Minimal Clues for Maximal Understanding: Solving Linguistic Puzzles with RNNs, Transformers, and LLMs

Stanford CS224N Custom Project

Ishan Khare
Department of Computer Science
Stanford University
iskhare@stanford.edu

Anavi Baddepudi
Department of Computer Science
Stanford University
anavib@stanford.edu

Emma Wang
Department of Computer Science
Hasso Plattner Institute of Design
Stanford University
emmwang@stanford.edu

Abstract

There is a critical gap in the ability of current deep learning models to mimic human-like reasoning and understand complex language phenomena from minimal data. To help understand these limitations, we apply a variety of modern deep-learning approaches to solve puzzles from the Linguistic Olympiads, which are designed to be solvable with minimal data and without the need for external linguistic knowledge. In particular, we implement three main approaches: i) RNNs with Attention, ii) finetuning pretrained transformer-based models, and iii) in-context learning with GPT-4. All three approaches are significant improvements from our baselines (BLEU scores ≈ 0.09), with the best performances achieved by transformer finetuning and GPT-4, which have BLEU-scores of 0.387 and 0.420, respectively. Perfect solutions achieve BLEU scores of 1.0, highlighting a large gap between the best humans and current state of the art models. We hypothesize that meta-learning approaches are essential to make further progress in solving linguistic puzzles from small data.

Key Information

Our 224N mentor is Tathagat Verma. All team members contributed equally at all stages *i.e.,* ideation, data collection, code-writing, experiments, analysis, and paper writing.

1 Introduction

Linguistic puzzles, especially those involving low-resource languages, pose a distinctive challenge within the field of computational linguistics. This study delves into "Rosetta Stone" puzzles, which require deciphering translations of sentences between a source and target language, utilizing only a minimal set of bidirectional translations available, as seen in figure 7. These puzzles are designed to be solvable without prior knowledge of the languages involved, leveraging the solver's ability to deduce linguistic frameworks and apply them effectively. Our research aims to address these puzzles computationally, evaluating various models to identify the most effective approach for solving such puzzles with limited data.

Our investigation seeks to understand the existing gap in computational linguistics by conducting a series of experiments to explore different methodologies for solving these puzzles with minimal data input. In their work, Magueresse et al. (2020) raise concerns that current NLP research predominantly concentrates on merely 20 of the $7,000$ known languages, leaving the vast majority of languages understudied. Echoing

these concerns, our work aspires to extend the focus to the largely overlooked linguistic diversity. Our results could be beneficial for translation tasks in crucial fields such as healthcare and education where being able to translate native, underrepresented language understanding could significantly impact outcomes. In this manner, we hope to extend the reach of NLP technologies to encompass a broader array of languages, while also highlighting the capabilities and limitations of current translations models when working with small datasets.

We experimented with three distinct approaches: i) recurrent neural network (RNN) models with attention, ii) finetuning pre-trained transformer-based neural machine translation (NMT) models, and iii) in-context learning with the GPT-4 LLM (OpenAI et al., 2024). This three-fold approach aims to span the spectrum of computational models and provide a comprehensive analysis of their strengths and limitations in addressing the challenges presented by Rosetta Stone puzzles. Although all three of our methods outperform the baselines, our results from sections 4.4 and 5 highlight a gap between current deep-learning and LLM approaches and the human-like reasoning and understanding necessary to solve these puzzles.

2 Related Work

For the specific application of small-data translations on linguistic puzzles, a notable contribution has been made by the Ubiquitous Knowledge Processing Lab (UKP) (Şahin et al., 2020). The authors use Moses (Koehn and Knowles, 2017) with default settings to train a 5-gram English LM, and then a RoBERTa base, where they tune the Byte Pair Encoding (BPE) merge parameter, learning rate and number of epochs. The UKP team's findings show the limitations of relying solely on default model configurations. Moreover, this study pushes for a shift towards models that can more closely mirror human cognitive processes, and highlights issues with current models in understanding the nuance of linguistic structures and semantics beyond the patterns and associations found within large datasets.

Although this paper sheds light on the areas where the existing models fall short, it is limited in its lack of in-depth error analysis for the models compared to the simple baselines, and performs manual error analysis for two of the models in total. Additionally, this paper emphasizes the dataset creation more than insight into model performance and limitations. Furthermore, this paper was written before the advent of more recent models – notably, LLMs.

On a more general level, broader research in the field of machine learning for low-resource languages has been advancing. Maillard et al. (2023) discusses the current state of machine learning for low-resource languages, introducing techniques like self-supervised learning and back translation to reduce dependency of large annotated datasets. The study suggests that small amounts of high-quality, translated data can improve machine translation performance using these techniques.

These collective efforts show a pivotal shift towards developing machine translation models capable of achieving higher accuracy and understanding with limited data. While the UKP study presents a critical look at the limitations of current models, the research by Maillard et al. (2023) offers a hopeful perspective on overcoming these barriers, marking a significant step towards making more efficient and effective translation models.

