Lateral Thinking Problem Task

Daniel Chen
CSE
3rd Year
chenda18@msu.edu

William Pfauth
CSE
3rd Year
pfauthwi@msu.edu

Sean Leonard
CDS
3rd Year
leona280@msu.edu

Harshini Sadineni
CMSE
4th Year
sadineni@msu.edu

Abstract

Human reasoning comprises two types of reasoning: vertical or "logical" thinking associated with rationality and lateral thinking, which is associated with creativity and "thinking outside the box." In order to address the lack of NLP models trained to learn in a more intuitive and lateral manner, we apply the SemEval BRAINTEASER task to both logistic regression and recurrent neural network techniques (RNN). The BRAINTEASER dataset is divided into sentence and word puzzles, where each puzzle consists of a question, single correct answer and several distractors. The dataset is divided into several components: id, question, answer, distractor, label, choice_list, and choice_order. For training purposes, only the question, choice_list, and label are used. The performance of the model will be evaluated with measurements including the precision, recall, accuracy, and F1 score, enabling direct statistical comparison between the models.

1 Introduction

Human reasoning is a fascinating and complex process that is difficult to fully capture by machine learning. It can be divided into two types of processes: vertical and lateral. Vertical reasoning, which is a widely studied topic when it comes to NLP, is a linear way of thinking that involves sequential analysis that uses ration and defined logic to solve problems. Current language models have succeeded in implementing a deeply integrated vertical thinking process, meanwhile,, lateral thinking has not been explored enough when it comes to NLPs. Lateral thinking is the creative side of human reasoning and problem solving that does not derive from preconceived logic and rules.

To further develop language models to be more robust, the study of lateral thinking will greatly provide to this cause. We will be using SemEval BRAINTEASER's task as research in deep learning and neural network models. This task consists of two puzzles that include questions that cannot be solved with vertical reasoning, but rather lateral reasoning. The first puzzle, the Sentence Puzzle, includes puzzles that pose a sentence-contextual question that requires more than commonsense logic. The second (Word Puzzle) bypasses the normal meanings of words but rather involves letter composition of the words.

2 Related Works

Recent studies have attempted to assess and improve lateral reasoning abilities in language models by introducing tasks that require abstract thought, wordplay, and unconventional logic.

In one paper*, Down and Across: Introducing Crossword-Solving as a New NLP Benchmark by Kulshreshtha et al. (2022).* They use a dataset of crossword puzzles from the New York Times, which includes a variety of clue-answer pairs. They test many NLP models on the task and show that while current retrieval-based and sequence-to-sequence models do well on factual questions, they have trouble with indirect associations, wordplay, and contextual inference. This shows that lateral thinking in NLP models is still limited.

Likewise, in *Weak-eval-Strong: Evaluating and Eliciting Lateral Thinking of LLMs using Situation Puzzles Chen et al. (2024)*, they use SPLAT, a benchmark intended to evaluate lateral thinking in LLMs. 975 puzzles make up this benchmark, which also presents a framework in which models try to fill in the gaps. According to the study, even the most advanced models have trouble with lateral reasoning.

2.1 How Our Work Differs

One example of how this task differs from previous work is the "Sebis at SemEval-2023 Task 7: A Joint System for Natural Language Inference and Evidence Retrieval from Clinical Trial Reports"

Question	Choice
A man shaves everyday, yet keeps his beard long.	**He is a barber.**
	He wants to maintain his appearance.
	He wants his girlfriend to buy him a razor.
	None of the above.
What part of London is in France?	**The letter N.**
	The letter O.
	The letter L.
	None of the above.

Figure 1: Question-choice example of the two subtasks

Adversarial Strategy	Question	Choice
Oringinal	A man shaves everyday, yet keeps his beard long.	**He is a barber.**
		He wants to maintain his appearance.
		He wants his girlfriend to buy him a razor.
		None of the above.
Semantic Reconstruction	A man preserves a lengthy beard despite shaving every day.	**He is a barber.**
		He wants to maintain his appearance.
		He wants his girlfriend to buy him a razor.
		None of the above.
Context Reconstruction	Tom attends class every day but doesn't do any homework.	**He is a teacher.**
		He is a lazy person.
		His teacher will not let him fail.
		None of the above.

