Department of Sociology and Social Research

University of Trento, Italy

Introduction to Machine Learning for Natural Language Processing

A.Y. 2021/2022

Final Essay

23-01-2023

Shaker Mahmud Khandaker

shaker.khandaker@studenti.unitn.it

Matricola: 229221

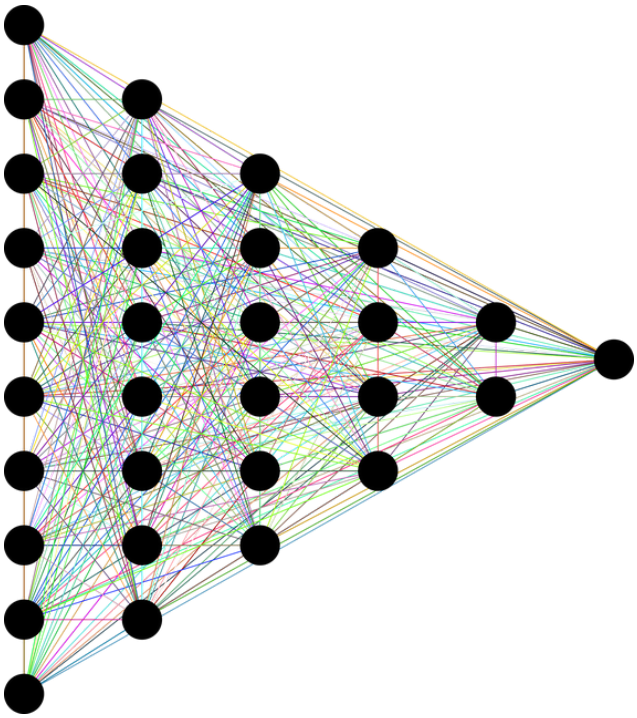
GitHub link : https://github.com/khandakerrahin/introduction_to_ML_for_NLP

Table of contents

Part 01: Algorithms

Adopt a Network

Analyzing the sentiment of movie reviews



1. Introduction

The task of sentiment analysis is to classify text data, such as movie reviews, into predefined categories such as positive, negative, or neutral. Sentiment analysis is an important problem in natural language processing as it enables various applications such as opinion mining, brand monitoring, and customer feedback analysis. In this example, the focus is on analyzing the sentiment of movie reviews. The hypothesis is that BERT, a pre-trained model, can be fine-tuned to perform well on the sentiment analysis task. BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model that has been pre-trained on a massive amount of text data and fine-tuning it on a specific task can lead to improved performance. The research question is to investigate the performance of the BERT model for sentiment analysis on movie reviews. The goal of this research is to fine-tune the pre-trained BERT model on a movie review dataset and evaluate its performance on the task of sentiment analysis. The movie review dataset is labeled as positive or negative and it will be used to train the model and evaluate its performance. The research will also investigate the impact of different hyperparameter settings, such as the number of training steps, on the performance of the model. The results of this research will provide insight into the effectiveness of BERT for sentiment analysis and the factors that influence its performance. Additionally, the model can be used for other similar tasks such as customer feedback analysis and opinion mining on other types of text data such as social media posts, product reviews, and news articles.

2. Data

The dataset used for this task is a dataset of movie reviews labeled as positive or negative. The dataset contains a large number of movie reviews, which is necessary to fine-tune the BERT model. The data is preprocessed by removing punctuation marks and stop words, and converting all the reviews to lowercase. The text of the movie reviews is tokenized and the input to the model is the tokenized text, padded or truncated to a fixed length. The output of the model is the sentiment of the review, which is binary in this case (positive or negative).

The dataset is appropriate for this task as it contains a large number of movie reviews labeled with their sentiment. However, one shortcoming is that the dataset may not be representative of all movie reviews, as it is limited to a specific dataset. The dataset might also have some biases, for example, the positive reviews might be more frequent than the negative ones. Furthermore, the dataset might not include all the nuances of the language, like sarcasm, irony, and figurative language which are commonly found in informal text.