3 Approach and Methodology

3.1 Baselines

We implemented two baselines for our evaluation of the translation tasks. The first is a *Random Words* baseline for the translation task, which is essential for gauging the efficacy of more advanced translation methodologies. It involves generating translations by randomly selecting a word from the target language's lexicon for each token in the source text. This method establishes a basic performance level that any effective model must exceed.

When applied to our linguistic puzzle data, the *Random Words* method achieved an average BLEU (Papineni et al., 2002) score of 0.083. The second baseline employs the translation alignment tool *FastAlign* (Dyer et al., 2013). Since *FastAlign* produces alignments for each training pair, we choose to post-process the output to create a translation dictionary separately for each direction. We then randomly choose from the translation entries for each token in the source test sentence. This baseline has similar performance to *Random Words*, with a BLEU score of 0.096.

3.2 Recurrent Neural Network (RNN)

We build a Neural Machine Translation (NMT) system by implementing a sequence-to-sequence (Seq2Seq) network with multiplicative attention (Luong et al., 2015). In addition to the following explanation, our Seq2Seq Model is depicted in Figure 1.[1]

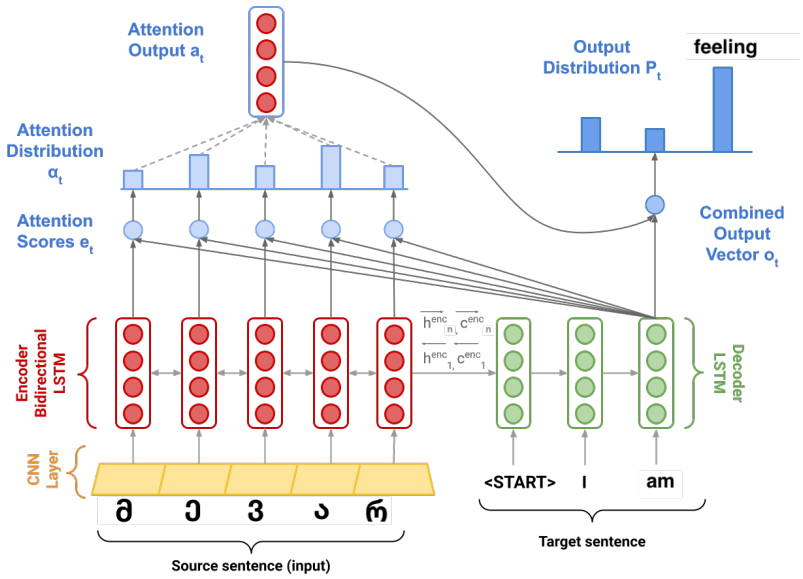

Figure 1: Model architecture of our Seq2Seq Model with Multiplicative Attention

First, we obtain the character or word embeddings from an embeddings matrix, which gives us $x_1, \ldots, x_n$ ($x_i \in \mathbb{R}^e$), where n is the length of the source sentence and e is the embedding size. We subsequently feed these embeddings to a convolutional layer f and ensure that the outputs maintain their shapes. Next, the outputs from the convolutional layer are given as input to the bidirectional LSTM encoder. This procedure yields hidden and cell states for both LSTMs (forwards and backwards), which we concatenate to get our hidden states hidden states $\mathbf{h}_i^{enc}$ and cell states $\mathbf{c}_i^{enc}$, as shown by equations 1 and 2:

$$\mathbf{h}_i^{\text{enc}} = [\overleftarrow{\mathbf{h}_i^{\text{enc}}}; \overrightarrow{\mathbf{h}_i^{\text{enc}}}] \text{ where } \mathbf{h}_i^{\text{enc}} \in \mathbb{R}^{2h \times 1}, \overleftarrow{\mathbf{h}_i^{\text{enc}}}, \overrightarrow{\mathbf{h}_i^{\text{enc}}} \in \mathbb{R}^{h \times 1} \qquad 1 \leq i \leq n \tag{1}$$

$$\mathbf{c}_i^{\text{enc}} = [\overleftarrow{\mathbf{c}_i^{\text{enc}}}; \overrightarrow{\mathbf{c}_i^{\text{enc}}}] \text{ where } \mathbf{c}_i^{\text{enc}} \in \mathbb{R}^{2h \times 1}, \overleftarrow{\mathbf{c}_i^{\text{enc}}}, \overrightarrow{\mathbf{c}_i^{\text{enc}}} \in \mathbb{R}^{h \times 1} \qquad 1 \leq i \leq n. \tag{2}$$

The next step is to initialize the decoder's first hidden and cell states using equations 3 and 4:

$$\mathbf{h}_0^{\text{dec}} = \mathbf{W}_h[\overleftarrow{\mathbf{h}_n^{\text{enc}}}; \overrightarrow{\mathbf{h}_1^{\text{enc}}}] \text{ where } \mathbf{h}_0^{\text{dec}} \in \mathbb{R}^{h \times 1}, \mathbf{W}_h \in \mathbb{R}^{h \times 2h} \tag{3}$$

$$\mathbf{c}_0^{\text{dec}} = \mathbf{W}_c[\overleftarrow{\mathbf{c}_n^{\text{enc}}}; \overrightarrow{\mathbf{c}_1^{\text{enc}}}] \text{ where } \mathbf{c}_0^{\text{dec}} \in \mathbb{R}^{h \times 1}, \mathbf{W}_c \in \mathbb{R}^{h \times 2h}. \tag{4}$$

