POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Figurative Language Understanding based on Large Language Models

Supervisors
Prof. Luca Cagliero
Prof. Natalie Parde
Dr. Giuseppe Gallipoli

Candidate
Susanna Olivero

Academic year 2023-2024

Table of Contents

List of Figures

List of Tables

Computers will understand sarcasm before Americans do

[Geoffrey Hinton]

Abstract

In the vast realm of Natural Language Processing, one of the areas that still presents a bottleneck is Figurative Language Understanding. However, this field is of fundamental interest both theoretically and practically; in all new applications of Artificial Intelligence, there is an increasing demand for a correct understanding of human language, which is naturally rich in rhetorical figures. To comprehend this figurative language, it is necessary for the model used to grasp all the different nuances and reasons, going beyond the mere literal meaning.
Until now, attempts to solve this problem have primarily involved training specific models on large databases of rhetorical figures. In this thesis, we seek to overcome this challenge by utilizing a Large Language Model, specifically LLaMA. These kinds of models have already demonstrated immense potential in many Natural Language Processing tasks, and with this project, we have observed that it is possible to achieve promising performance simply by harnessing the potential of Large Language Models without specifically training them for our specific goal.
In particular, we explore the use of LLaMA employing different types of prompting and through two separate tasks. The first one involves entailment classification of figurative language, while the second one is a text generation task where the model is asked to generate an explanation for the previously made choice.
Through this exploration, we aim to contribute to the understanding of figurative language, emphasizing the transformative potential of Large Language Models in deciphering figurative expressions.

1 Introduction

1.1 Motivation

In the last decade, the so-called Artificial Intelligence (AI) has been at the forefront of a true revolution. Within this vast area, we identify the field of Natural Language Processing (NLP) as one of the most important, as it defines and enables interaction between humans and computers. NLP applications have made significant progress recently, successfully solving complex tasks with practical implications. However, one area that still presents a bottleneck is Figurative Language Understanding, which emerges as an intriguing and complex frontier.

The Figurative Language Understanding is a field of fundamental interest both theoretically and practically. Figurative language, including metaphors, similes, and other non-literal expressions, constitutes a rich aspect of human communication and the mistaken comprehension of a rhetorical figure can often change the entire meaning of a sentence.
In real-world AI applications, as chatbots and virtual assistants, there is an increasing demand for a correct interaction with the users and since the human language is naturally rich in rhetorical figures the ability to comprehend figurative language becomes crucial.

Unfortunately, this task is not so easy. This kind of language is indeed characterized by ambiguity and subjectivity, and, in order to comprehend it, the model used must grasp all the different nuances and interpretations, going beyond mere literal meaning. This is a formidable challenge for traditional language models, which have often failed in this field. For this reason, we chose to explore the usage of Large Language Models (LLMs).
The LLMs have literally exploded in the last years, as can be seen in the Figure 1.1, and they have become increasingly widespread across various domains.
LLMs are renowned for their ability to generate good human-like text and have found implementations in many NLP applications such as chatbots, virtual assistants, translation, context generation, or assisting researchers. LLMs are characterized by an expansive pre-trained knowledge and present a promising avenue for addressing the figurative understanding challenges.
Our investigation seeks to leverage LLMs in deciphering figurative expressions,

exploring the potential synergies between these advanced models and the complexities of figurative language.
Beyond these theoretical motivations, the decision to employ LLMs regards also a pragmatic reason since, as said before, the increasing prevalence of such models in real-world applications.

In summary, this thesis is motivated by the intertwined goals of deepening theoretical understanding of figurative language and exploring the transformative potential of Large Language Models.

