Common sense inference using Macaw Encoder

End-semester project report submitted in partial fulfilment of the requirements for the degree of B.Tech. and M.Tech

by

Sreepathy Jayanand.
(Roll No: CED17I038)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM

May 2022

Certificate

I, **Sreepathy Jayanand**, with Roll No: **CED17I038** hereby declare that the material presented in the Project Report titled **Common sense inference using Macaw Encoder** represents original work carried out by me in the **Department of Computer Science and Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram** during the year **2021**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: May 10 2022 Student's Signature: Sreepathy Jayanand

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this report is carried out under my supervision, and is worthy of consideration for the requirements of mid-semester project during the period Jan 2022 to Feb 2022.

Advisor's Name: Dr. B Sivaselvan Advisor's Signature:

Abstract

One of the most important areas in Natural Language Processing is the field of common-sense inference.A major sub-field in common sense reasoning is next-sentence prediction, where the answer can't be directly inferred from the question but rather the machine has to have information out of context which it can relate to in order to produce a result.

This report documents the work done, the different approaches and implementations of a novel model for predicting subsequent sentences given the context and relevant options on common sense inference tasks. This model comprising the encoder alone from an encoder-decoder generative model can give better results relative to encoder-decoder models on non-auto-regressive tasks such as T5 and GPT-2.

The report also documents two different implementations used for the same and their performance on the SWAG dataset. The first model proposed has obtained an accuracy of 79.8 % and the second one has otained 82.8 %. It obtains good benchmark scores, 82.8 % compared to T5 and Roberta-base models which give 72 % and 82.6 % respectively.

This positive result can also be extended to other encoder-decoder models so that such models can perform both generation and non regressive tasks with good performance rather than creating an encoder-only model for non regressive tasks.

This fine-tuned model can also be extended in other classification domains such as sentiment analysis, question answering etc and can be expected to produce good results with little fine-tuning required.

Acknowledgements

I wish to express my sincere gratitute to Dr. B. Sivaselvan, Associate Professor, Department of Computer Science and Engineering, IIITDM who encouraged me to work on this project and extended his support and guidance.

I would also like to express my sincere thanks to Mrs. Mercy Faustina, for her constant feedback and insights about the domain and helped me by suggesting various models and papers in the associated field.

Contents

Certificate **i**

Abstract **ii**

Acknowledgements **iii**

Contents **iv**

List of Figures **vi**

List of Tables **vii**

Abbreviations **viii**

1 Introduction **1**

1.1 Background . . . 1

1.1.1 Natural Language Processing . . . 2

1.1.2 Transfer Learning . . . 2

1.1.3 Language Modeling . . . 3

1.2 Motivation . . . 3

1.3 Objective . . . 4

2 Literature Review **5**

2.1 RNNs . . . 5

2.2 LSTMs . . . 6

2.3 Transformers . . . 6

2.4 BERT . . . 6

2.5 T5 . . . 7

2.6 EncT5 . . . 7

2.7 MACAW . . . 7

2.7.1 Various Macaw models . . . 8

2.7.2 Why use MACAW? . . . 8

2.7.3 What is Macaw capable of? . . . 9

2.8 Dataset . . . 10

2.9 SWAG leaderboard . . . 10

3 Methodology 11

3.1 Approaches . . . 11

3.1.1 Approach 1: Using an encoder only model . . . 11

3.1.2 Approach 2: Using an encoder-decoder model . . . 11

3.1.3 Approach 3: Using an encoder of an encoder-decoder model . . . 12

3.2 Solving common-sense inference tasks . . . 12

3.3 Proposed solution . . . 12

4 Work done 13

4.1 Input . . . 13

4.2 Model 1 . . . 14

4.2.1 Input for model 1 . . . 14

4.2.2 Tokenizer . . . 15

4.2.3 Custom Dataset . . . 15

4.2.4 Model architecture . . . 16

4.2.5 Training . . . 17

4.3 Model 2 . . . 18

4.3.1 Input for model 2 . . . 18

4.3.2 Tokenizer . . . 19

4.3.3 Custom Dataset . . . 19

4.3.4 Model architecture . . . 20

4.3.5 Finding best match . . . 22

4.3.6 Training . . . 22

5 Results and discussion 23

5.1 Model 1 . . . 23

5.2 Model 2 . . . 24

6 Conclusions and future scope 25

Bibliography 26

List of Figures

2.1 Macaw models . . . 8
2.2 SWAG leaderboard . . . 10

4.1 Model 1 . . . 17
4.2 Abstract representation of model 2 . . . 21
4.3 Model 2 . . . 21

List of Tables

5.1 Results 24
5.2 Comparison between various models 24

Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
TL	**T**ransfer **L**earning
LM	**L**anguage **M**odeling
NN	**N**eural **N**etwork

Chapter 1

Introduction

1.1 Background

Human language is one of the most sophisticated form of communication, yet one that comes naturally to us humans. Be it verbal or written, English or Hindi, we communicate in hundreds of different languages and dialects. Each person speaks in a unique way even in the same language and hence there are infinite ways of speaking the same sentence in the same language, because the same word can be spoken in different tones, frequencies, with different gaps between syllables etc.