To overcome these shortcomings, the data can be augmented by using techniques such as back-translation, data generation using GPT-3, and adversarial training. Additionally, the data can be balanced by oversampling the minority class and/or undersampling the majority class. To further improve the model's performance, the dataset can also be augmented by including additional labeled data from other sources, such as twitter, product reviews or customer feedback.

3. Model

The model used for this task is the BERT (Bidirectional Encoder Representations from Transformers) model, which is a pre-trained transformer-based model for natural language processing tasks. BERT is trained on a massive amount of text data and fine-tuned on specific tasks can lead to improved performance. The BERT model architecture consists of a multi-layer transformer encoder with attention mechanisms. The transformer encoder allows the model to learn the contextual relationships between words in the input text by attending to different parts of the input.

In this task, the BERT model is fine-tuned on the movie review dataset by keeping the pre-trained weights fixed and updating only the final layers to adapt the model to the sentiment analysis task. The fine-tuning process is done by adding a classification head on top of the pre-trained model and training the model on the labeled movie review dataset. The classification head is a fully connected layer with two neurons (for positive or negative sentiment) connected to the final hidden state of the pre-trained model.

The choice of BERT for this task is appropriate because it has been shown to perform well on sentiment analysis tasks and fine-tuning the model allows it to be adapted to the specific task of analyzing movie reviews. Additionally, BERT has been pre-trained on a massive amount of text data, which allows it to have a good understanding of the language, and it can be fine-tuned on a smaller dataset to perform well on a specific task.

It is worth mentioning that BERT is not the only architecture for NLP and there are other architectures such as ALBERT, RoBERTa, XLNET etc which can be used for similar task with fine-tuning. The choice of architecture can be based on the dataset size and the computational power available. Additionally, other architectures such as GPT-2, GPT-3, and T5 can also be considered for this task.

4. Training Regime

The BERT model for sentiment analysis is fine-tuned using a standard training-validation-testing split approach. The training dataset is used to adjust the pre-trained BERT model, the validation dataset is utilized to fine-tune the hyperparameters, and the test dataset is used to evaluate the model's performance. The fine-tuning procedure includes adding a classification head on top of the pre-trained model and training it on the labeled movie review dataset. The classification head is a fully connected layer with two neurons that are responsible for identifying positive or negative sentiment and connected to the final hidden state of the pre-trained model. The model is trained using a binary cross-entropy loss function, which compares the predicted sentiment to the actual sentiment.

The training process can be further enhanced by using techniques such as learning rate scheduling, early stopping, and data augmentation. Learning rate scheduling reduces the learning rate over time to help the model converge more quickly and prevent overfitting. Early stopping stops training when the performance on the validation dataset stops improving. Data augmentation increases the size of the training dataset and makes the model more robust to variations in the data.

The model's performance is evaluated by measuring the accuracy, precision, recall, and F1 score on the test dataset. These metrics are commonly used to evaluate the performance of classification models and provide a comprehensive understanding of the model's performance. The evaluation process can also include other metrics such as AUC-ROC and confusion matrix.

Additionally, the choice of hyperparameters such as batch size, learning rate, and number of training steps can significantly impact the model's performance. The performance of the model can be further enhanced by tuning these hyperparameters using techniques such as grid search or random search.