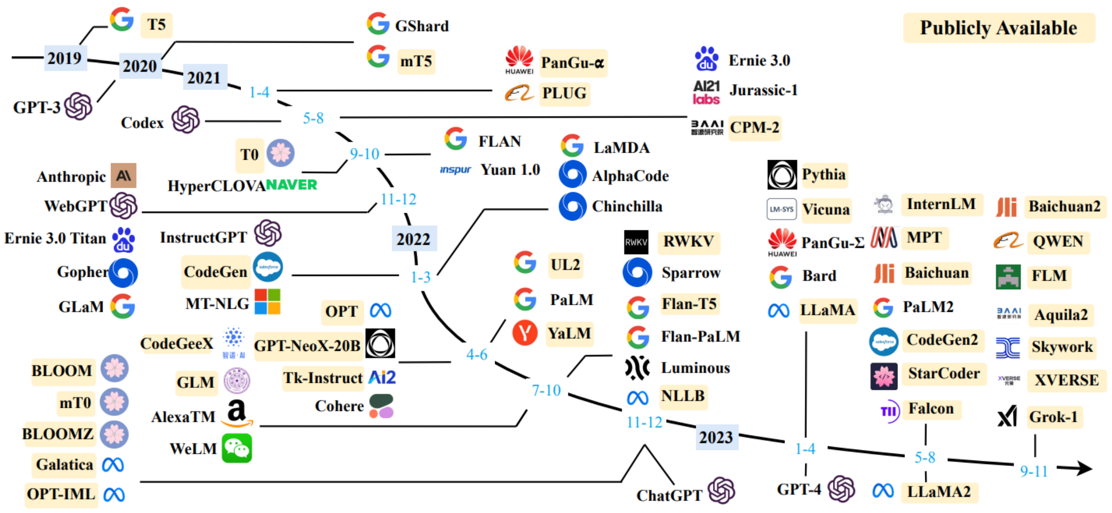


Figure 1.1: A timeline of existing LLMs in recent years, established according to the release date [46].

This project draws inspiration from the FigLang 2022 Shared Task, a competition proposed in 2022 with the aim of addressing the challenges posed by figurative language. In particular, the challenge was based on two different aspects: the first involves classification, aiming to discern whether there exists entailment or contradiction between two sentences, one of which contains a rhetorical figure. Meanwhile, the second task involves generating an explanation that clarifies the preceding choice. The works produced through this competition serve as the foundation for our own investigations.

1.2 Contribution

In undertaking the exploration of Figurative Language Understanding, this thesis makes a distinctive contribution by embracing the utilization of LLMs, notably focusing on the implementation of LLaMA 2 Chat. Our decision to integrate LLMs into the fabric of our research represents a choice driven by the recognition of their transformative potential in surmounting the challenges inherent in figurative language comprehension. The models of LLaMA family, with their extensive training on vast corpora, have demonstrated an unparalleled ability to capture the nuances of language, making them well-suited to decipher figurative expressions.

Compared to the solutions presented for the FigLang 2022 challenge, our approach offers a fresh perspective. While most solutions involved training the model in a specific manner to tackle the task, we attempted to solve it without any specialized training on the model. By solely utilizing LLaMA, we aim to showcase the potential of LLMs in expanding the limits of figurative language comprehension.

To tackle the proposed challenge, we specifically tasked LLaMA with addressing two distinct tasks: one involving classification and the other text generation. The first regards an entailment classification of figurative expression, while for the second we ask the model to generate an explanation for the previously made choice. Together, these tasks provide insight into the reasoning process that the LLM undertakes to understand and decipher various rhetorical figures.
We interacted with the model without any training, solely utilizing the Prompting method, which is fast and computationally inexpensive, making it much more practical than traditional training on a vast database. Throughout the study, various forms of prompting were employed, including zero-shot prompting, few-shot prompting, and Chain-Of-Thought prompting (CoT).
So, the contribution of the thesis also lies in exploring the use of LLMs for two different tasks and with different modalities (zero-shot, few-shot, CoT).

To evaluate our work, we started using the basic metrics proposed by the FigLang 2022 challenge. Since these metrics were primarily designed to assess only the first task, the classification, we conducted more detailed evaluations to thoroughly assess the second task, text generation. To accomplish this, we utilized different metrics, some of which were not included in the challenge but are commonly used for this type of evaluation.

1.3 Structure

In this section we provide a brief overview of the entire thesis structure to offer a roadmap into the project. Following this introductory chapter, there are six additional chapters.

The second chapter is primarily theoretical and provides the foundations upon which the research is built. We begin with a general introduction to NLP, navigating through the evolution of language models. We discuss world representations, progressing from the simplest forms to the deeply contextualized ones. We then proceed to examine sequence-to-sequence models and transformers, with a particular focus on the attention technique. The final part of this chapter is dedicated to LLMs, with a specific emphasis on the LLaMA 2 family, the LLM used in our experiments.

With the third chapter, we shift the focus towards the challenges posed by Figurative Language Understanding. After an introduction to this field and an overview of the principal datasets, we introduce the *FLUTE* dataset, the one we used. Subsequently, we present in detail the FigLang 2022 challenge, providing an insightful overview of participants, principal results, and systems. As mentioned earlier, this challenge is the starting point of our project.
In this chapter, we also introduce the *DREAM* method and its adaptation, *DREAM-FLUTE*; both methodologies are employed in this thesis.

The fourth chapter outlines the methodology adopted in the experiments. It elucidates the approach taken in conducting the research and its various phases, as data preparation, model adaptation and evaluation. Towards the end of the chapter, we also provide some information regarding implementation details, focusing on the practical aspects of the research. This chapter acts as a bridge, connecting the theoretical foundations established in the preceding chapters with the practical development of the research.

The fifth chapter is dedicated to a theoretical exploration of evaluation metrics. We provide a comprehensive overview of principal metrics in the fields of classification and text generation, with particular attention to those employed in our experiments. Additionally, we delve into the realm of explainable artificial intelligence metrics, presenting the most relevant results obtained so far.

