Reinforcement Learning Agent for Text-Based Games

A PROJECT REPORT

Submitted by

M Rakesh Raj

(Shiv Nadar University Chennai)

Under the Guidance of

Dr. S. Usha Kiruthika

(Assistant Professor, Department of Computer Science and Engineering)

For

Summer Internship 2023

Submitted to

National Institute of Technology

Tiruchirappalli

July 2023

Table of Contents

ABSTRACT

Combining Reinforcement Learning (RL) and Natural Language Processing (NLP) has been the trend now, the introduction of **Chat-GPT** has revolutionised the field of NLP and RL. Implementation of NLP and RL in text-based game involves several models used in combination. Unlike **Chat-GPT** it doesn't involve any human feedback.

The use of these sub-models that constitutes the main model will be explained in this report together with how these models are used and their benefits and drawbacks. Several models were tried out during the course of the internship and their comparative analysis is yet to be performed.

The Text-based game being used here is **ZORK - I** which has been executed using **frotz** compiler, it is a terminal based game which involves no GUI. Therefore, the **pexpect** library in python was used for the input and output retrieval.

1. INTRODUCTION

Reinforcement Learning is branch of AI that involves training a computer to act and behave like a human brain, like how we humans when starting any activity new fail in the first several trials, learn from those mistakes and make ourselves better.

Text-based game is not a great application of use but NLP models involved in that can be used in several other places for understanding text and context. Text based games like this depend on long term rewards.

ZORK-I is the first game of the Zork trilogy. Each game has a vast map that incorporates almost more than 20 different locations that are found by the player using the clues hidden in the description of each location. The game has been around since 1980 and is commended as one of the most interesting text based game trilogy.

2. Topic Briefs

2.1. Reinforcement Learning

➡ While reinforcement learning has been a topic of much interest in the field of AI, its widespread, real-world adoption and application remain limited. Noting this, however, research papers abound on theoretical applications, and there have been some successful use cases.Current use cases include, but are not limited to, the following:

◉ gaming

◉ resource management

◉ personalised recommendations

◉ robotics

➡ Gaming is likely the most common usage field for reinforcement learning. It is capable of achieving superhuman performance in numerous games. A common example involves the game *Pac-Man*.

➡ One of the barriers for deployment of this type of machine learning is its reliance on exploration of the environment. The currently considered game is one of that sort taking consecutive best action is very tough due to the vast possible command palette. However the training of the model can result in better outcomes if the environment is exploited.

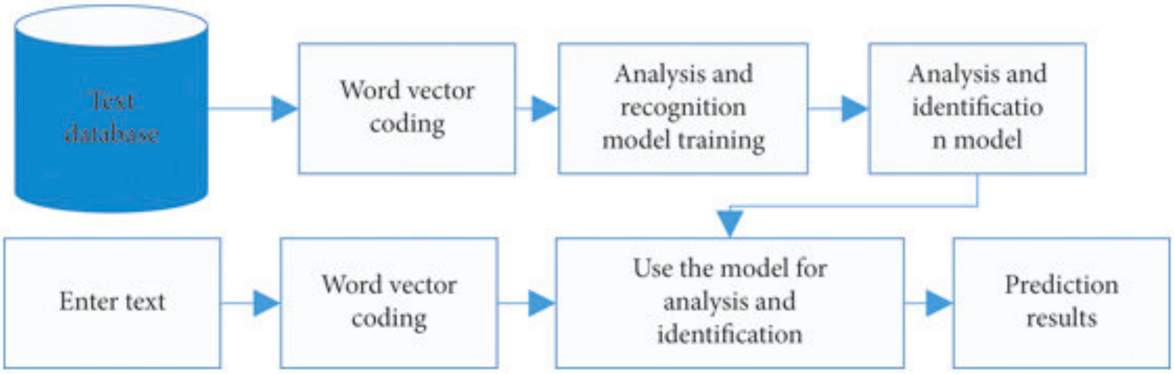
2.2. NLP

➡ Natural Language Processing also provides computers with the ability to read text, hear speech, and interpret it, by extracting the context and representation using word embeddings.

➡ Word Embedding such as Word2Vec, Glue, etc were considered as huge improvements as the words resented by integers were able to capture the meaning and differentiate between other words. Other advanced models such as BERT have taken this to the next level by producing better representation of words that might express a different meaning in different context.