The input to the decoder on the t^{th} step is the embedding to the t^{th} subword $\mathbf{y}_t \in \mathbb{R}^{e \times 1}$ concatenated with $\mathbf{o}_{t-1} \in \mathbb{R}^{h \times 1}$, where $\mathbf{o}_{t-1}$ is the combined output vector from the previous step. This concatenated result is represented as $\overline{\mathbf{y}_t} \in \mathbb{R}^{(e+h) \times 1}$ in the equation $\mathbf{h}_t^{\text{dec}}, \mathbf{c}_t^{\text{dec}} = \text{Decoder}(\overline{\mathbf{y}_t}, \mathbf{h}_{t-1}^{\text{dec}}, \mathbf{c}_{t-1}^{\text{dec}})$ where $\mathbf{h}_t^{\text{dec}} \in \mathbb{R}^{h \times 1}, \mathbf{c}_t^{\text{dec}} \in \mathbb{R}^{h \times 1}$. The remaining part of the model architecture is to compute multiplicative attention over $\mathbf{h}_1^{\text{enc}}, \ldots, \mathbf{h}_n^{\text{enc}}$ by using $\mathbf{h}_t^{\text{dec}}$ as in equations 5 and 6:

$$\mathbf{e}_{t,i} = (\mathbf{h}_t^{\text{dec}})^T \mathbf{W}_{\text{attProj}} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{e}_t \in \mathbb{R}^{n \times 1}, \mathbf{W}_{\text{attProj}} \in \mathbb{R}^{h \times 2h} \qquad 1 \leq i \leq n \tag{5}$$

$$\mathbf{a}_t = \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{a}_t \in \mathbb{R}^{2h \times 1}; \alpha_t = \text{softmax}(\mathbf{e}_t) \in \mathbb{R}^{n \times 1}. \tag{6}$$

[1] Figure adapted from CS 224N Assignment 4.

Note that $\mathbf{W}_{\text{attProj}}$ represents the attention projection in the above equations. To obtain the combined output vector introduced earlier, we concatenate the decoder hidden state with the attention output and pass this through a linear layer, tanh, and dropout. Thus, we have $\mathbf{u}_t = [\mathbf{h}_t^{\text{dec}}; \mathbf{a}_t]$ where $\mathbf{u}_t \in \mathbb{R}^{3h \times 1}$; $\mathbf{v}_t = \mathbf{W}_u \mathbf{u}_t$ where $\mathbf{v}_t \in \mathbb{R}^{h \times 1}, \mathbf{W}_u \in \mathbb{R}^{h \times 3h}$; and $\mathbf{o}_t = \text{dropout}(\tanh(\mathbf{v}_t))$ where $\mathbf{o}_t \in \mathbb{R}^{h \times 1}$. Now, we create a probability distribution $\mathbf{P}_t$ as described in the equation $\mathbf{P}_t = \text{softmax}(\mathbf{W}_{\text{vocab}}\mathbf{o}_t)$ where $\mathbf{P}_t \in \mathbb{R}^{V_t \times 1}, \mathbf{W}_{\text{vocab}} \in \mathbb{R}^{V_t \times h}; V_t = |\text{target vocab}|$. Thus, we can use the cross entropy loss $J_t(\theta) = \text{CrossEntropy}(\mathbf{P}_t, \mathbf{g}_t)$ between $\mathbf{P}_t$ and $\mathbf{g}_t$, where $\mathbf{g}_t$ is the one-hot vector of the target subword at timestep t to train the network.

3.3 Transformer Finetuning

We further implemented a series of pre-trained transformer-based neural machine translation (NMT) models known as OPUS-MT, with the models pre-trained on parallel corpora as described in Tiedemann and Thottingal (2020). These models, designed around the standard transformer architecture, feature an encoder and decoder mechanism enhanced with self-attention to analyze and interpret the significance and context of words within inputs. Each model incorporates six layers – each layer is equipped with eight attention heads, enables simultaneous focus on various elements of the input, facilitating nuanced translation processes. Specifically, we performed fine-tuning on three OPUS-MT models for translations from English to Swahili, Georgian, and Malay, respectively, leveraging their pre-trained capabilities to adapt to these specific language pairs.

There was no OPUS-MT model available for Norwegian to English. Hence, we chose to use and fine-tune on the NLLB-200 model for Norwegian as an alternative. Like OPUS-MT, NLLB-200 aims to provide high-quality machine translations using the transformer architecture and a reliance on self-attention mechanisms. NLLB-200 combines self-attention and cross-attention in an encoder-decoder configuration, contributing to its capture of semantic subtleties across a broad linguistic range and making it especially effective for low-resource languages. This model has extensive multilingual training, and was pre-trained on a massive, diverse corpus of texts from 202 languages, as described in Costa-jussà et al. (2022). NLLB-200 utilizes multi-head attention as well, which allows the model to focus on distinct aspects of the inputs simultaneously.