In the sixth chapter, we unveil the outcomes of our experiments. After detailing the experimental setup, we present the results, looking at the performance of the systems employed.

In the final chapter we sum up the journey, providing a comprehensive conclusion. Additionally, we cast a forward-looking perspective into the future, finding possible areas for potential future research and development.

2 Fundamentals of DeepNLP

In this chapter, we provide the theoretical foundations upon which the research is built. After a general introduction to NLP, we discuss word representations, progressing from the simplest forms to the deeply contextualized ones. We then proceed to examine language models, sequence-to-sequence models and transformers, with a particular focus on the attention mechanism. The final part is dedicated to Large Language Models, with a specific emphasis on the LLaMA 2 family.

2.1 Introduction to NLP

Natural Language Processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence that focuses on the interaction between computers and human language. Specifically, it deals with how to program computers to process and analyze human language in the form of text or voice data.

NLP is a pivotal force in contemporary machine intelligence applications. It powers various real-world uses, including language translation, voice-activated systems, and rapid text summarization. Common encounters with NLP include voice-operated GPS, digital assistants, speech-to-text tools, and customer service chatbots. Beyond consumer applications, NLP is increasingly integral to enterprise solutions, contributing to streamlined business operations, enhanced employee productivity, and the simplification of crucial business processes.
The main tasks in this field can be divided into two broad sectors, although they are closely interconnected and interact continuously in real-world applications.

1. Natural Language Understanding:
   This involves the process of extracting meaningful information from text, including unstructured or semi-structured text. It includes tasks such as tokenization, text categorization, sentiment analysis and name entity recognition.

2. Natural Language Generation:
   The goal here is to produce human-like text or speech based on structured data or specific instructions. This encompasses tasks like summarization, machine translation, and question answering.

Like other advancements in artificial intelligence, NLP has seen significant growth in the past decade, both in terms of the programming techniques employed and in term of the size and complexity of the utilized models. This technology should enable computers to process natural language and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

The initial NLP algorithms were rooted in linguistics, employing rule-based models to mimic human language. Subsequently, there was a transition to statistical models, and eventually, the evolution led to the adoption of machine learning and deep learning models (Figure 2.1).

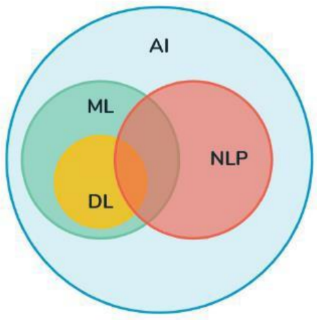


Figure 2.1: NLP: Natural Language Processing, AI: Artificial Intelligence, ML: Machine Learning, DL: Deep learning

The earliest NLP applications were hence manual, rule-based systems capable of executing only some specific NLP tasks. These systems faced challenges in handling an extensive array of exceptions and coping with the growing volumes of textual and vocal data.
Statistical NLP emerged as a solution, integrating computer algorithms with machine learning and deep learning models. This approach automates the extraction, classification, and labeling of elements, assigning a statistical likelihood to each potential interpretation of these elements.
In contemporary NLP, called DeepNLP, the focus is the utilization of deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), that empower NLP systems to dynamically ’learn’ during their operation. Consequently, these systems can extract increasingly precise meanings from vast collections of raw, unstructured, and unlabeled text and voice datasets.

2.2 Deep Contextualized Word Representation

NLP has undergone a remarkable evolution, progressing from early word embeddings to the more sophisticated realm of contextualized embeddings.
The inception of NLP can be traced back to the advent of word embeddings, which aimed to represent words as dense, continuous vectors in a semantic space. Pretrained word embeddings, popularized by models like Word2Vec and GloVe, compute a single static representation for each word. The representation is hence fixed and independent from the context in which the word appears. This fact leads to important deficits, these models lacked the ability to capture the nuances of sentence structure and meaning comprehensively, they ignore the role of context and do not capture the long-term dependencies.
This limitation spurred the development of sentence embeddings, which sought to encapsulate the meaning of entire sentences in vector representations. Sentence embeddings, exemplified by models such as Doc2Vec, Sent2Vec and InferSent, provided a leap forward by considering the context of words within sentences. Despite their advancements, they still had a hard time dealing with words having multiple meanings and language details that depend on the context.