5. References

[1] https://arxiv.org/pdf/1902.04094

[2] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[3] https://huggingface.co/blog/bert-101

Part 02: Paper Review

Multi-Agent Cooperation and the Emergence of (Natural) Language

Angeliki Lazaridou - Alexander Peysakhovich - Marco Baroni

Summary

The paper "Multi-Agent Cooperation and the Emergence of (Natural) Language" by Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni, presented at ICLR 2017, presents a research program that focuses on the use of multi-agent coordination communication games as a way to study the emergence of natural language. The authors propose a framework for language learning that is based on multi-agent communication and they test this framework using referential games. In these games, two agents, a sender and a receiver, are presented with a pair of images and the sender is tasked with sending a message from a fixed vocabulary to the receiver to help the receiver identify the target image. Through this process, the agents learn to develop their own language interactively out of the need to communicate. The authors show that networks with simple configurations are able to learn to coordinate in the referential game and they explore ways to adjust the game environment to make the "word meanings" induced in the game more closely align with intuitive semantic properties of the images. They also present a strategy for grounding the agents' code into natural language as a step towards developing machines that can communicate with humans productively. The authors also discuss the importance of designing environments that foster the development of a language that is portable to new situations and to new communication partners, specifically humans, and they propose studying the agents association of general conceptual properties, such as broad object categories, to the symbols they learn to use and by examining whether the agents' "word usage" is partially interpretable by humans in an online experiment. The authors also compare their approach to other related works and discuss their benefits and drawbacks.

Strengths and weaknesses

This paper proposes a new way to study the emergence of natural language through the cooperation of multiple agents in referential games. The authors suggest that a language can be developed through the need for communication and mutual learning between

agents. The paper demonstrates that simple networks can learn to coordinate in referential games, which is a crucial step towards creating machines that can communicate with humans effectively.

The task used in the experiment is simple enough to be easily understood and analyzed. It was noteworthy that informed agents made use of multiple symbols to transmit the message, while agnostic agents relied on only two symbols. Additionally, the paper emphasizes the importance of designing environments that promote the development of a language that can be applied to different situations and communication partners, specifically humans. This can be achieved by studying the agents' association of general conceptual properties, such as broad object categories, to the symbols they learn to use and by examining whether the agents' "word usage" is interpretable by humans in an online experiment.

The experiments presented in the paper are relatively basic and have certain limitations. The paper is based on simulation and not on real-world interactions, and it does not provide an in-depth analysis of the results. Also, the paper is weak on references and related work, specifically in the area of modeling language evolution as multi-agent cooperation. For instance, the paper does not mention the Lewis Signaling Game which is similar to the proposed framework in the paper. Finally, the task ultimately reduces to image classification if the two images sent are from different categories. The symbols used are effectively the image class, which the second agent learns to assign to one of the images. This approach is similar to a transfer learning problem, which could likely be trained faster using a different method than reinforcement learning.

Overall, the paper presents an intriguing approach to studying the emergence of natural language, but further research is necessary to fully evaluate its potential.

Potential impact

This paper addresses a question that has a potentially wide impact on the way we model natural language and the development of interactive machines. The idea of using multi-agent communication games to study the emergence of natural language is a novel approach that could lead to a better understanding of how natural language develops and how it can be replicated in machines.

The proposed framework for language learning that relies on multi-agent communication is an important step toward the development of conversational agents that can effectively communicate with humans. The ability to develop a language that is portable to new situations and to new communication partners is essential for the development of machines that can interact with humans naturally and productively.

The results of the paper also have implications for other aspects of Computational Linguistics, AI, and Cognitive Science. The idea of using referential games to study the emergence of natural language can be extended to other multi-agent communication games, which can lead to a better understanding of how language develops in different contexts and how it can be replicated in machines.

Furthermore, the idea of studying the agents' association of general conceptual properties, such as broad object categories, to the symbols they learn to use is important for understanding how language is grounded in the real world and how it can be used in applications such as computer vision and robotics.

Overall, the paper addresses a question that has a potentially wide impact on the way we model natural language and the development of interactive machines and could lead to important advances in the field of Computational Linguistics, AI, and Cognitive Science.

The soundness of the work

The experimental work in the paper, in my opinion, appears to be sound and addresses the research question of how to design environments that foster the development of a language that is portable to new situations and to new communication partners. The authors use referential games, which is a simple and well-defined environment to study the emergence of natural language through multi-agent communication.