Each model was initially fine-tuned using the Adam Optimizer (Kingma and Ba, 2017) on a substantial external corpus of translations encompassing a wide range of translations. The model was then tested on the linguistic puzzles. Following this, for comparison, the models were subjected to a targeted fine-tuning process using a small, specific training set composed of linguistic puzzles. This step was designed to hone the models' proficiencies in the nuanced task of puzzle translation.

3.4 GPT-4 In-Context Learning

Using the in-context learning capabilities of GPT-4 (an example of a prominent Large Language Model), we sought to analyze the performance of LLMs on the Rosetta Stone puzzle challenge. GPT-4 is characterized by its comprehensive training across a wide array of corpora that includes an extensive range of human knowledge and linguistic nuances. This background means that it can proficiently deduce linguistic rules/patterns and structures from specific inputs without explicit instructions. By providing the model with examples of translations within the puzzle and instructive prompts that mimic the structure and objectives of the puzzles, we primed the model's understanding and application of linguistic patterns and rules specific to the puzzle at hand. The goal of this approach was to simulate a learning environment where the model could draw on its extensive pre-training in combination with the provided context to generate accurate translations.

4 Experiments

4.1 Data

Our puzzle dataset is from the Linguistic Olympiad dataset and is solvable without previous understanding of the source languages. These languages are chosen to represent a broad spectrum geographic locations, thereby ensuring a comprehensive evaluation of our models across diverse linguistic contexts. We chose Swahili, Georgian, Norwegian and Malay, which come from different linguistic families as well as varying character representations and syntactical rules. Our experimental framework distinguishes between two primary datasets for training purposes: a $10,000$-sentence dataset ("large dataset") and a more constrained dataset ("small dataset"). It is important to note that our nominally large dataset is only $10,000$ sentences, whereas most language datasets consist of hundreds of thousands or millions of sentences.

The small dataset consists of the Olympiad linguistic puzzles provided in the PuzzLing Machines dataset by Şahin et al. (2020) for our four chosen languages. This dataset's training component features bidirectional translations provided within the Olympiad context. For evaluation, we included sentences from the Olympiad that had not been translated. Each language has a JSON file containing a dictionary with two keys: training and test data set. The training key has a list of completed puzzles, and the test data set has incomplete puzzles that require translations. We wrote a script for combining the JSON files from Linguistic Olympiad websites as well external files for Swahili (OzC, UK Linguistics Olympiad (2020)), Georgian, Malay and Norwegian (NAC, Vig (2019)), into larger files that contained all the "training data" and all the "test data" found into one JSON file per language.

The large training dataset, used for our RNN and transformer finetuning experiments, is taken from the TedTalk corpus Neu (2024). Due to their varying subject matter and accuracy in translations, we used the TedTalk corpus for the 4 languages to finetune the models. For each language's TedTalks corpus, we implemented a random division of data: allocating 80% to the training set, and dividing the remaining equally between the validation and test sets, with 10% for each.

4.2 Evaluation method

One automatic evaluation metric we can use is to compare the average BLEU score of the model's output when compared with the target answer. The BLEU score for candidate $\mathbf{c}$ with respect to $\mathbf{r}_1, \ldots, \mathbf{r}_k$ is: $BLEU = BP \times \exp\left(\sum_{n=1}^{4} \lambda_n \log p_n\right)$, where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights that sum to 1, $p_n = \dfrac{\sum_{\text{ngram}\in\mathbf{c}} \min\left(\max_{i=1,\ldots,k} \text{Count}_{\mathbf{r}_i}(\text{ngram}),\ \text{Count}_{\mathbf{c}}(\text{ngram})\right)}{\sum_{\text{ngram}\in\mathbf{c}} \text{Count}_{\mathbf{c}}(\text{ngram})}$, and $BP = \begin{cases} 1 & \text{if } len(c) \geq len(r) \\ \exp\left(1 - \frac{len(r)}{len(c)}\right) & \text{otherwise} \end{cases}$.

BLEU is the standard metric in machine translation, and it prevents the assignment of partial credit to subword matches, which may be especially relevant for foreign language targets. Other shortcomings may exist in that a system could achieve high BLEU scores by producing translations that mimic the reference but that lack natural language flow due to the method's dependence on n-gram matching. From this, we additionally utilized the characTER, chrF, and Exact Match metrics as well, which operate on the level of word pieces.

CharacTER (Wang et al., 2016) is derived from Translation Edit Rate (TER), where the edit rate is calculated as the minimum number of character edits required to adjust a hypothesis until it matches the reference, normalized by the length of the hypothesis sentence. CharacTER calculates the character level edit distance while performing the shift edit on the word level. ChrF (Popović, 2015) calculates the similarity between a machine translation and reference using character n-grams, not word n-grams – it is a simple F-measure reflecting the precision and recall of matching character n-grams. Lastly, we calculate the exact match score, which is a 1 if the hypothesis and reference sentences match and 0 otherwise.