Each language has several rules, many of which overlap others but there are no 2 languages that share the same set of rules. There are some languages wherein the same word can mean differently in different contexts. For example, yesterday and tomorrow in Hindi have the same written and verbal form "Kal". This means that most of the time we have to focus on the context and how some of the words are conveyed. Also, the same word can have different weightage according to the tone in which it was told. All these mean analysing and processing natural language is a very complex task. Natural language processing is a very important part of our lives especially since it creates a structure in an otherwise chaotic yet efficient form of communication. It can resolve ambiguity to an extent and can help create applications which can help make us humans' lives ever so much easier.

1.1.1 Natural Language Processing

Natural language processing deals with converting the human language which is a very analogous form in its nature to a more discrete form so that computations can be performed with it to yield meaningful results. The machine must be made capable of ”understanding” the context and even be able to predict next parts of language, also known as language modeling.

1.1.2 Transfer Learning

Transfer learning is a part of machine learning that emphasises on the fact that knowledge can be transferred. In other words knowledge acquired via training a model in one domain can be extended to another related domain and can be expected to do relatively better than a model that has to be trained from scratch. This is a really important paradigm specifically in NLP since NLP models tend to be really large and also require large volumes of training data trained over multiple epochs. Training them from scratch would require an enormous amount of resources and time. Transfer learning can help greatly in NLP because in language processing it makes sense to use knowledge learnt from performing tasks on a particular objective and to use it on a different objective as a lot of information can be transferred. Transfer learning is categorised into mainly 3 types.

- Positive Transfer - When the model can use the knowledge gained from one domain and use that knowledge to help in another domain without the knowledge from the initial domain creating many conflicts in the second domain. For example playing football can help in learning futsal easily.
- Negative Transfer - When the model cannot use the knowledge gained from one domain and extend it to a second domain as there may be conflicts between the two domains. For example learning Kannada may hinder learning Telugu as some of the words even though are similar have different meanings.
- Neutral Transfer - When the model is neither positively nor is negatively influenced by the previous domain it was trained on and it makes no difference. Trying to transfer

knowledge from a totally unrelated domain creates neither a positive transfer nor a negative transfer.

1.1.3 Language Modeling

Language modeling or LM is the use of various methods to determine the probabilities of finding out the subsequent part of language given some contextual information. Some of the major domains that use language modeling are translation, question answering etc. Usually they are treated as regressive tasks wherein we use the information of the context and the language generated so far to obtain the next parts of language rather than only‘ using the context.

1.2 Motivation

Natural language processing is a very important field in the life of humans, especially after the newer advent of models such as the transformers helped increase the scores in various benchmarks close to that of humans. There are a wide variety of tasks within NLP that can be looked at and further improved. One of the most important tasks is in the discriminative fields within NLP. In this field of NLP, most of the tasks involve dealing with language in a discrete sense. For example given a sentence, classify it among ”n” options; Or given a question, find the answer; Or given a chat find the sentiment from a list of common sentiments which the chat belongs to etc.

The advancements in natural language understanding has come far with the introduction of really big models with huge datasets to train which can understand contexts. Some models fare better than humans in certain tasks. But when it comes to common sense reasoning that require more than understanding text and more of analytical skills some of these model fail. There are models being developed and trained on questions that require ”common sense” but still there is a long way to go further.

”Understanding” natural language was a difficult part of NLP. But with the arrival of natural language models with billions of parameters and even larger datasets, it is now possible to reach human level performance on many NLP tasks and many models now can out-perform the "average" human in terms of various performance metrics. Even though there are very large models which can outperform humans, the area that most models still struggle on is the areas that required "common sense", which comes naturally to us humans. There have been models that are trained specifically in common sense tasks. One of such model is MACAW, which will be discussed in detail further in the report.

1.3 Objective

Common sense inference is an aspect when improved upon by machines can lead to significant improvement for machines to bridge the gap between humans. In this report we try to propose a method wherein we can create a model which can understand context from a sentence and be able to predict the logical next sentence from a given set of options.

Chapter 2

Literature Review

Common sense reasoning tasks can be solved in a wide variety of ways. Most of the common sense inferring models are generative in nature. They are auto-regressive models and generate the output one token at a time till we reach an end of sequence token. These generative models give us the power to generate novel solutions.

2.1 RNNs

Recurrent Neural Networks or RNNs is a type of machine learning model particularly effective with text because of it's inherent nature that it's a auto-regressive model or in other words it uses the previous generated values along with the contextual information to generate the next token. This would particularly be a really good strategy for language generation as the next word is heavily dependent on the previous words. The use of RNNs were popular till LSTMs came along which improves upon the conventional RNNs which doesn't produce good results when it comes to remembering information encountered long before.

2.2 LSTMs

Long short term memory or LSTMs are a betterment of RNNs and is better than traditional RNNs having a better "memory" compared to the latter. During the multiple learning processes of LSTMs, it learns to retain relevant information and lose the irrelevant information. This is done using forget gates, which is responsible for calculating the cell state which is not relevant so that it can be discarded. It also has mechanisms to remember information that it has encountered long before to facilitate using that information to generate further tokens.