The authors demonstrate that two networks with simple configurations can learn to coordinate in the referential game, which is an important first step toward developing machines that can communicate with humans productively. They also explore how to make changes to the game environment to cause the "word meanings" induced in the game to better reflect the intuitive semantic properties of the images.
In addition, the authors present a simple strategy for grounding the agents' code into natural language, which is an important step towards the development of machines that can communicate with humans productively.

The idea of using interactive referential communication games to train natural language systems through the cooperation of multiple agents is an innovative and promising approach. The authors mention that there has been some previous research in this area, but it's an area worth exploring further. The combination of these games with supervised learning, as demonstrated in the experiments described in Section 5 and suggested in Section 6, appears to be a particularly effective method.

Overall, the experimental work in the paper seems sound and addresses the research question, and the authors present a clear and well-structured methodology. However, further research and validation in more complex scenarios should be done to ensure the generalization and the reliability of the proposed framework

Replicability

If I were tasked to reproduce the work done in the paper, I would be missing some information about the authors' pipeline. The paper does not provide enough detail about the data preparation, such as the dataset used for training and testing, the pre-processing steps that were applied to the data, or the specific architecture of the neural networks used in the experiment.

Additionally, in **General Training Details** section, the paper uses the hyperparameters without any tuning and does not provide enough information about the hyperparameter tuning process, such as the range of values that were used for the different hyperparameters, the method used for selecting the final set of hyperparameters, or the criteria used for determining the best-performing model.

Furthermore, the paper does not provide any information about the computational resources that were used for training the model, such as the number of GPUs, the amount of memory, or the computational time.

Without these details, it would be difficult to reproduce the work done in the paper, and it would be challenging to determine if the results can be replicated or if they are specific to the particular dataset, architecture, or computational resources used in the experiment.

It should be noted that replicability is an important aspect of scientific research, and authors should provide enough information about their pipeline to allow others to reproduce their work.

Substance

This paper aims to explore how environments can be designed to promote the development of natural language in a portable and adaptable way, specifically for communication with humans. The authors propose a new framework that utilizes

multi-agent communication in referential games to study this. However, there are several other important considerations that could be further explored to enhance understanding of this framework, such as:

- How does the size of the vocabulary affect the emergence of natural language in the proposed framework?
- How does the presence of a third agent or more agents in the game affect the emergence of natural language?
- How does the complexity of the game environment affect the emergence of natural language?
- How does the presence of a teacher agent that provides feedback to the agents affect the emergence of natural language?
- How does the proposed framework compare to other related approaches for the development of interactive machines such as reinforcement learning or evolutionary algorithms?

Answering these questions would provide a deeper understanding of the proposed framework and its potential applications in natural language processing and interactive machine development.

Final Thoughts

The authors propose an innovative approach to studying communication between multiple agents by first learning communication between two agents and then grounding this communication in human language. This is a unique approach compared to standard sequence-to-sequence models which tend to focus on statistical properties rather than functional aspects. The proposed task and framework, as well as the analysis and visualization of the communicated tokens, are valuable contributions that can serve as a foundation for future research. Overall, this paper should be accepted as it offers a valuable perspective on the challenging topic of learning shared representations for communication in a multi-agent setup.

Part 03: Analysis

Introduction to Reinforcement Learning

Learning politeness: An Italian café

1. Introduction

For the analysis part of the report, we are selecting the practical Introduction to Reinforcement Learning, which includes Q-learning algorithm implementation in Python. The goal is to learn the optimal policy for a dialogue system. The dialogue system is represented as a set of states that the agent can transition between based on the actions it takes, which are represented as words. The agent starts in the state 0, and based on the word it selects, it may transition to a new state with a certain probability and receive a certain reward. The agent's goal is to maximize the total reward it receives. The Q-matrix is initialized with 0s and is updated during the training process using the Q-learning update rule. The agent explores the environment by randomly selecting an action from the set of possible actions at each state, and exploits the knowledge it has gained by choosing the action with the highest Q-value. The agent's performance is evaluated by the total rewards it receives during each episode.