4.3 Experimental details

For both the RNN and Transformer Finetuning experiments, we trained four models for each language for a total of 16 RNN models and 16 finetuned transformer models. More specifically, for each source language `SRC` we trained the following models: i) `SRC` to English on the large dataset, ii) `SRC` on the large dataset, iii) `SRC` to English on the small dataset, and iv) English to `SRC` on the small dataset.

All experiments were performed on NVIDIA A100 GPUs with 80GB RAM. With this high-performance compute, the training time for a single RNN model was shortened to about 16 minutes, on average. This gives a time of about 4.5 hours to train the final 16 RNN models. Furthermore, transformer finetuning took about 45 minutes per model for a training time of 12 hours. Thus, training for the RNN and transformer finetuning experiments collectively took about 16.5 hours [2].

[2]Note that the time for hyperparameter finetuning is not included.

4.3.1 Recurrent Neural Network (RNN)

We performed hyperparameter tuning for the learning rate, batch size, and dropout by evaluating on the validation set. We experimented with various learning rates ranging from 0.01 to 0.0001. Eventually, we settled on a learning rate of 5×10^{-4}, which provides a good balance between convergence speed and stability. Additionally, our experiments involved batch sizes of 4, 16, and 32. We observed that a batch size of 32 yielded the best performance with respect to convergence and computational efficiency for the models trained on the large dataset, whereas a batch size of 4 was better for the models trained on the smaller dataset. To prevent overfitting, we applied dropout regularization with rates of 0.2, 0.3, and 0.4. A dropout rate of 0.3 was found to be effective in regularizing the model without sacrificing too much learning capacity. The final models were trained with hyperparameters: `learning rate` $= 5 \times 10^{-4}$ with `decay` $= 0.5$, `batch size` $= \begin{cases} 4, \text{ for small dataset} \\ 32, \text{for large dataset} \end{cases}$, `dropout` $= 0.3$, `patience` $= 1$, and `kernel size` $= 2 \times 2$.

4.3.2 Transformer Finetuning

We performed hyperparameter tuning for the learning rate, batch size, and number of epochs by evaluating on the validation set. We experimented with various learning rates ranging from 0.001 to 1×10^{-5}. Eventually, we settled on a learning rate of 2×10^{-5}. Additionally, our experiments involved batch sizes of 16, 32, and 64. We observed that a batch size of 32 yielded the best performance with respect to convergence and computational efficiency. Finally, we considered training for 1, 2, 4, and 10 epochs and eventually settled on 4 for the balance between accuracy and time. The final models were trained with hyperparameters: `learning rate` $= 2 \times 10^{-5}$, weight decay[3] $= 0.01$, `batch size` $= 32$, and `num_epochs` $= 4$.

4.3.3 GPT-4 In-Context Learning

Using GPT-4, we wrote a script to process the JSON files which each had the Rosetta Stone puzzles for a language. The script extracted the training examples section from these files, and used it as an input to guide the model in generating translations. The model was then directed to create a new JSON file that contained the test output in the same format as the input JSON file. A critical aspect of our experimental setup was the adjustment of the temperature parameter of GPT-4. We decided to set the temperature to 0.0 to minimize the randomness in the model's responses. This setting ensured that the translations generated by the model were as deterministic and accurate as possible and returned with a high degree of confidence in the translations derived from the input data.

Further, to ensure we maintained the integrity of the challenge, our prompts explicitly instructed the model to not include any pre-exisitng language-specific knowledge. This instruction tried to simulate a scenario where the model approaches the puzzle from a "new" perspective, just as a participant in the Linguistic Olympiad might.

4.4 Results

We present our results averaged across all languages for each model in table 4.4. In particular, we report the BLEU, ChrF, CharacTER, and exact match scores. For the sake of brevity, we have provided additional in-depth data tables in Appendix section A.2.

Model	BLEU	CHRF	CTER	EM
Random Words	0.083	0.271	0.188	0.000
Fast Align	0.096	0.295	0.237	0.000
RNN	0.132	0.295	0.059	0.000
Transformer Finetuning	0.387	1.710	11.142	0.000
GPT-4	0.420	0.793	0.243	0.000

Table 1: Averaged results across all languages for each model.

Compared to the *Random Words* and *FastAlign* baselines, we find our models have better BLEU and ChrF scores, as expected. Additionally, it is clear that GPT-4 has the best performance on the linguistic puzzle tasks,

[3]The weight decay applied to all layers except all bias and LayerNorm weights in the AdamW optimizer.

which meets our expectations. One anomaly, however, is that the transformer-based finetuning method leads to a very high average CharacTER score.

