon Musk support in recent months?
Coinbase
What was Elon Musk working with to improve system transaction efficiency?
Dogecoin
What did Elon Musk support in recent months?
Fiat money
Fiat money
Ethereum
Trustless
Smart contracts
Cryptocurrency
Bitcoin
What was Elon Musk working with to improve system transaction efficiency?


localhost:8501
Answers
Who tweeted that he was working with developers of dogecoin?
Elon musk
What did Elon Musk support in recent months?
Bitcoin
What was Elon Musk working with to improve system transaction efficiency?
Dogecoin
What did Elon Musk support in recent months?
Cryptocurrency
Made with Streamlit

# How we will proceed :

Workflow of MCQ generation-



# Summarisation & Question Generation

## Code for Summarisation :

We've imported the pre trained model

```
from transformers import T5ForConditionalGeneration,T5Tokenizer
summary_model = T5ForConditionalGeneration.from_pretrained('t5-base')
summary_tokenizer = T5Tokenizer.from_pretrained('t5-base')
```

Pre-trained model

Using it as a summarizer

```
def summarizer(text,model,tokenizer):
  text = text.strip().replace("\n"," ")
  text = "summarize: "+text
  # print (text)
  max_len = 512
  encoding = tokenizer.encode_plus(text,max_length=max_len, pad_to_max_length=False,truncation=True, return_tensors="pt")

  input_ids, attention_mask = encoding["input_ids"], encoding["attention_mask"]

  outs = model.generate(input_ids=input_ids,
                                attention_mask=attention_mask,
                                early_stopping=True,
                                num_beams=3,
                                num_return_sequences=1,
                                no_repeat_ngram_size=2,
                                min_length = 75,
                                max_length=300)


  dec = [tokenizer.decode(ids,skip_special_tokens=True) for ids in outs]
  summary = dec[0]
  summary = postprocesstext(summary)
  summary= summary.strip()
```

function to summarize

Proceed2

# T5 Model

So the T5 is a pre trained model based on transfer learning and is trained and created by the google team.

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text forma

It is useful for summarisation.

Encoders like BERT are used mainly for making a single prediction for an entire sequence,

While decoders are capable of taking input and generating text.

On the other hand  t5 utilises of both encoders and decoders, while also using sequence to sequence training.

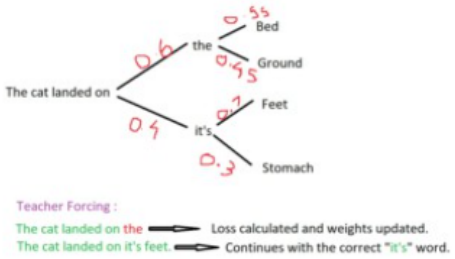
# How it is trained

## Denoising Objective :

Few words are masked with the goal of predicting masked tokens

It uses greedy decoding :

i.e selecting word with highest probability every time




Teacher Forcing :

The cat landed on the ⟸ Loss calculated and weights updated.

The cat landed on it's feet. ⟸ Continues with the correct "it's" word.

How it works.

# Teacher Forcing

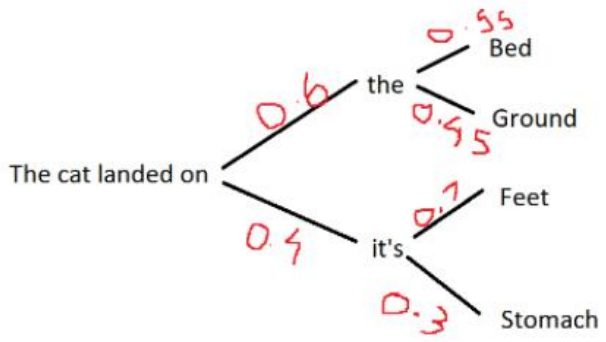
The target word is passed as next input to decoder.

If it doesn't generate the ground truth :

- The weights are updated accordingly

- The correct word is put there and it continues.

T5 utilises cross entropy loss.




Teacher Forcing :

The cat landed on the ⟹ Loss calculated and weights updated.

The cat landed on it's feet. ⟹ Continues with the correct "it's" word.

How it works.

Proceed3

## Training on SQUAD dataset

```
args_dict = dict(
    batch_size=4,
)

args = argparse.Namespace(**args_dict)


model = T5FineTuner(args,t5_model,t5_tokenizer)

trainer = pl.Trainer(max_epochs = 1, gpus=1,progress_bar_refresh_rate=30)

trainer.fit(model)

print ("Saving model")
save_path_model = '/content/gdrive/My Drive/t5/model/'
save_path_tokenizer = '/content/gdrive/My Drive/t5/tokenizer/'
model.model.save_pretrained(save_path_model)
t5_tokenizer.save_pretrained(save_path_tokenizer)
```


```
args_dict = dict(
    batch_size=4,
)

args = argparse.Namespace(**args_dict)


model = T5FineTuner(args,t5_model,t5_tokenizer)

trainer = pl.Trainer(max_epochs = 1, gpus=1,progress_bar_refresh_rate=30)

trainer.fit(model)

print ("Saving model")
save_path_model = '/content/gdrive/My Drive/t5/model/'
save_path_tokenizer = '/content/gdrive/My Drive/t5/tokenizer/'
model.model.save_pretrained(save_path_model)
t5_tokenizer.save_pretrained(save_path_tokenizer)
```