➡ The very popular ChatGPT uses RL combined with NLP for text generation. It was trained using Human Feedback mechanism for RL and a transformer decoder for the output. GPT-3 having over 175 million parameters requires high computational resources but also executes a human like response to any conversation, and the evolution to GPT-4 has increased this to 100 trillion parameters. Chat-GPT-4 has promised to be a revolutionary invention that can also act like a personal tutor together with several other capabilities.

➡ The image below explains a very brief flow chart of how an NLP model progresses.


Text database
Word vector coding
Analysis and recognition model training
Analysis and identificatio n model
Enter text
Word vector coding
Use the model for analysis and identification
Prediction results

3. Model and Implementation

3.1 Actor-Critic Model

✓ For the model the Advantage Actor-Critic model has been used which combines the value function returned but the critic network and the Q-value from the agent network to create a formula that computes the advantage.

✓ Given below is the cost function of an A2C model

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ A^w(s, a)\right] \quad \text{Advantage Actor-Critic}$$

3.1.1 A2C Implementation

✓ The Game starts off by displaying the description of the place the game starts which is the **West of House** (WoH). The context of this description is necessary to decide the consecutive action. For this purpose a **distilbert-base-uncased** model has been deployed to get the word embeddings of the words in the sentence that follows an **LSTM** layer to capture the long term dependencies (The LSTM layer consists of 128 LSTM cells), from which only the last time-step representation has been retrieved to serve as the essence of the description. The values fetched from the LSTM are then normalised using a **NormalisationLayer** and needed into the **Feed-Forward-NN** which outputs a probability distribution of **75** different commands that can be executed in the game. These probability values can be considered as the Q-value for each action in that state. The same LSTM outputs are also fed into another ANN for the **critic network**.

✓ The Loss functions for these networks are derived from the A2C algorithm and **epsilon** values are altered accordingly for balance between exploration and exploitation.

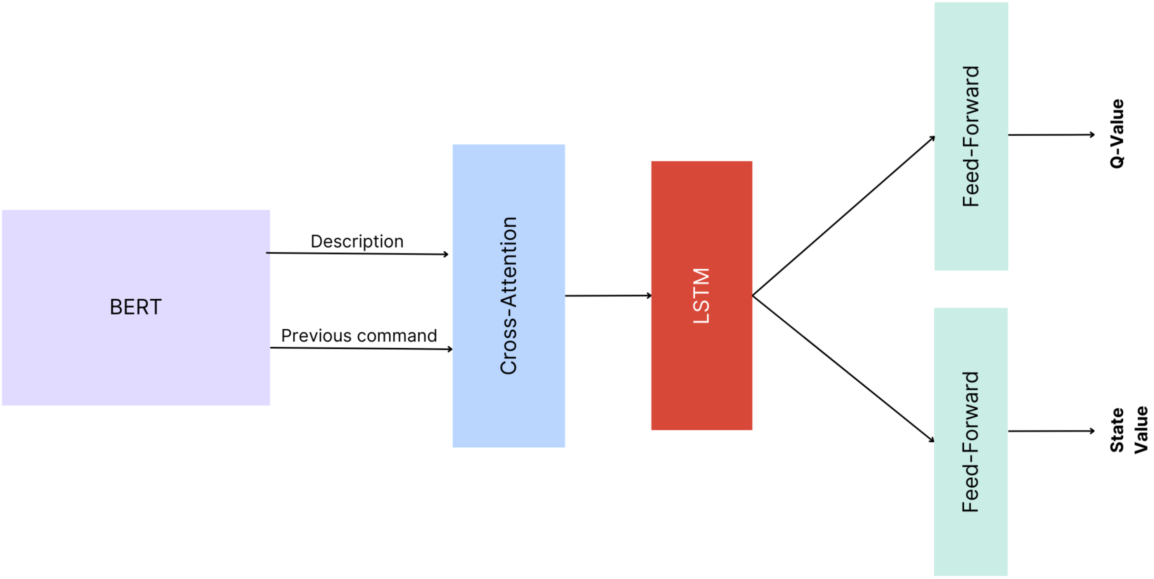
3.1.2 Sentiment Analysis

✓ The sentiment model used is a separately fine-tuned model of **DistilBert.**

✓ The **Dataset** used for the sentiment analysis model is the **StanfordSentimentTree** (SST) that is considered to be one of the best datasets for sentiment analysis. SST contains close to 11000 sentences of 5 different sentiments, namely **most-negative, negative, neutral, positive, most-positive**, obtained from **RottenTomato.**

✓ The embeddings of words for the model is derived from DistilBert and passed through a **Bi-LSTM Layer** which captures representation from either sides of the input, which is then sent to a Feed-Forward-NN to undergo sentiment classification.