We hypothesize that part of this reason may be due to the inadequate handling of rare or out-of-vocabulary words. This can contribute to extremely high outlier CharacTER scores even if the model performs well on more common vocabulary. As shown in table A.1, many of the CharacTER values are extreme outliers. Another plausible reason is domain shift. Fine-tuning may not adequately address domain shifts between the training and evaluation data. Since the test data differs significantly from the training data in terms of vocabulary, syntax, and style, the model may struggle to generalize, leading to high CharacTER scores despite good BLEU scores.

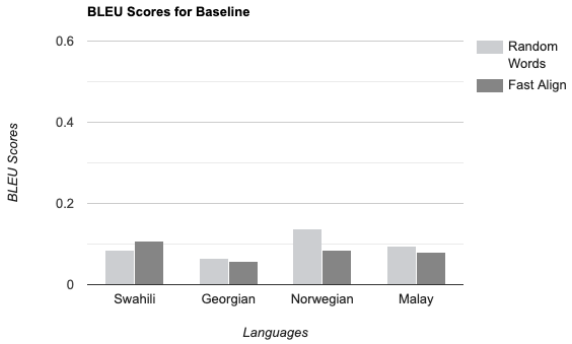

Figure 2: Baseline BLEU scores by language

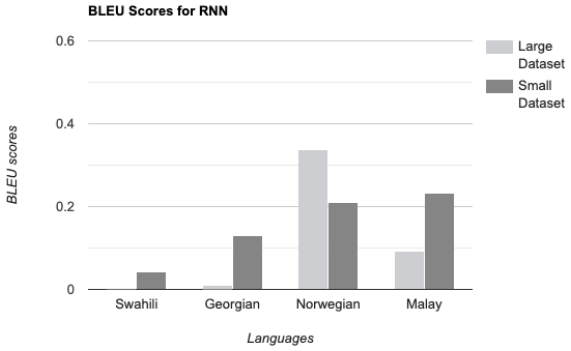

Figure 3: RNN BLEU scores by language

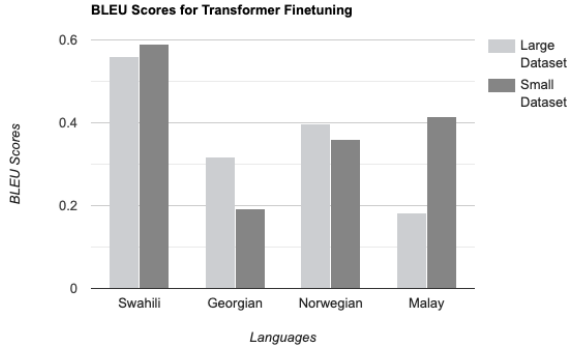

Figure 4: Transformer BLEU scores by language

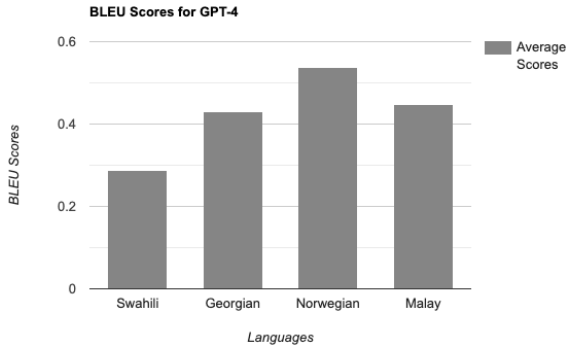

Figure 5: GPT-4 BLEU scores by language

5 Analysis

In general, we expect transformer models to achieve higher BLEU scores when fine-tuned on larger datasets, since these datasets provide more examples for the model to learn and generalize from, leading to more accurate translations that capture nuances in the target language. Swahili exhibited the highest BLEU scores for finetuning on both the large and small datasets. Though surprising, this may be because Swahili has the furthest linguistic distance from English, so translations end up being much simpler, with limited translation options and more direct translations. There may also be less linguistics experts with knowledge of Swahili to create complex linguistics puzzles. From this, the puzzles in Swahili may be easier to solve, making it easier for the models to generate correct translations and high BLEU scores.

Malay, on the other hand, uses the Latin script to simplify text processing but overall has significantly different grammatical and contextual structures compared to English due to their distinct language families (Azmi et al., 2016). Consequently, the model may exhibit proficiency in translating Malay when finetuned with a concise, smaller, puzzle-specific dataset by picking up on patterns deliberately set in the linguistic puzzle, but struggle

to find more complex connections between Malay and English when given a more extensive, varied external corpus. These results are also reflected in finetuning the RNN model with Malay.

Furthermore, we can see that both the RNN and GPT-4 models generated the highest BLEU scores for Norwegian. This can be attributed to several key factors. Firstly, Norwegian is a language that utilizes the Latin alphabet similar to English and has close proximity to English, as they are both Germanic languages – this closeness enhances the models' abilities to grasp and replicate the subtleties of the language pair. Additionally, the higher quality of the Norwegian-English datasets – stemming from Norway's presence in the European sphere – provides a rich training ground for our models. These datasets likely contain nuanced sentence constructions and diverse vocabulary, which is crucial for the models to learn and predict accurate translations.