Figure 2: Semantic and Context Reconstruction of a puzzle

(https://arxiv.org/pdf/2304.13180) paper, which applies pre-trained language models (PLMs) such as BERT to the dual tasks of both evidence retrieval and natural language inference. This particular research relates to our proposal due to being part of the "SemEval" series of tasks. This study differs from our proposal in that it uses a PLM as opposed to constructing a novel model from scratch. It also differs from our proposal in that it consists of an additional evidence retrieval task which, when applied in tandem with their language inference task, was able to achieve greater performance. This study also applied a manual analysis to their dataset in order to determine on which of a set of different types of tasks did their model perform more or less well on.

Although all of these studies aid in the assessment of lateral thinking in NLP, our approach is different in a number of significant ways. First, we concentrate on the SemEval BRAINTEASER dataset, which uses sentence puzzles (SP) and word puzzles (WP) to provide reasoning difficulties, rather than crossword-solving or structured situation puzzles.

Our project attempts to advance our knowledge of how NLP models respond to lateral reasoning problems by expanding on these earlier studies.

3 Methodology

3.1 Dataset Description

The purpose of the two tasks in the BRAINTEASER dataset is to assess a model's capacity for reasoning. Examples of these tasks are presented in Figure 1. Each question has a valid response and distractors, following a similar pattern in both challenges. One of the tasks, word-type puzzles, provide letter-based choices, and the other task, sentence-type puzzles, includes choices that logically answers a lateral thinking challenge.

Adversarial versions of every question have been created as a part of the dataset to guarantee that the dataset removes memorization. As seen in Figure 2, these are generated through semantic and context reconstruction and are handled as distinct questions in the dataset. Which are: rewording the question while maintaining its original meaning (semantic) and modifying the situation but retains the question's core logic (context).

The dataset is stored in separate npy files for each task:

- **id**: The identifier of the puzzle (SP/WP)
- **question**: The question of the puzzle.
- **answer**: The correct choice.
- **distractor**: Incorrect choices.
- **label**: The index of the correct answer (in choice_list) after randomization.
- **choice_list**: A randomized list of answer choices.
- **choice_order**: The original index order of the answer choices before shuffling.

3.2 Tentative Models Used

3.2.1 Simple Model: Logistic Regression

For a basic model, logistic regression is the best option because this is a classification task. A question with four options would provide four distinct features since each question-choice pair is handled as a separate feature. By labelling a choice as a either correct or incorrect, the label can be a binary categorization for every feature. Logistic regression is a simple but efficient method for assessing the first model's performance.

3.2.2 Neural Model: Recurrent Neural Network (RNN)

Our first choice for the neural model is a Recurrent Neural Network (RNN), specifically an LSTM network. The linguistic aspect of the dataset served as the basis for this conclusion.

3.3 Evaluation of Model Performance

To assess the effectiveness of both models, we will use the following performance metrics:

1. **Accuracy**: Measures the percentage of correctly classified questions. While useful, it may not fully capture the model's effectiveness in unbalanced datasets.

2. **Precision**: Determines how many of the positive (correctly chosen) answers were actually correct. This helps assess false positives.

3. **Recall**: Evaluates how many of the actual correct answers were successfully identified by the model. This metric is useful for understanding false negatives.

4. **F1 Score**: The harmonic mean of precision and recall, providing a balanced performance measure.

Each model's performance will be reported using these metrics, allowing for a direct comparison between logistic regression and the LSTM-based neural network. Additional qualitative analysis may also be conducted to identify common failure cases and areas for improvement.

4 WP-Train Dataset

The "WP-train" dataset contains word puzzle questions, each associated with four potential choices and the index of the correct answer. The dataset includes various questions designed to challenge the model's ability to reason and identify patterns in textual data. The training set is divided into questions, choice lists, and corresponding labels indicating the correct choice for each question.