3.1.3 The Complete Approach

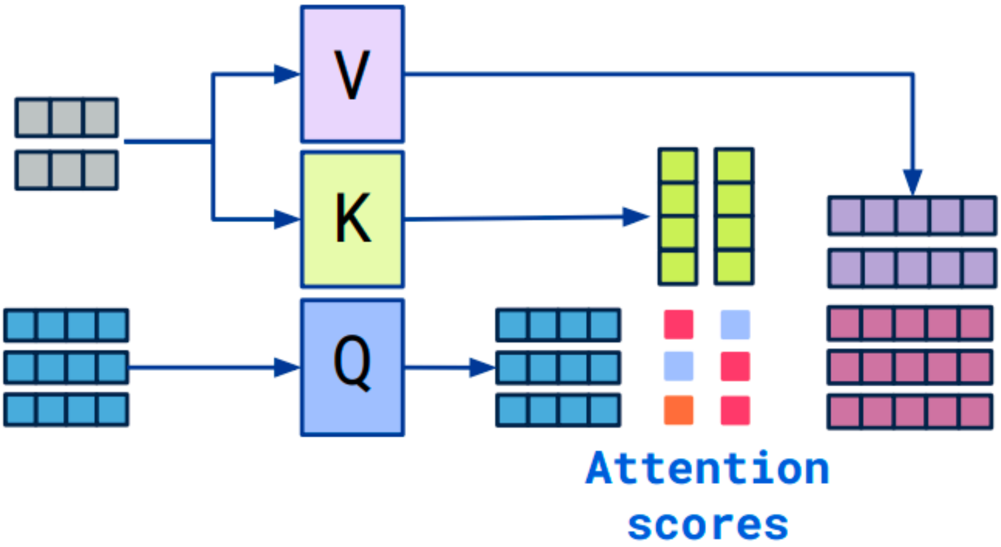


1. The first component of this model consists of the multilayered neural network mentioned above that outputs probability of each action.

2. Since the agent is considered to have no knowledge about the environment and possible commands we use cosine similarity to pair verbs (probabilities in the last hidden state of the NN) and nouns together. The actions chosen from the NN and objects available in the environment and items (stuff in the inventory) are passed through DistilBert and confine similarities of their embedding are taken to access their validity.

3. The finalised action is then passed to the game via the process spawned and the response is recorded. If the action results in a positive reward the response is added to the location description else the response is neglected.

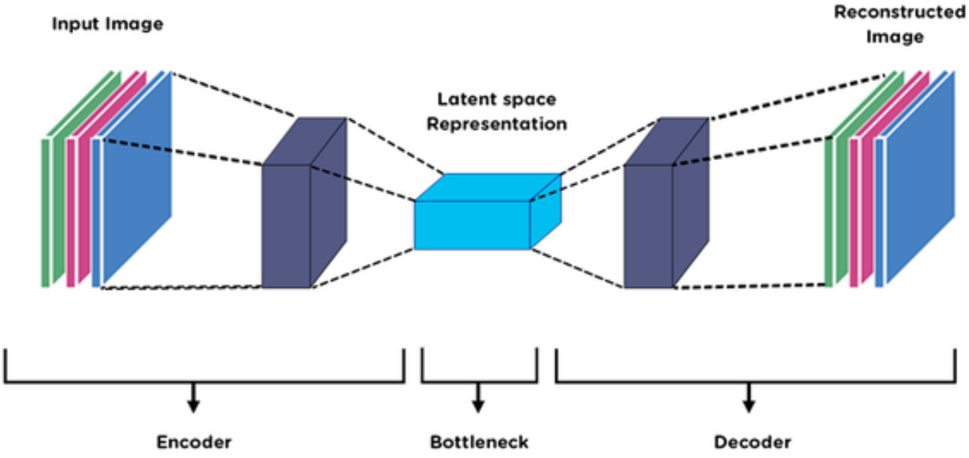
4. There exists a custom **cross-attention layer** in the NN whose working was borrowed from that of transformers. This layer takes in two inputs the description of the location and the previous action taken (passed only if the action taken results in a positive reward). In this way the part of the description that relates to the action taken is given lesser importance than the other parts, so only the unattended parts of the description is of use for the consecutive step.