Additionally, we used the interactive tool BertViz (Vig, 2019) to visualize attention for several attention heads in our finetuned transformer Swahili and Georgian models. We decide to compare these two languages since the transformer models on the small dataset have the highest BLEU score for Swahili and the lowest for Georgian so it is easier to find discrepancies. We display these visualizations in figure 6.

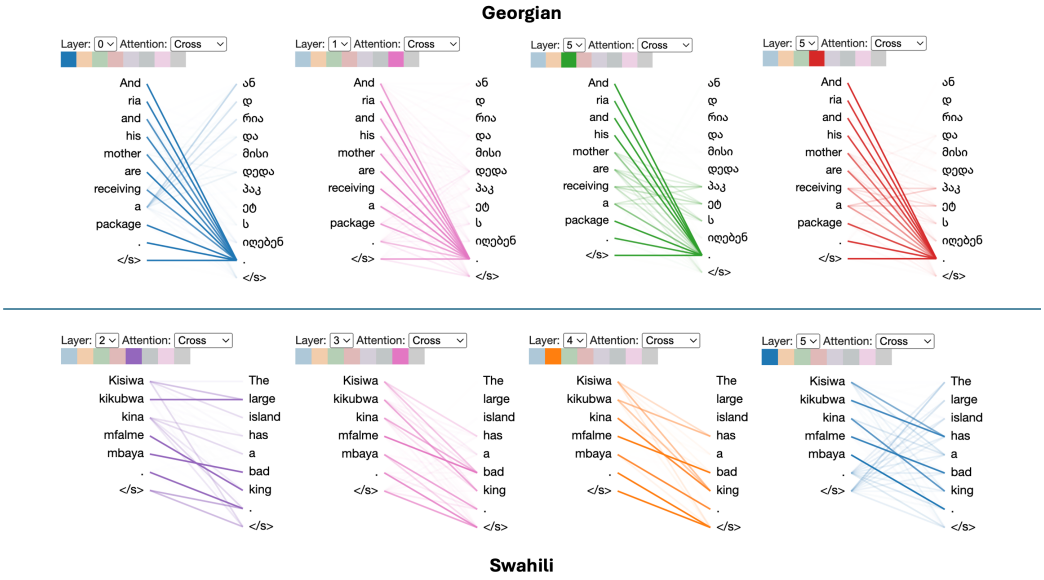

Figure 6: Attention heads are examined across layers for the Georgian and Swahili finetuned models

Notice that almost all attention in the attention heads from the Georgian model is focused on the <.> punctuation mark across layers. This suggests a weak understanding of the relationships necessary to solve a linguistic puzzle. On the other hand, the attention heads for Swahili clearly focus on different English words across layers. For example, the layer 2 attention head for Swahili shows a strong link between "mfaime" and "king" and "mbaya" and "bad", both of which are correct translation pairs. Thus, it is not surprising that the finetuned Swahili model has better performance that the Georgian one.

6 Conclusion

Linguistic puzzles, particularly those involving low-resource languages, pose a distinctive challenge within the field of computational linguistics. Our work focuses on solving such puzzles for Swahili, Georgian, Norwegian, and Malay. We implement three main approaches: i) RNNs with Attention, ii) finetuning pretrained transformer-based models, and iii) in-context learning with GPT-4. All three approaches are improvements from our baselines (BLEU scores $\approx .09$) with the best performances achieved by transformer finetuning and GPT-4, which have BLEU-scores of 0.387 and 0.420, respectively. Moreover, we highlight a gap between state of the art deep-learning and LLM approaches and the human-like reasoning and understanding necessary to solve linguistic puzzles. Thus, we propose that meta-learning approaches are essential to make further progress beyond the performance of LLMs. A limitation of our work is that linguistic puzzle difficulty differs by language. We suggest future work should standard the difficulty of puzzles using human performance as a metric.

References

Australian Computational and Linguistics Olympiad (OzCLO). https://ozclo.org.au/. Accessed 16 Mar. 2024.

What Is NACLO? http://naclo.org/. Accessed 16 Mar. 2024.

2024. NeuLab-TedTalks Dataset. https://opus.nlpl.eu/NeuLab-TedTalks/en&ms/v1/NeuLab-TedTalks. Accessed 16 Mar. 2024.

Mohd Nazri Latiff Azmi, Lidwina Teo Pik Ching, Norbahyah, Muhammad Nur Haziq, Muhammad Habibullah, Muhammad Ammar Yasser, and Kauselya A/P Jayakumar. 2016. The comparisons and contrasts between english and malay languages. *English Review: Journal of English Education*, 4(2):209–218.

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges.