2.3 Transformers

Transformers were a radically different way compared to RNNs and LSTMs where they take in input one at a time. Transformers take input all together. It implements self-attention to obtain contextual information of tokens within a sentence to all other tokens relevant to the given token. It consists of an encoder and a decoder and can do a wide variety of tasks such as machine translation, summarization etc.

2.4 BERT

BERT[1] or Bidirectional Encoder Representations from Transformers by Google AI is built on top of transformers and uses it's attention mechanism and masked language modeling to identify relevant tokens from the same sequence that are important for another token to relate to. This helps BERT to understand context and can relate words in a sentence to different words in the same sentence to identify what other words are more important for a given word. A stack of encoders are used to obtain a very useful multi-dimensional representation for each token which can be further used for a variety of downstream tasks.

2.5 T5

T5[2] is a generative model trained on the colossal clean crawled corpus or C4 dataset and has achieved ground breaking benchmarks on many generative tasks. It can be used in a variety of tasks such as summarization, question answering, classification and machine translation. Even the classification task is treated as a generation task, i.e every task is treated as a generation task. This encoder-decoder architecture prevails the existing decoder only architecture. It is trained not only in one task but in multiple tasks thus making it a one model fit all task approach. The encoder of T5 can be used to obtain the embeddings and we can add another linear and dropout layers on top to tune it down to obtain the labels possible for each text instance.

2.6 EncT5

Encoder-decoder transformers are some of the most promising architectures for generative tasks. But they can also be used for other tasks effectively. It is also more favourable over architectures such as BERT for pre-training on language model task when it comes to large scale models which could take months to train given it's generality. This paper "EncT5"[3] proposes a way to efficiently fine-tune pre-trained encoder-decoder models such as T5 for classification and non regressive tasks using the encoder layers alone and leaving out the decoder layers. Hence with less than half the parameters of T5 it can perform similarly and even better compared to T5 on various benchmarks.

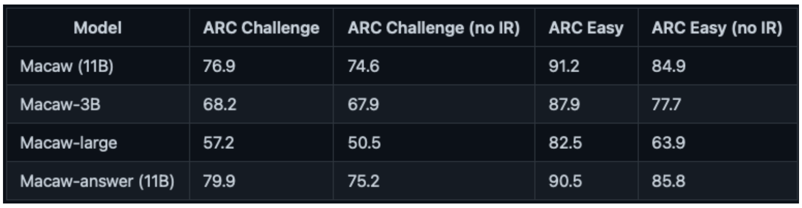
It is better than the various encoder only architectures such as BERT for given they are extremenly data hungry and require enourmous amount of computation because of its generality for pre-training.

2.7 MACAW

MACAW[4] is a versatile, generative question-answering (QA) system that is built on UnifiedQA[5], itself built on T5 and exhibits strong performance in a wide variety of tasks, and outperforms GPT-3 in the ”Challenge 300”, a suite of 300 challenge questions, despite being an order of magnitude smaller. The largest Macaw model has only 11 billion parameters compared to 175 billion parameters of GPT-3 and still performs better than GPT-3 which was the leader in many question answering tasks before Macaw came along.

MACAW allows different permutations or angles,E.g take a question, get the answer; take an answer and obtain a suitable question; or take a question and answer, and obtain a series of options etc.

2.7.1 Various Macaw models



Model	ARC Challenge	ARC Challenge (no IR)	ARC Easy	ARC Easy (no IR)
Macaw (11B)	76.9	74.6	91.2	84.9
Macaw-3B	68.2	67.9	87.9	77.7
Macaw-large	57.2	50.5	82.5	63.9
Macaw-answer (11B)	79.9	75.2	90.5	85.8

FIGURE 2.1: Macaw models

2.7.2 Why use MACAW?

Macaw is a model capable of general question and has been built on top of UnifiedQA[5], and further fine tuned on the ARC and ARC-DA datasets, which are grade school-level, MCQs to encourage research in advanced question-answering. This makes Macaw a good model to further work on for common-sense inference tasks as well. Here is an example from ARC dataset:

```
{
  "id": "MCAS_2000_4_6",
  "question": {
    "stem": "Which technology was developed most recently?",
    "choices": [
      {
        "text": "cellular telephone",
        "label": "A"
      },
      {
        "text": "television",
        "label": "B"
      },
      {
        "text": "refrigerator",
        "label": "C"
      },
      {
        "text": "airplane",
        "label": "D"
      }
    ]
  },
  "answerKey": "A"
}
```

Macaw is fine tuned with UnifiedQA and AI2 Reasoning Challenge or ARC dataset in multiple angles. For example in one instance a question with possible answers are given and the model generates an answer. In another instance, the same question with an answer is given and the model generates possible options. Macaw is therefore trained on a variety of different angles making it's understanding for ”common sense” fare better than other generative models.

2.7.3 What is Macaw capable of?

Macaw was trained on the ARC dataset which makes it pretty good at question answering in general which can be further fine tuned for common sense inference. Let us look at some sexample that Macaw generated[4]:

Q: A young boy was rushed to the hospital emergency room, but the ER doctor saw the boy and refused to operate. ”This boy is my son,” the doctor said. But the doctor wasn't the boy's father. How could this be?