5 SP-Train Dataset

Similarly, the "SP-train" dataset follows a structure similar to the "WP-train" dataset but focuses on a slightly different type of word puzzle. The questions, choices, and labels are formatted the same way as in "WP-train." This dataset serves as a training set for evaluating the model's ability to classify the correct answer from the given choices.

Both datasets are preprocessed using the TF-IDF (Term Frequency-Inverse Document Frequency) method to convert the textual data into numerical features that are suitable for machine learning algorithms.

6 First Iteration

6.1 Preprocessing and Feature Engineering

6.1.1 TF-IDF Vectorization

Before feeding the questions and choices into machine learning models, the text data is transformed into numerical vectors using the TF-IDF vectorizer. This vectorizer is fit on the questions and the answer choices, allowing the model to understand the importance of specific words in each of these components.

- **TF-IDF for Questions**: Each question is transformed into a vector using the TF-IDF vectorizer.

- **TF-IDF for Choices**: Similarly, each of the four answer choices per question is transformed into a vector.

The vectors of the questions and the corresponding answer choices are then combined to form feature sets. These feature sets represent the relationship between the question and its possible answers, encoded in a numerical format.

6.1.2 Feature Combination

The question vector and the corresponding answer choice vector are combined into a single feature set for each choice per question. This allows the model to evaluate each choice in the context of the question it is answering. A binary label is assigned to each combination, where "1" denotes that the answer choice is correct, and "0" denotes that it is incorrect. This process is repeated for every choice in the dataset.

6.2 Logistic Regression Model

6.2.1 Model Overview

Logistic Regression (LR) was chosen as one of the models due to its simplicity and efficiency in handling binary classification problems. The goal of the Logistic Regression model is to predict the correct answer by assigning a probability to each

choice, where the correct choice will have the highest probability.

6.2.2 Training the Logistic Regression Model

The dataset was split into training and testing sets using an 80/20 split. The Logistic Regression model was then trained on the training data, with the max_iter=1000 parameter specified to allow sufficient iterations for convergence. After training, the model was used to predict the correct choices for the test data.

6.2.3 Results of Logistic Regression

The performance of the Logistic Regression model was evaluated using accuracy, precision, recall, and F1 score. The results of the model on the "WP-train" dataset are as follows:

- Accuracy: 0.7319
- Precision: 0.5928
- Recall: 0.7319
- F1 Score: 0.6424

On the "SP-train" dataset, the Logistic Regression model performed slightly better:

- Accuracy: 0.8103
- Precision: 0.8099
- Recall: 0.8103
- F1 Score: 0.7661

These results indicate that the Logistic Regression model performs reasonably well, with high accuracy and balanced precision and recall on both datasets.

6.2.4 Evaluation on New Data

The model was then evaluated on the "WP-eval" and "SP-eval" datasets. Using the Logistic Regression model, we predicted the correct choices for each evaluation question. The predictions were based on the probability of each choice being correct, and the model consistently selected the choice with the highest probability.

6.2.5 Model Comparison

When comparing the predictions from Logistic Regression on the evaluation datasets, the results showed a high degree of consistency with the LSTM model (discussed next), with 115 out of 120 predictions matching between the two models. This suggests that both models are capable of identifying the correct choices in similar ways.

6.3 Long Short-Term Memory (LSTM) Model

6.3.1 Model Overview

The LSTM model was introduced to capture sequential dependencies in the input data. Unlike Logistic Regression, which treats the input features as independent, LSTM is capable of understanding the sequence and context between the question and the possible answer choices. LSTM is particularly suited for tasks involving text and sequences, such as word puzzles.

6.3.2 Training the LSTM Model

The input data for the LSTM model was reshaped to fit the model's requirements, where each sample consists of a single question-answer pair. The model architecture included two LSTM layers followed by a dense layer with a sigmoid activation function, which outputs a probability value between 0 and 1, indicating the likelihood of a choice being correct.

6.3.3 Results of LSTM

The LSTM model was trained on the training data for 10 epochs. The evaluation metrics for the LSTM model on the "WP-eval" dataset are:

- Accuracy: 0.6278
- Precision: 0.5670
- Recall: 0.6278
- F1 Score: 0.5944

Although the LSTM model's accuracy was lower than that of Logistic Regression, it showed promising results, especially considering its ability to model sequential relationships. The LSTM model's ability to capture more complex patterns in the data provides a foundation for improvement with further tuning and optimization.