Jean Maillard, Cynthia Gao, Elahe Kalbassi, Kaushik Ram Sadagopan, Vedanuj Goswami, Philipp Koehn, Angela Fan, and Francisco Guzman. 2023. Small data, big impact: Leveraging minimal data for effective machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2740–2756, Toronto, Canada. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, and Sam Altman et. al. 2024. Gpt-4 technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Gözde Gül Şahin, Yova Kementchedjhieva, Phillip Rust, and Iryna Gurevych. 2020. PuzzLing Machines: A Challenge on Learning From Small Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1241–1254, Online. Association for Computational Linguistics.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.

UK Linguistics Olympiad. 2020. UKLO Training Resources. https://archives.uklo.org/wp-content/uploads/2020/10/UKLO-Training-Resources.pdf. Accessed 16 Mar. 2024.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. 2016. CharacTer: Translation edit rate on character level. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 505–510, Berlin, Germany. Association for Computational Linguistics.

A Appendix

A.1 Puzzle Example

Chickasaw	English
Ofi'at kowi'ã lhiyohli.	The dog chases the cat.
Kowi'at ofi'ã lhiyohli.	The cat chases the dog.
Ofi'at shoha.	The dog stinks.
Ihooat hattakã hollo.	The woman loves the man.
Lhiyohlili.	I chase her/him.
Salhiyohli.	She/he chases me.
Hilha.	She/he dances.

Translate the following into Chickasaw:

?	*The man loves the woman.*
?	*The cat stinks.*
?	*I love her/him.*

Translate the following into English:

Ihooat sahollo.	*?*
Ofi'at hilha.	*?*
Kowi'ã lhiyohlili.	*?*

Figure 7: This image shows an example *Rosetta Stone* linguistic puzzle in Chickasaw.

A.2 Data Tables

Source Languages	Baseline Model	BLEU	CHRF	CTER	EM
Swahili	Random Words	0.084	0.219	0.184	0.000
	FastAlign	0.108	0.242	0.189	0.000
Georgian	Random Words	0.066	0.255	0.150	0.000
	FastAlign	0.058	0.247	0.178	0.000
Norwegian	Random Words	0.138	0.324	0.222	0.000
	FastAlign	0.086	0.326	0.341	0.000
Malay	Random Words	0.095	0.285	0.194	0.000
	FastAlign	0.081	0.364	0.241	0.000

Table 2: Results across all languages for the *Random Words* and *FastAlign* baselines.

Source Languages	Dataset Type	RNN Model	BLEU	CHRF	CTER	EM
Swahili	Large	Source to English	0.000	0.144	0.063	0.000
		English to Source	0.000	0.203	0.068	0.000
	Small	Source to English	0.087	0.260	0.100	0.212
		English to Source	0.000	0.250	0.124	0.022
Georgian	Large	Source to English	0.022	0.130	0.030	0.000
		English to Source	0.000	0.116	0.061	0.000
	Small	Source to English	0.258	0.384	0.039	0.083
		English to Source	0.000	0.337	0.064	0.000
Norwegian	Large	Source to English	0.289	0.450	0.058	0.143
		English to Source	0.385	0.547	0.061	0.143
	Small	Source to English	0.138	0.271	0.067	0.000
		English to Source	0.283	0.348	0.066	0.000
Malay	Large	Source to English	0.101	0.202	0.046	0.000
		English to Source	0.083	0.302	0.041	0.000
	Small	Source to English	0.345	0.411	0.030	0.000
		English to Source	0.119	0.358	0.029	0.000

Table 3: Results for all four RNN models per language.

Source Languages	Dataset Type	Transformer Model	BLEU	CHRF	CTER	EM
Swahili	Large	Source to English	0.489	1.334	1.672	0.000
		English to Source	0.629	1.557	1.729	0.000
	Small	Source to English	0.491	1.403	1.605	0.000
		English to Source	0.692	1.679	1.572	0.000
Georgian	Large	Source to English	0.346	2.198	6.967	0.000
		English to Source	0.289	0.852	9.295	0.000
	Small	Source to English	0.187	0.946	10.897	0.000
		English to Source	0.199	0.883	10.897	0.000
Norwegian	Large	Source to English	0.495	1.888	1.143	0.000
		English to Source	0.300	1.515	3.781	0.000
	Small	Source to English	0.422	1.618	15.429	0.000
		English to Source	0.300	1.985	18.286	0.000
Malay	Large	Source to English	0.264	2.623	35.000	0.000
		English to Source	0.414	2.317	20.000	0.000
	Small	Source to English	0.264	2.317	20.000	0.000
		English to Source	0.414	2.317	20.000	0.000

Table 4: Results for all four finetuned transformer models per language.

Source Languages	GPT-4 Model	BLEU	CHRF	CTER	EM
Swahili	Source to English	0.244	0.642	0.571	0.256
	English to Source	0.333	0.817	0.798	0.333
Georgian	Source to English	0.375	0.506	0.517	0.250
	English to Source	0.487	0.839	0.827	0.125
Norwegian	Source to English	0.394	0.884	0.852	0.250
	English to Source	0.680	0.984	0.943	0.333
Malay	Source to English	0.448	0.877	0.850	0.000
	English to Source	N/A	N/A	N/A	N/A

Table 5: Results for GPT-4 model per language