## Question Generator and loading the model

```
question_model = T5ForConditionalGeneration.from_pretrained('C:/Users/Usha Gautam/Desktop/Sem 6/NLP project/t5/model/')
question_tokenizer = T5Tokenizer.from_pretrained('C:/Users/Usha Gautam/Desktop/Sem 6/NLP project/t5/tokenizer/')
question_model = question_model.to(device)
```

```
def get_question(context,answer,model,tokenizer):
  text = "context: {} answer: {}".format(context,answer)
  encoding = tokenizer.encode_plus(text,max_length=384, pad_to_max_length=False,truncation=True, return_tensors="pt").to
  input_ids, attention_mask = encoding["input_ids"], encoding["attention_mask"]

  outs = model.generate(input_ids=input_ids,
                                  attention_mask=attention_mask,
                                  early_stopping=True,
                                  num_beams=5,
                                  num_return_sequences=1,
                                  no_repeat_ngram_size=2,
                                  max_length=72)


  dec = [tokenizer.decode(ids,skip_special_tokens=True) for ids in outs]


  Question = dec[0].replace("question:","")
  Question= Question.strip()
  return Question

```

Model

Proceed4

# Keyword Extraction & Generating Distractors

## Keyword Extraction :

We use the pke library with unsupervised.MultipartiteRank()
to find the keywords in the paragraph.

It uses topicRank to find the important keywords

MMR

The idea behind using MMR is that it tries to reduce redundancy and
increase diversity in the result

MMR selects the phrase in the final keyphrases list according
to a combined criterion of query relevance and novelty of information.

The latter measures the degree of dissimilarity between the document being
considered and previously selected ones already in the ranked list.

# Generating Distractors :

Distractors :

Can't be too similar or different.

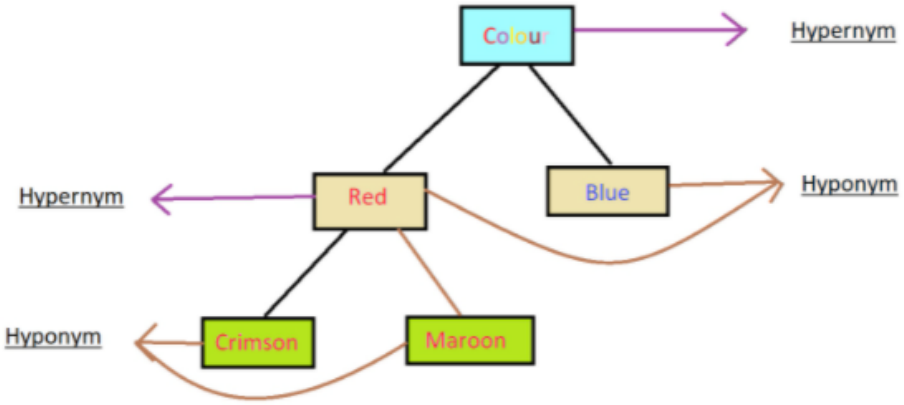
Must be mutually exclusive i.e no Synonyms

Wordnet :

It is a large lexical database of English

Captures Broader Relationship among words.

Labels semantic relations among words.




Examples od Hypernym and Hyponym

Sense2Vec :

Unlike word vectors which are context independent
they capture contextual information.

```
!python -m wget https://github.com/explosion/sense2vec/releases/download/v1.0.0/s2v_reddit_2015_md.tar.gz
```

```
Saved under s2v_reddit_2015_md.tar.gz
```

```
!tar -xvf  s2v_reddit_2015_md.tar.gz
```

```
x ./._s2v_old
x ./s2v_old/
x ./s2v_old/._freqs.json
x ./s2v_old/freqs.json
x ./s2v_old/._vectors
x ./s2v_old/vectors
x ./s2v_old/._cfg
x ./s2v_old/cfg
x ./s2v_old/._strings.json
x ./s2v_old/strings.json
x ./s2v_old/._key2row
x ./s2v_old/key2row
```


```
!tar -xvf  s2v_reddit_2015_md.tar.gz
```

```
x ./._s2v_old
x ./s2v_old/
x ./s2v_old/._freqs.json
x ./s2v_old/freqs.json
x ./s2v_old/._vectors
x ./s2v_old/vectors
x ./s2v_old/._cfg
x ./s2v_old/cfg
x ./s2v_old/._strings.json
x ./s2v_old/strings.json
x ./s2v_old/._key2row
x ./s2v_old/key2row
```


```
%ls s2v_old
```

```
 Volume in drive C is Windows
 Volume Serial Number is 32D0-0110

 Directory of C:\Users\Usha Gautam\Desktop\Sem 6\NLP project\s2v_old

28-09-2019  16:26    <DIR>          .
28-09-2019  16:26    <DIR>          ..
18-11-2019  04:04               174 ._cfg
28-09-2019  16:26               174 ._freqs.json
28-09-2019  16:26               174 ._key2row
28-09-2019  16:26               174 ._strings.json
28-09-2019  16:26               174 ._vectors
18-11-2019  04:04               424 cfg
28-09-2019  16:26        49,969,681 freqs.json
28-09-2019  16:26        16,492,891 key2row
28-09-2019  16:26        26,188,439 strings.json
28-09-2019  16:26       611,973,760 vectors
              10 File(s)    704,626,065 bytes
               2 Dir(s)  137,899,466,752 bytes free
```

Proceed5