A: The doctor mistook the boy for his own son

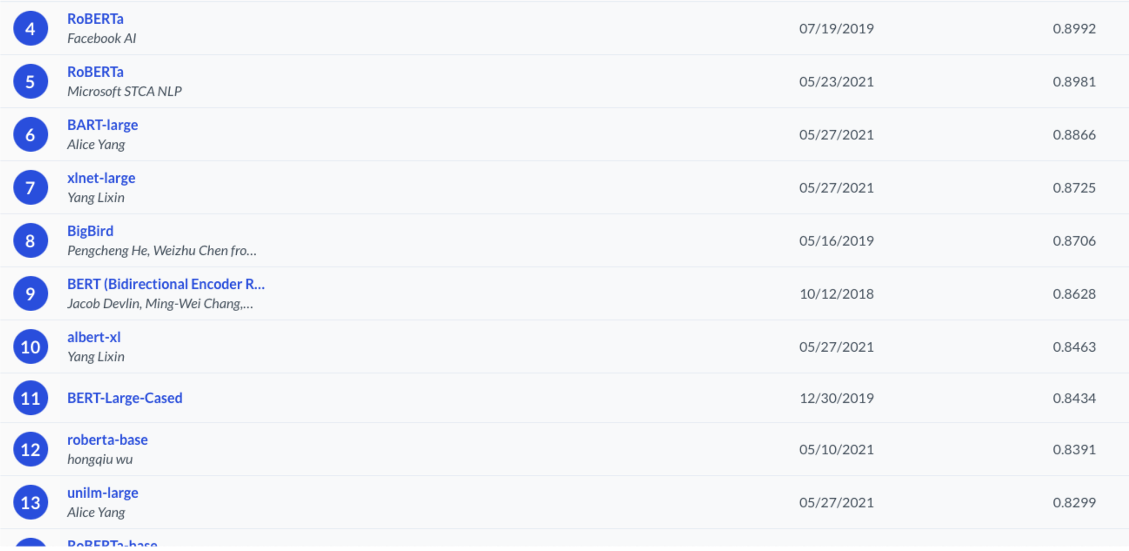
Q: How would you make a house conduct electricity?

A: Paint it with a metal paint

2.8 Dataset

The Macaw model was fine tuned on ARC dataset, a high school based science question and answer dataset. It has the ability to reach logical conclusions from being trained on this dataset. The dataset chosen for this project is the Situations With Adversarial Generations or SWAG[6] dataset, which contains logical follow on sentences from a given source sentence although it is not specifically a question and answer dataset. The dataset contains descriptions like ”she opened the hood of the car,” and we are expected to reason out the situation and anticipate what might come next, E.g - ”then, she examined the engine”.

2.9 SWAG leaderboard



4	RoBERTa *Facebook AI*	07/19/2019	0.8992
5	RoBERTa *Microsoft STCA NLP*	05/23/2021	0.8981
6	BART-large *Alice Yang*	05/27/2021	0.8866
7	xlnet-large *Yang Lixin*	05/27/2021	0.8725
8	BigBird *Pengcheng He, Weizhu Chen fro...*	05/16/2019	0.8706
9	BERT (Bidirectional Encoder R... *Jacob Devlin, Ming-Wei Chang,...*	10/12/2018	0.8628
10	albert-xl *Yang Lixin*	05/27/2021	0.8463
11	BERT-Large-Cased	12/30/2019	0.8434
12	roberta-base *hongqiu wu*	05/10/2021	0.8391
13	unilm-large *Alice Yang*	05/27/2021	0.8299
	RoBERTa-base		

Figure 2.2: SWAG leaderboard

Chapter 3

Methodology

The task that we are trying to solve can be summarized as follows - Obtaining the most suitable option from the give list of possible sequence continuations, while we also have information about the context. Even though this task is a classification task it is not necessary to treat it so. We can also use generative models to generate one of possible options.

3.1 Approaches

3.1.1 Approach 1: Using an encoder only model

Since the task at hand is a classification task, it makes sense to use a model with encoders only such as BERT or variants such as Roberta. We have seen that some of the best models in the leaderboard for the SWAG dataset comprises of encoder only models. This approach would involve adding additional layers on top of the model which would aid in the classification task.

3.1.2 Approach 2: Using an encoder-decoder model

The second approach is to treat this classification task like a generation task. The encoder-decoder model would generate an answer from the context and the list of possible follow on options. Using T5 is a good option in this approach as it is trained on the basis that, whatever type of task you give the model, it always tries to generate the answer.

3.1.3 Approach 3: Using an encoder of an encoder-decoder model

From the literature review section we have observed the EncT5 paper. It basically tells us that taking the encoder alone of an encoder-decoder model such as T5 or Macaw can yield better results for non auto-regressive tasks. Also it would contain lesser parameters within the model compared to the whole encoder-decoder model. So the approach 3 would involve adding additional layers on top of the encoder obtained from the model and these additional layers would do the classification part.