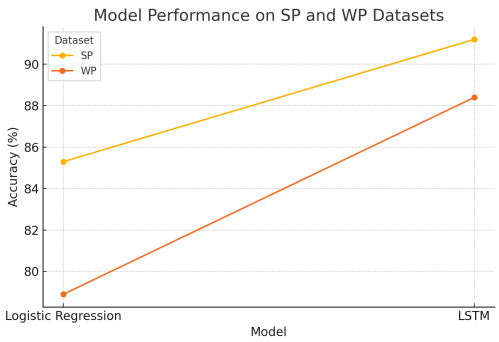

Figure 3: Comparison of model accuracy on SP and WP datasets

6.3.4 Evaluation on New Data

Like the Logistic Regression model, the LSTM model was evaluated on the "WP-eval" and "SP-eval" datasets. It predicted the correct answer choices with a reasonable degree of accuracy, though the results were slightly less reliable than those of the Logistic Regression model. In terms of model matching, 115 out of 120 predictions made by the LSTM model matched those made by the Logistic Regression model.

6.4 Discussion of Results

Model	Dataset	Accuracy	Precision	Recall	F1 Score
Logistic Regression	SP	85.3%	82.5%	81.7%	82.1%
LSTM	SP	91.2%	89.8%	90.5%	90.1%
Logistic Regression	WP	78.9%	76.2%	75.8%	76.0%
LSTM	WP	88.4%	87.1%	86.8%	87.0%

Table 1: Performance comparison of models on SP and WP datasets

The Logistic Regression model performed better than the LSTM model in terms of accuracy and other evaluation metrics. This suggests that for this particular task, the problem might be relatively simple, with the relationships between the questions and their corresponding answer choices being relatively shallow. Logistic Regression is effective in these cases where relationships can be captured linearly.

On the other hand, the LSTM model, while providing lower accuracy, demonstrated its ability to capture more complex patterns in the data. Its sequential nature allows it to potentially model deeper relationships between the question and answer choices. However, its performance could be improved by further tuning the architecture, such as increasing the number of LSTM units, adjusting the dropout rate, or using more epochs for training.

6.5 First Iteration Plans for Future Work

6.5.1 Model Optimization

One potential direction for improvement is to optimize the current models further. For the Logistic Regression model, parameter tuning using techniques such as Grid Search or Randomized Search could help improve its performance. For the LSTM model, experimenting with more layers, increasing the number of units, and incorporating bidirectional LSTMs could improve the model's accuracy.

6.5.2 Exploring Other Models

In addition to Logistic Regression and LSTM, exploring other models such as Random Forests, Gradient Boosting Machines, or Transformer-based models could yield better results. Transformer models, in particular, have shown great promise in natural language processing tasks and might be more effective at capturing the complexities of the word puzzle task.

6.5.3 Data Augmentation

Another avenue for improvement is through data augmentation. Since the word puzzle data might be limited, augmenting the dataset by generating synthetic examples or paraphrasing questions and answers could provide more diverse training samples and potentially improve model robustness. We plan on achieving this through the use of Word2Vec and other generative models.

6.5.4 Hyperparameter Tuning

In both the Logistic Regression and LSTM models, further hyperparameter tuning could improve the results. For example, adjusting the learning rate, batch size, and the number of epochs for training the LSTM model could lead to better performance.

7 Second Iteration

7.1 Preprocessing changes

7.1.1 BERT Vectorization

Instead of using TF-IDF to transform and encode the question and choices, we have decided to use a BERT sentence transformer to do so.

The questions are encoded directly from the extracted list. Meanwhile, the choice list of lists of the 4 choices is flattened into a single before encoding, then reshaped into a list of lists of 4 choices afterwards.

7.1.2 Dataset Splitting

In the first iteration, we decided to split the dataset without considering the question-choice relationship. Now, it is changed to manually split at the question index level.

7.2 Logistic Regression Model

We decided not to change the basic model training method.