In this project, we aimed to design an RL agent that learns some obvious politeness rules in an Italian café. The café has a sign above the counter that displays the prices of a coffee based on the politeness of the customer's request: "un caffè" costs 3 EUR, "buongiorno, un caffè" costs 2 EUR, and "buongiorno, un caffè per favore" costs 1 EUR. The goal of the agent is to learn the optimal sequence of actions that results in the lowest cost for a coffee.

2. Data and Model

To implement this RL problem, we would need to define the environment, including the states, actions, and rewards as described earlier, initialize the agent with a policy, such as an epsilon-greedy strategy, which balances exploration and exploitation, implement the value function, such as using Q-learning, to update the agent's knowledge of the expected rewards for different actions in different states, run the algorithm for a set number of

episodes or until the agent reaches a satisfactory level of performance and finally, update the policy as the agent learns more about the environment.
The environment for this problem was defined as a set of states, actions, and rewards. The states represent the different phases of the customer's request, the actions represent the different phrases the customer can say, and the rewards represent the cost of the coffee. The environment for this RL problem can be defined as follows:

Cases:

Case 1: The café owner is in a good mood
Case 2: The café owner is busy

States:

The states in this problem represent the various stages of the interaction between the agent (customer) and the café owner. There are 16 states in total. The states represent the following:

State 0: The initial state where the customer can say "buongiorno" or "un caffè"
State 1: The customer says "un caffè"
State 2: The customer says "per favore" or "EOS"
State 3: The customer says "EOS"
State 4-6: The customer is charged 1-3 EUR for the coffee
State 7: The customer says "un caffè per favore"
State 8: The customer says "per favore" or "EOS"
State 9-11: The customer is charged 1-3 EUR for the coffee
State 12: The customer is charged 2 EUR for the coffee
State 13-15: The customer is given a free coffee with a big smile