3.2 AutoEncoder Model

3.2.1 AutoEncoder

- **AutoEncoders** (AE) are networks that are used to capture the most essential components of the input and try to reproduce the entire input using them. In this application the AE instead of reproducing the output produces the action that has to be performed.



-

3.2.2 Complete Approach

- This model receives a small tweaking in the architecture from the initial one. It replaces the Feed-Forward-NN with an AE that computes the embedding of the action to be performed by taking in the description.
- The AE takes in the sentence representation from the **GRU** (the LSTM was switched with GRU) which is of size **128** and outputs an embedding of size **768**.

- The loss function is also similar to that of the first model, the probability of the actions become the similarity between the embedding and the action embedding.

3.3 Command Generation Model

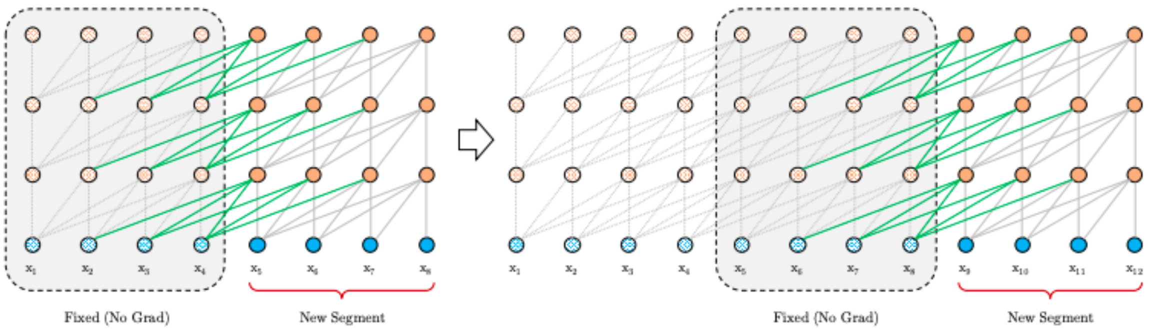
- Text generation models are used with growing importance and in this implementation it is fine tuned to support the command generation.

3.3.1 Dataset

- The Dataset used was obtained from a **GitHub** repository. It contains almost 1.5 lakh sample commands and the description for training. The drawback of this dataset is that it doesn't contain any negative commands of commands that are not valid.

3.3.2 Transformer XL

- Transformer XL (T-XL) is a step up from the vanilla transformer model. T-XL model takes into consideration the relation between different segments of the data and retain the past segments representation as an input to the next segment, whereas the vanilla transformer doesn't gather inter-segment relations.
- It uses a method known as relative **Positional Encoding** scheme.


x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Fixed (No Grad)
New Segment

- T-XL also has an embedding layer that uses a loop-up table for word embedding but the model will use BERT for this purpose.

3.3.3 Bert2Transformer-XL

- Bert2Transformer-XL (B2XL) is used as a **seq2seq** model for command generation. The Bert part of the model acts as the encoder and the T-XL part acts as the decoder of the model. The description is passed through the Bert model and the Bert embedding of the commands are passed as the input to the decoder and as labels. The model is trained separately and the saved model is used in the application

4. Future Improvements and Conclusion

❖ The model’s object and item selection done to pair verbs and nouns will be enabled using a **MaskedLM** model fine-tuned on command datasets.

❖ Create a implementable library of this model with Exception Handling.

❖ All the three models were designed without error the performance of some are yet to be tested.

5.References

-> Recursive Deep Models for Semantic Compositionality Over a SentimentTreebank Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)

-> Prithviraj Ammanabrolu, Mark O. Riedl, Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning

-> Karthik Narasimhan, Tejas Kulkarni, Regina Barzilay, Language Understanding for Text-based Games using Deep Reinforcement Learning

-> Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Google AI Language, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.