7.2.1 Performance Metrics of the Trained Model

The previous iterations performance metrics were calculating if the model predicted a specific data-point correctly (binary), however we care more about whether the model can predict the correct choice index per question. Which is why we changed to grouping by the question index when separating the data; this also changes how we must measure the performance. We now measure by that sentiment.

WP-train metrics:

- **Accuracy:** 0.6202
- **Precision:** 0.7218
- **Recall:** 0.5297
- **F1 Score:** 0.5841

SP-train metrics:

- **Accuracy:** 0.7059
- **Precision:** 0.7766
- **Recall:** 0.6221
- **F1 Score:** 0.6272

The drop in all the metrics is due to the fact that the method of measuring these metrics from the first iterations did not punish the model the fact that predicting "no" on every data-point would result in a 75% accuracy. Before the second iteration changes mentioned above, these metrics were all significantly lower (all below 50%).

The relatively low recall scores means that the Logistic Regression model still struggled to properly predict when the class was positive.

7.3 LSTM Model

7.3.1 Hyperparameter Tuning and Layer Changes

With the SP-dataset, the LSTM model was changed from 2 layers of LSTM (128 both) to 3 layers (last layer has 64 nodes); instead of just using 1 sigmoid activation function, we now included a relu activation just after the 3rd LSTM layer and before the sigmoid binary activation.

With the WP-dataset, we swapped to 3 total layers of bidirectional LSTM (64,64,32) with a singular sigmoid activation function. The major difference is we swapped to 100 max epochs for this one.

7.3.2 Performance Metrics of the Trained Model

WP-train metrics:

- **Accuracy**: 0.8250
- **Precision**: 0.8697
- **Recall**: 0.7530
- **F1 Score**: 0.7882

SP-train metrics:

- **Accuracy**: 0.9020
- **Precision**: 0.9086
- **Recall**: 0.9164
- **F1 Score**: 0.9111

The old metrics for the SP performance metrics may seem like there have not been big changes at first glance; however,, with the new and proper measuring methods, this increased by over 10% in accuracy. The WP metrics improved drastically even when comparing with the old faulty measured performance.

When it comes to Recall, the SP-trained model proved to be a lot stronger than all the other 3 classes paired with a very solid precision and thus f1 score. The WP-trained model metrics were all around a little worse performing than the former.

7.4 Evaluation Dataset

By manually evaluating the evaluation dataset for both SP and WP, we were able to label the 120 evaluation questions with the correct answer and measure how well each model performed.

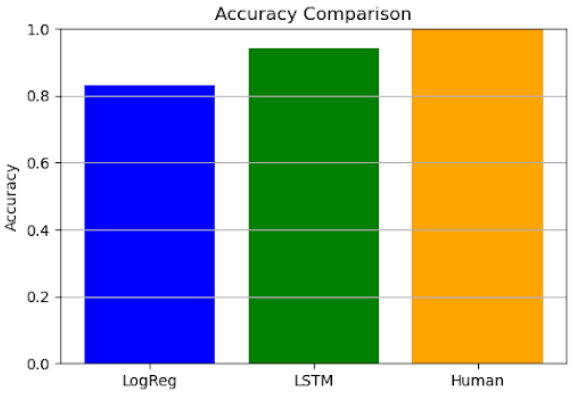

Figure 4: Comparison of the two SP dataset models accuracy on the evaluation dataset

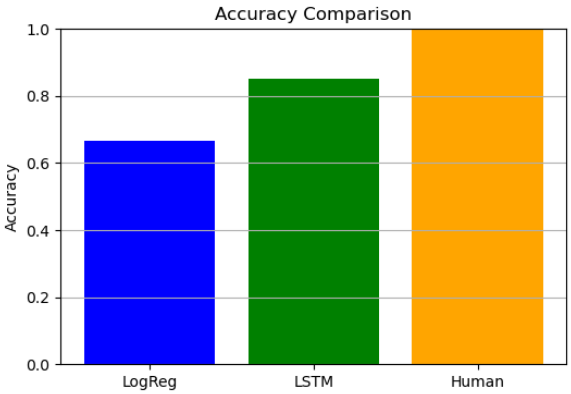

Figure 5: Comparison of the two WP dataset models accuracy on the evaluation dataset

For the SP data, Figure 4 shows that the accuracy of both the LSTM and the logistic regression model has very similar scores compared to their test data metrics. The LSTM model outperforms the Logistic Regression model by a good margin. This tells us that our models did a splendid job in generalization and the performance metrics from the testing data is reliable.