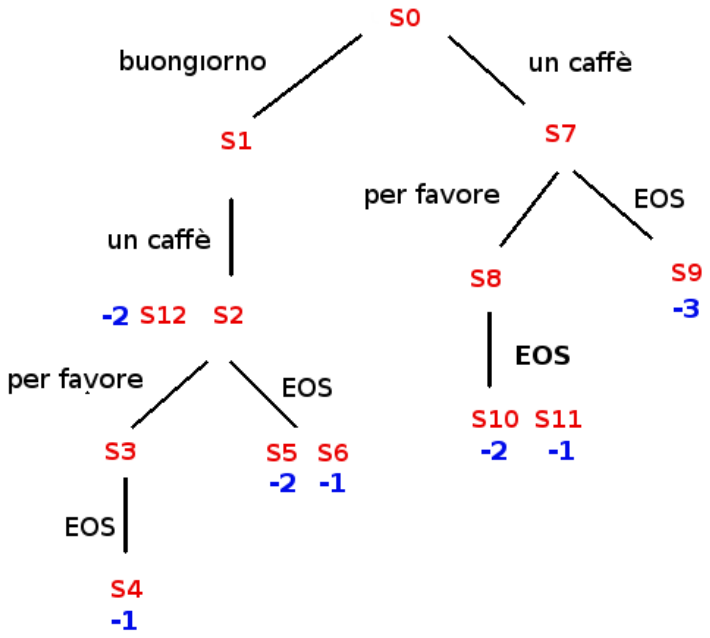


Figure 01: States diagram for actions and rewards without free coffee

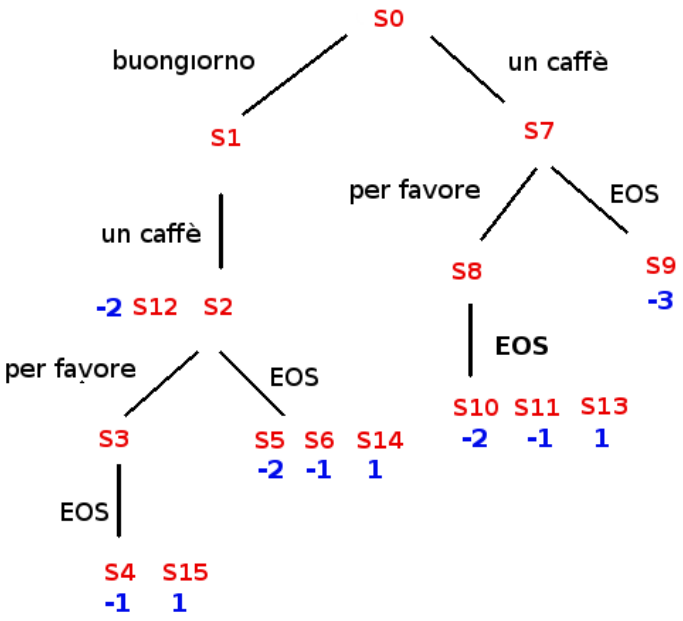


Figure 02: States diagram for actions and rewards with free coffee

Actions:

Action 1: Say "Buongiorno, un caffè per favore" *(most polite)*
Action 2: Say "Buongiorno, un caffè" *(polite)*
Action 3: Say "Un caffè, per favore" *(not very polite)*
Action 4: Say "Un caffè" *(impolite)*

Rewards:

In Case 1:

Action 1a: 1 for 0 EUR *(most polite, café owner is in a good mood, offers free coffee)*
Action 1b: -1 for 1 EUR *(most polite, café owner is in a good mood)*
Action 2a: 1 for 0 EUR *(polite, café owner is in a good mood, offers free coffee)*
Action 2b: -1 for 1 EUR *(polite, café owner is in a good mood)*
Action 3a: 1 for 0 EUR *(not very polite, café owner is in a good mood, offers free coffee)*
Action 3b: -1 for 1 EUR *(not very polite, café owner is in a good mood)*
Action 4: -3 for 3 EUR *(impolite, café owner is in a good mood)*

In Case 2:

Action 1: -2 for 2 EUR *(most polite, café owner is busy)*
Action 2: -3 for 3 EUR *(polite, café owner is busy)*
Action 3: -2 for 2 EUR *(not very polite, café owner is busy)*
Action 4: -3 for 3 EUR *(impolite, café owner is busy)*

The agent's goal is to learn the Q-values for each state-action pair, which represent the expected future reward for taking a specific action in a specific state.

3. Experimental setup

The environment is defined as a dictionary, where the keys are the states and the values are the possible actions that can be taken from that state and the corresponding next state, reward and probability as we can see in Figure 03.

```
# First int is state we're transitioning to,
# second the reward,
# third probability of transitioning to that state
environment = {
    0: [('buongiorno',[[1,0,1]]),('un caffè',[[7,0,1]])],
    1: [('un caffè',[[2,0,0.8],[12,-2,0.2]])],
    2: [('per favore',[[3,0,1]]),('EOS',[[5,-2,0.4],[6,-1,0.4],[14,1,0.2]])],
    3: [('EOS',[[4,-1,0.8], [15,-1,0.2]])],
    7: [('per favore',[[8,0,1]]),('EOS',[[9,-3,1]])],
    8: [('EOS',[[10,-2,0.7],[11,-1,0.2], [13,1,0.1]])]
}
```

Figure 03: Environment setup with free coffee

The Q-matrix is initialized as a 2D list with the same number of rows as states and 4 columns corresponding to the number of actions.

The agent starts in a random initial state, and at each step, it selects an action randomly from the possible actions.

The agent then transitions to the next state based on the selected action, and the corresponding reward is obtained.