3.2 Solving common-sense inference tasks

In order to solve various commonsense inference tasks we can go with any of the above mentioned methods. Obtaining abstract high dimensional representations of tokens given in the input text is the most difficult part. This part is natural for the usage of BERT as it is trained for generating powerful embeddings. We can try exploring the embedding powers of Macaw by only using the encoder of the model rather than using Macaw as a generation model as described in ENCT5 [2].

3.3 Proposed solution

The EncT5 paper promises pretty good performance from the encoder of an encoder-decoder model. Macaw is a T5 based model, which is an encoder-decoder model and is specifically trained for reasoning based question and answering. This makes for a good case that the encoder of the Macaw model can be further used elsewhere. So the proposed model would involve the encoder of the MACAW-large (700 million parameters) model and added on top of this would be a series of linear layers, which would give the probabilities for the various follow on sentences.

Chapter 4

Work done

Based on the way we give the input we would have different models. This section details 2 of the models which have produced decent results. Both the models are implemented using pytorch and huggingface libraries in python. First let's look at an input instance of the SWAG dataset.

4.1 Input

An instance of the SWAG dataset looks the following:

```
{
  "video-id": "anetv_dm5WXFiQZUQ",
  "fold-ind": "18419",
  "startphrase", "He rides the motorcycle down the hall and into the elevator. He",
  "sent1": "He rides the motorcycle down the hall and into the elevator."
  "sent2": "He",
  "gold-source": "gold",
  "ending0": "looks at a mirror in the mirror as he watches someone walk through a
    door.",
  "ending1": "stops, listening to a cup of coffee with the seated woman, who's
    standing.",
  "ending2": "exits the building and rides the motorcycle into a casino where he
    performs several tricks as people watch.",
  "ending3": "pulls the bag out of his pocket and hands it to someone's grandma.",
  "label": 2,
}
```

4.2 Model 1

4.2.1 Input for model 1

The major components of an input instance we are concerned are the startphrase, ending0, ending1, ending2, ending3, label. The ”startphrase” would give us the contextual information needed to predict which of the ending among the 4 given would follow. The one to follow would be indicated by ”label”. Macaw was trained with instances in the form of ”question = QUESTION; mcoptions = (A) Option1 (B) Option2 (C) Option3 (D) Option 4”. Inorder to take advantage of the way in which it was trained, we train by first converting SWAG dataset into this format. If the label was 0, we want the model to generate the vector l0 and if the label was 1, we want the model to generate vector l1 and so on.

```
temp_df_list = []
l0 = [1, 0, 0, 0]
l1 = [0, 1, 0, 0]
l2 = [0, 0, 1, 0]
l3 = [0, 0, 0, 1]
cnt = 0
for i in dataset_swag["train"]:
    cnt = cnt + 1
    temp_list_2 = []
    temp_list_2.append("$question$ = " + i["startphrase"] + " ; $mcoptions$ = (A) "
     + i["ending0"] + " (B) " + i["ending1"] + " (C) " + i["ending2"] + " (D) " + i
    ["ending3"])
    if (i["label"] == 0):
        temp_list_2.extend(l0)
    elif (i["label"] == 1):
        temp_list_2.extend(l1)
    elif (i["label"] == 2):
        temp_list_2.extend(l2)
    else:
        temp_list_2.extend(l3)
    temp_df_list.append(temp_list_2)
train_df = pd.DataFrame(temp_df_list, columns = ['startphrase', 'ending0', 'ending1
    ', 'ending2', 'ending3'])
```

This would result in instances getting converted to such a form:

```
"$question$ = He rides the motorcycle down the hall and into the elevator. He ;
    $mcoptions$ = (A) looks at a mirror in the mirror as he watches someone walk
    through a door. (B) stops, listening to a cup of coffee with the seated woman,
    who's standing. (C) exits the building and rides the motorcycle into a casino
    where he performs several tricks as people watch. (D) pulls the bag out of his
    pocket and hands it to someone's grandma."
```

Similarly the validation and the testing dataset is also converted into the following structure.

4.2.2 Tokenizer

Once we have converted each instance of the dataset into the format that we require, we can tokenize it. Here we use the T5 tokenizer since Macaw is built on T5. We enable trucation and padding to ensure that if the number of tokens exceeds ”max-length” we trunctate it and if it falls short of ”max-length” we pad it with pad tokens

```
tokenizer = T5Tokenizer.from_pretrained("allenai/macaw-large", Truncation = True,
    padding="max_length")
```

4.2.3 Custom Dataset

Now we define the complete dataset which takes the dataset that we have and then tokenizes it and also generates attention masks which help in identifying which tokens are important and which are simply pad tokens.

```
inputs = self.tokenizer.encode_plus(
            title,
            None,
            add_special_tokens=True,
            max_length=self.max_len,
            padding='max_length',
            # return_token_type_ids=True,
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt'
        )

        return {
            ’input_ids’: inputs[’input_ids’].flatten(),
            ’attention_mask’: inputs[’attention_mask’].flatten(),
            # ’token_type_ids’: inputs["token_type_ids"].flatten(),
            ’targets’: torch.FloatTensor(self.targets[index])
        }
```

4.2.4 Model architecture

The model consists of mainly 3 important components.