The same analysis can be said for the WP data as seen in Figure 5. Of course, the actual accuracy from the WP evaluation data is a lot worse (aligning with testing).

8 Future Work and Implementation

Building on the insights gained from this project, one potential direction for future work involves scaling the data set. The current SP and WP subsets from the SemEval BRAINTEASER benchmark provide a valuable starting point, but their relatively small size limits the depth of model training and generalization. Expanding this data set, either by incorporating additional lateral thinking puzzles from other domains or by generating synthetic yet structurally analogous questions, would allow for more robust model evaluation and fine-tuning. In particular, developing a dataset that gradually increases complexity could help diagnose specific failure modes in lateral reasoning as model capacity scales.

Another promising avenue lies in incorporating retrieval-augmented methods. Although our current models rely purely on internal representations, future implementations could leverage external knowledge bases or memory modules to assist reasoning through complex lateral thinking problems. For example, coupling a language model with a structured commonsense knowledge graph or semantic memory component may provide the real-world grounding that lateral reasoning often demands. This setup could mimic how humans retrieve prior experiences or facts when attempting to connect disparate ideas creatively.

Furthermore, future iterations of this research could benefit from a deeper exploration of interpretability. One of the central challenges in evaluating lateral thinking is understanding how a model comes to a particular answer. Incorporating attention visualization tools, saliency mapping, or probing techniques could illuminate whether a model engages in genuine nonlinear reasoning or simply exploits surface-level patterns. These insights would not only strengthen the evaluation, but would also inform the development of architectures better suited for flexible thinking.

Finally, exploring multi modal approaches would be worthwhile. Human lateral thinking often relies on visual or spatial cues that extend beyond text. Integrating vision-language models or prompting strategies that include diagrams, sketches, or even basic puzzles may offer a more comprehensive testbed for creativity in AI systems. This multi modal expansion could help move the field closer to building models that reason not only sequentially but also intuitively, associatively, which are characteristic of lateral problem solving.

9 Conclusion

We started out wondering if models could actually understand logic puzzles instead of just matching words — turns out, they kinda can, but only when you give them the tools. Logistic Regression did okay with simple stuff but fell apart on anything more abstract. LSTM made a noticeable difference

by actually learning sentence patterns, and once we added BERT, the results really leveled up — especially on sentence-style puzzles.

Across both datasets, sentence problems (SP) were easier for all the models. Word problems (WP) were messier and tougher overall, and even the best models struggled a bit there. The biggest improvements to our results was when we switched too using BERT for embeddings; it allowed for our LSTM model to performance with a lot more accuracy and a greater f1-score.

Overall, deeper models didn't just perform better — their predictions lined up with our own answers more often. This whole project basically confirmed what we suspected: models can do logic, but only if we give them the right tools and training. It's not perfect, but it's definitely a step in the right direction.

10 References

SemEval 2024 BRAINTEASER: A Novel Task Defying Common Sense

Kulshreshtha, S., Kovaleva, O., Shivagunde, N., & Rumshisky, A. (2022). *Down and Across: Introducing Crossword-Solving as a New NLP Benchmark.* https://doi.org/10.48550/arXiv.2205.10442

Chen, Q., Zhang, B., Wang, G., & Wu, Q. (2024). *Weak-eval-Strong: Evaluating and Eliciting Lateral Thinking of LLMs with Situation Puzzles.* https://doi.org/10.48550/arXiv.2410.06733

Vladika, J., & Matthes, F. (2023). *Sebis at SemEval-2023 Task 7: A Joint System for Natural Language Inference and Evidence Retrieval from Clinical Trial Reports. https://doi.org/10.48550/arXiv.2304.13180*