The Q-value for the current state-action pair is updated using the Q-learning update rule:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma max(Q(s',a')) - Q(s,a))$$

The agent continues to take actions and update its Q-matrix until it reaches an exit state.

The Q-matrix was initialized with zeros, one for each state-action pair. The agent then interacts with the environment by taking random actions and updating the Q-matrix using the Q-learning algorithm. The agent goes through 1000 episodes, each episode representing one interaction with the environment. The learning rate and discount factor were set to 0.01 and 0.99 respectively. In total, we are running our simulation for the following setups:

Setup 01: *discount factor = 0.9, learning rate = 0.01, episodes = 1000*

Setup 02: *discount factor = 0.99, learning rate = 0.1, episodes = 1000*

Setup 03: *discount factor = 0.95, learning rate = 0.05, episodes = 5000*

Setup 04: *discount factor = 0.8, learning rate = 0.01, episodes = 10000*

Setup 05: *discount factor = 0.99, learning rate = 0.001, episodes = 100000*

4. Results

The agent was able to learn the most efficient way to obtain a coffee with minimal cost through the use of a Q-matrix, which converged to optimal values after multiple iterations. However, the agent's behavior was not always desirable due to the randomness in the transitions and rewards, which could lead to suboptimal behavior. The study was conducted in five different scenarios as outlined in Section 3, and the average distance between the final Q-matrix and the optimal Q-matrix was calculated for each. The evaluation results can be found in Table 1.

Setup No.	**Discount Factor**	**Learning Rate**	**Episodes**	**Distance (Error)**
01	0.9	0.01	1000	6.098
02	0.99	0.1	1000	7.125
03	0.95	0.05	5000	7.807
04	0.8	0.01	10000	6.078
05	0.99	0.001	100000	8.022

Table 01: Evaluation for different setups

(Total difference between the final Q-matrix and the optimal Q-matrix)

5. Discussion

The agent's behavior may not always be polite due to the randomness in the selection of actions and the low learning rate. The code uses randomness to choose the next action, instead of always choosing the action with the highest estimated Q-value. This means that even though the agent has learned that using polite language leads to lower costs, it may still take impolite actions with some probability. The low learning rate parameter also makes it harder for the agent to learn the optimal policy. The agent is only trained for 5 different scenarios, which may not be enough for it to fully converge to the optimal policy. Among all the scenarios, scenario 04 performed the best. To improve the agent's politeness, you could try increasing the number of episodes, increasing the learning rate, or implementing a more advanced action selection strategy such as an epsilon-greedy strategy.

One of the ways to improve the agent's politeness is to increase the rewards for polite behavior and decrease the rewards for impolite behavior. Additionally, we can also try changing the probability of transitions for certain states, for example, increasing the probability of transitioning to a state where the agent receives a free coffee when it uses a polite phrase. Another way to improve the agent's behavior is by introducing a penalty for impolite behavior. The agent should be penalized for using impolite phrases, which will make it less likely to take those actions in the future. Furthermore, adding more states and actions to the environment can also help the agent learn more nuanced and sophisticated politeness rules. This can make the agent more versatile and adaptable to different situations.

In conclusion, the RL agent was able to learn the optimal sequence of actions that resulted in the lowest cost for a coffee. However, there is still room for improvement, particularly in terms of the agent's politeness. By adjusting the rewards, transitions, and states, we can make the agent more polite and sophisticated in its behavior.

6. Appendix

[1] **Github** link for the code:

https://github.com/khandakerrahin/introduction_to_ML_for_NLP

[2] https://github.com/ml-for-nlp/reinforcement-learning

[3]

https://github.com/simoninithomas/Deep_reinforcement_learning_Course/blob/master/Q_Learning_with_FrozenLakev2.ipynb

[4] https://gymnasium.farama.org/

[5] https://simoninithomas.github.io/deep-rl-course/