- Macaw Encoder
- Dropout (0.3)
- Dense layer (1024 x 4)

The Macaw encoder is responsible for providing the embeddings for our dataset instance. Each token has an embedded form of a 1024 dimensional vector. We take the embedding of the first token of our dataset instance which represents the begin token or BOS. This 1024 dimensional embedding would contain all the contextual information as it passes through the encoder layers of the Macaw encoder and obtains information about the context. This 1024 dimensional vector can be then used for the further classification by sending it to a dropout and then a dense layer.

Here is the model architecture definition:

```
class MacawClass(torch.nn.Module):
    def __init__(self):
        super(MacawClass, self).__init__()
        self.macaw_model = T5EncoderModel.from_pretrained(’allenai/macaw-large’,
    return_dict=True)
        self.dropout = torch.nn.Dropout(0.3)
        self.linear = torch.nn.Linear(1024, 4)

    def forward(self, input_ids, attn_mask):
        output = self.macaw_model(
            input_ids,
            attention_mask=attn_mask,
            # token_type_ids=token_type_ids
        )
        hidden_states = output[0]
        pooled_output = hidden_states[:, 0, :]
        pooled_output = self.dropout(pooled_output)
        logits = self.linear(pooled_output)
        return logits

model = MacawClass()
model.to(device)
```

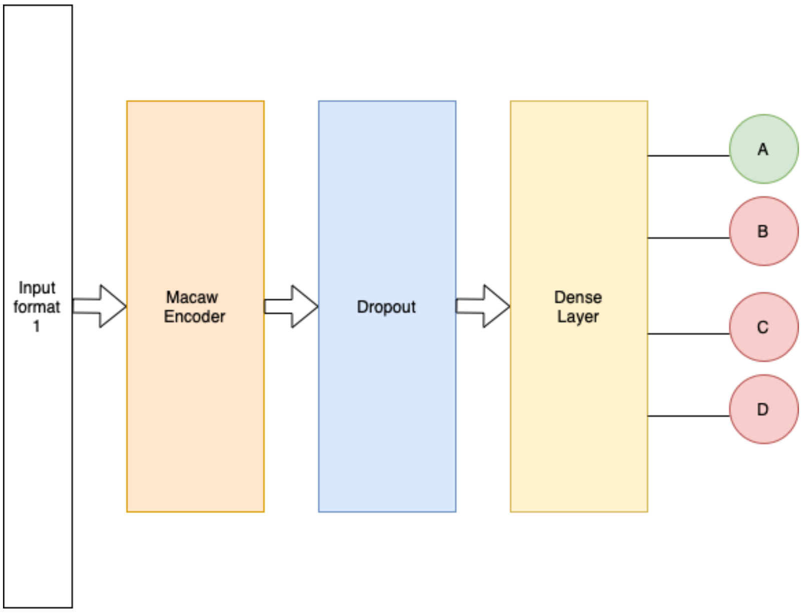


FIGURE 4.1: Model 1

4.2.5 Training

The number of training instances where 73545. Each epoch took approximately 1.5hrs. After each epoch the model checkpoint was saved and loaded when training again. The loss function used was binary cross entropy loss or BCE Loss.

4.3 Model 2

4.3.1 Input for model 2

The major components of an input instance we are concerned are the startphrase, ending0, ending1, ending2, ending3, label. The ”startphrase” would give us the contextual information needed to predict which of the ending among the 4 given would follow. The one to follow would be indicated by ”label”. This time instead of using the question concatenated with all the options we would have 4 seperate instances, i.e question concatenated with each of the option seperately. Then we would choose the instance wherein the question and option match is the highest and that would be taken as the most suitable option.

```
temp_df_list = []
l0 = [0]
l1 = [1]
cnt = 0
for i in dataset_swag["train"]:
    cnt = cnt + 1
    for j in range(4):

        temp_list_2 = []
        t = "ending" + str(j)
        temp_list_2.append("$question$ = " + i["startphrase"] + " ; $mcoptions$ = "
     + i[t])
        if (i["label"] == j):
            temp_list_2.extend(l1)
        else:
            temp_list_2.extend(l0)
        temp_df_list.append(temp_list_2)
train_df = pd.DataFrame(temp_df_list, columns = [’startphrase’, ’ending’])
```

This would produce 3 instances of mismatches and 1 instance of matching occurance. From here on we take equal number of mismatches and matches so that the testing data isn’t biased towards predicting mismatches.

Once we train the model we can find use the model to find which option can produce the highest match score and that would be our answer.

Similarly the validation and the testing dataset is also converted into the following structure.

4.3.2 Tokenizer

Once we have converted each instance of the dataset into the format that we require now we can tokenize it. Here we use the T5 tokenizer since Macaw is built on T5. We enable trucation and padding ensure that if the number of tokens exceeds ”max-length” we trunctate it and if it falls short of ”max-length” we pad it with pad tokens

```
tokenizer = T5Tokenizer.from_pretrained("allenai/macaw-large", Truncation = True,
    padding="max_length")
```

4.3.3 Custom Dataset

Now we define the complete dataset which takes the dataset that we have and then tokenizes it and also generates attention masks which help in identifying which tokens are important and which are simply pad tokens.

```
inputs = self.tokenizer.encode_plus(
            title,
            None,
            add_special_tokens=True,
            max_length=self.max_len,
            padding=’max_length’,
            truncation=True,
            return_attention_mask=True,
            return_tensors=’pt’
        )

        return {
            ’input_ids’: inputs[’input_ids’].flatten(),
            ’attention_mask’: inputs[’attention_mask’].flatten(),
            # ’token_type_ids’: inputs["token_type_ids"].flatten(),
            ’targets’: torch.FloatTensor(self.targets[index])
        }
```

4.3.4 Model architecture

The model consists of mainly 3 important components.

- Macaw Encoder
- Dropout (0.3)
- Dense layer (1024 x 1)

The Macaw encoder is responsible for providing the embeddings for our dataset instance. Each token has an embedded form of a 1024 dimensional vector. We take the embedding of the first token of our dataset instance which represents the begin token or BOS. This 1024 dimensional embedding would contain all the contextual information as it passes through the encoder layers of the Macaw encoder. This 1024 dimensional vector can be then used for the further classification by sending it to a dropout and then a dense layer.

Here is the model architecture definition:

```
class MacawClass(torch.nn.Module):
    def __init__(self):
        super(MacawClass , self).__init__()
        self.macaw_model = T5EncoderModel.from_pretrained(’allenai/macaw-large’,
    return_dict=True)
        self.dropout = torch.nn.Dropout(0.3)
        self.linear = torch.nn.Linear(1024, 1)

    def forward(self , input_ids , attn_mask):
        output = self.macaw_model(
            input_ids ,
            attention_mask=attn_mask ,
        )
        hidden_states = output[0]
        pooled_output = hidden_states[:, 0, :]
        pooled_output = self.dropout(pooled_output)
        logits = self.linear(pooled_output)
        return logits

model = MacawClass()
model.to(device)
```

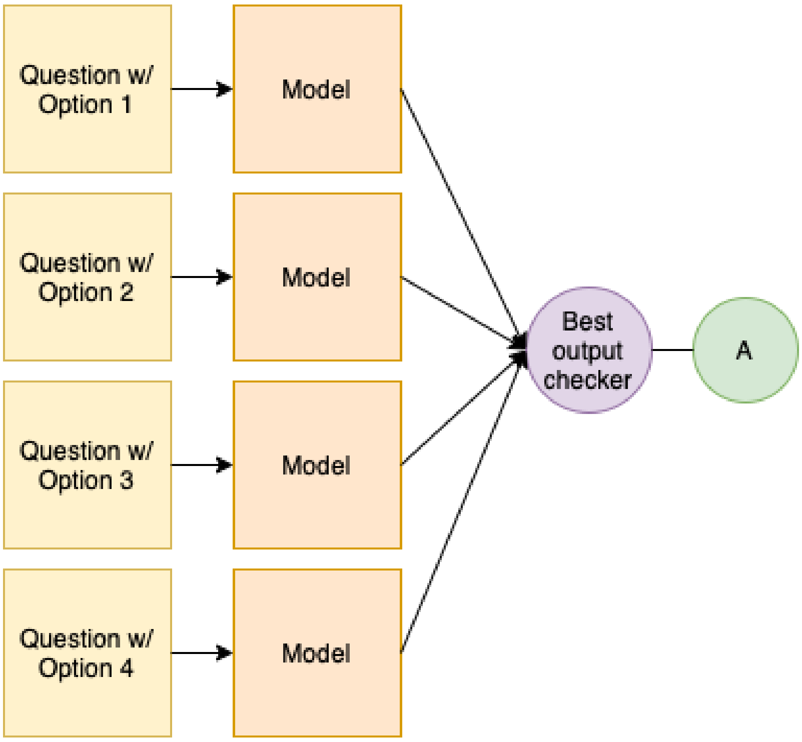


FIGURE 4.2: Abstract representation of model 2

Each "Model" block represents the model given below.

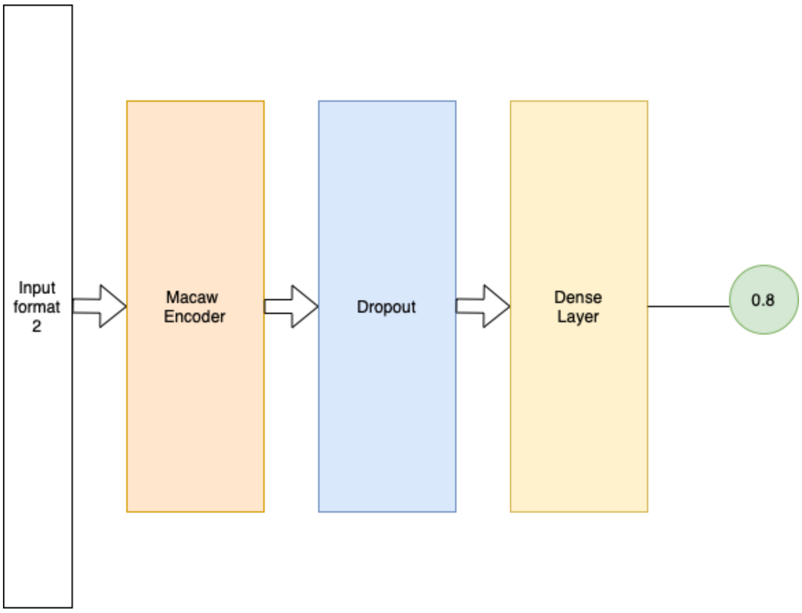


FIGURE 4.3: Model 2

4.3.5 Finding best match

For finding the option matching the most, we compare the final output for all the 4 instances, i.e question with each option and take the maximum valued output. This would be our final match.

4.3.6 Training

The number of training instances where 147090. Each epoch took approximately 2.7hrs. After each epoch the model checkpoint was saved and loaded when training again. The loss function used was binary cross entropy loss or BCE Loss.

Chapter 5

Results and discussion

The two different models have been trained for 7 epochs each, with a learning rate of 0.00001 in batches of 8. The test datasets had an equal distribution of questions having options 1, 2, 3 and 4.

5.1 Model 1

The number of testing instances in Model 1 was a total of 20,008 with each class having approximately one forth of the total instances. 15,974 instances gave the correct output, resulting in an accuracy of 79.87 percent. These are the F1 scores per class obtained for model 1.

- Class A - 0.800
- Class B - 0.794
- Class C - 0.801
- Class D - 0.798

5.2 Model 2

The number of testing instances in Model 1 was a total of 20,006 with each class having approximately one forth of the total instances. 16,561 instances gave the correct output, resulting in an accuracy of 82.8 percent. These are the F1 scores per class obtained for model 2.

- Class A - 0.825
- Class B - 0.820
- Class C - 0.827
- Class D - 0.828

Model	**Accuracy**	**F1 score**	**MCC**
Model 1	79.8	.797	.731
Model 2	82.8	.825	.816

Table 5.1: Results

Model	**Accuracy**
Human performance	88.0
Model 2	82.7
T5 large	72.6
RoBERTa base	82.6

Table 5.2: Comparison between various models

The SWAG dataset[7] leader as of writing this report is DeBERTa, by Microsoft AI with an accuracy of 91. But we have obtained data showing us the capabilities of using Macaw encoder which has produced a higher accuracy than T5 large on which Macaw is based. T5 has an encoder and a decoder whereas this model only has an encoder i.e with less than half the parameters of T5 this model has outperformed it.

Chapter 6

Conclusions and future scope

The Model 1 proposed, followed the input scheme preferred by the Macaw model for its generative tasks and we obtained an accuracy of 79.8 percent compared to Model 2, in which we redefined the input scheme for more of a ”similarity” task. Further we got an accuracy of 82.8 percent indicating that the second way of encoding the input is better suited for the model.

We have seen that the model 2 mentioned in this report does well in SWAG dataset compared to some of the other generative models such as T5. This results gives us a promising insight on the capabilities of a generative model's encoder and that with very little fine tuning these models can do pretty well in non auto-regressive tasks.

Although this model hasn't performed as well as some of the other encoder based architectures out there we have gathered enough data to show that the model we have created can outperform some of the bigger generative models in which this encoder is a part of.

This model can also be fine-tuned on other classification tasks such as sentiment analysis, sentence similarity, answer span prediction etc and expect it to deliver good results.

Bibliography

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[2] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, vol. abs/1910.10683, 2019. [Online]. Available: http://arxiv.org/abs/1910.10683

[3] F. Liu, S. Shakeri, H. Yu, and J. Li, "Enct5: Fine-tuning T5 encoder for non-autoregressive tasks," *CoRR*, vol. abs/2110.08426, 2021. [Online]. Available: https://arxiv.org/abs/2110.08426

[4] O. Tafjord and P. Clark, "General-purpose question-answering with macaw," *CoRR*, vol. abs/2109.02593, 2021. [Online]. Available: https://arxiv.org/abs/2109.02593

[5] D. Khashabi, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi, "Unifiedqa: Crossing format boundaries with a single QA system," *CoRR*, vol. abs/2005.00700, 2020. [Online]. Available: https://arxiv.org/abs/2005.00700

[6] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, "SWAG: A large-scale adversarial dataset for grounded commonsense inference," *CoRR*, vol. abs/1808.05326, 2018. [Online]. Available: http://arxiv.org/abs/1808.05326

[7] "Swag leaderboard." [Online]. Available: https://leaderboard.allenai.org/swag/submissions/public

[8] "Language models." [Online]. Available: https://insights.daffodilsw.com/blog/what-are-language-models-in-nlp

[9] "Types of transfer learning." [Online]. Available: https://www.trainerslibrary.org/transfer-of-learning-types-and-theories/

[10] Y. Zhang, S. Sun, M. Galley, Y. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "Dialogpt: Large-scale generative pre-training for conversational response generation," *CoRR*, vol. abs/1911.00536, 2019. [Online]. Available: http://arxiv.org/abs/1911.00536

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762