The turning point in NLP came with the introduction of contextualized embeddings. Models like ELMo, GPT (Generative Pre-trained Transformer), and BERT (Bidirectional Encoder Representations from Transformers) revolutionized the field by considering the context in which words and sentences appeared. Contextualized embeddings capture the dynamic nature of language, providing a more nuanced understanding of meaning in different contexts.
Unlike static word embeddings, contextualized embeddings are representations of words in context. They can circumvent many of the limitations associated with word and sentence embeddings, as they can model complex characteristics of word use (e.g., syntax and semantics), as well as they can capture how these uses vary across linguistic contexts.
Contextualized word embeddings are dynamic, the same word can be assigned different embeddings if it appears in different contexts. Instead of receiving words as distinct units and providing independent word embeddings for each, contextualized models receive the whole text span (the target word along with its context) and provide specialized embeddings for individual words which are adjusted to their context.
The training of contextualized embeddings is carried out at a pretraining stage, independently from the main task, on a large text corpus. The trained model can then generate contextualized representations for all the words in the given text. Depending on the sequence encoder used in language modeling, these models can be put into two broad categories: RNN, mostly Long Short-Term Memory

(LSTM), and Transformer.
Unlike knowledge-based sense representations, these embeddings do not rely on annotated data or external lexical resources, and can be learned in an unsupervised manner.

Among the various textual embedding techniques just mentioned, the two main ones are Word2Vec and BERT, and we will briefly present them below.

- **Word2Vec** is a fundamental encoder in the field of NLP and one of the first actually used. Its distinctive feature is the ability to represent words as dense, fixed-size vectors in a continuous vector space. Google' researchers employed unsupervised learning techniques tring to capture the semantic meaning of words based on their contexts in large corpora of text.
  Word2Vec is based on the principle of distributional semantics, where words with similar meanings tend to occur in similar contexts. By training neural networks, there are two foundamental tasks that can solve, the first is to predict the target word given its context, continuous bag of words model (CBOW), and the second is to predict the surrounding words given a target word, skip-gram model. Word2Vec effectively learns high-dimensional representations of words that preserve semantic relationships. These word embeddings have revolutionized various NLP tasks, including semantic similarity measurement, text classification, and sentiment analysis, by providing dense and meaningful representations of words that capture their contextual semantics. Word2Vec's simplicity, efficiency, and effectiveness have made it a cornerstone in modern NLP applications, shaping the landscape of text processing and understanding.

- **BERT** (Bidirectional Encoder Representations from Transformers) stands as a groundbreaking development in the NLP field. This model has revolutionized the way computers comprehend human language and represents a significant leap forward in language understanding capabilities.
  BERT is a pre-trained language model, unlike previous models that processed text in a unidirectional manner, it considers both the left and right context of each word in a sentence and this fact increases considerably its ability to capture contextual information. This approach, called bidirectional, allows BERT to better grasp intricate linguistic nuances and understand the contextual meaning of words in a given sentence.
  It is a very versatile and powerful tool instead, its pre-trained nature enables fine-tuning on specific tasks with minimal additional training data. Moreover, powered by the Transformer architecture, BERT enables parallel processing of words, facilitating more efficient and effective language understanding.

BERT, Figure 2.2, has quickly become a cornerstone in both academic research and industrial applications, due to its remarkable performance across various NLP tasks, as sentiment analysis, named entity recognition, and question answering.

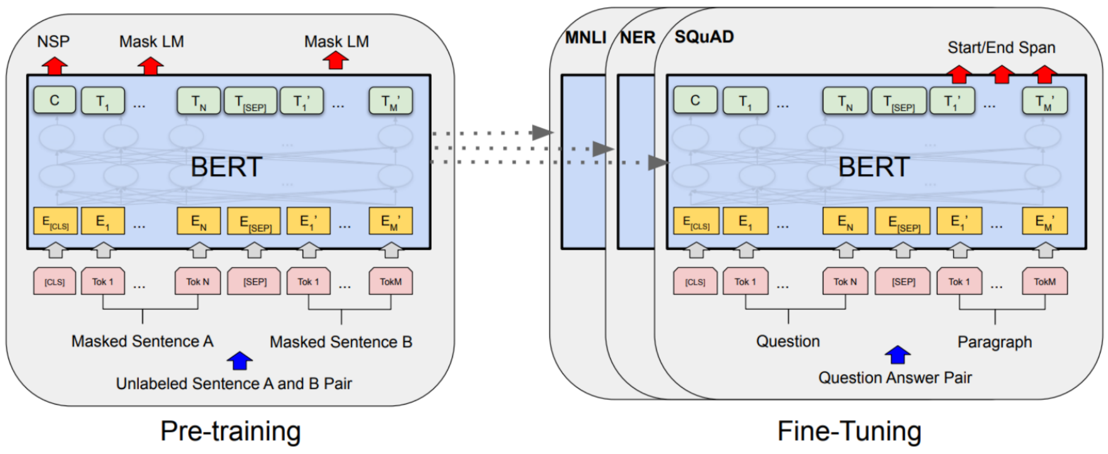


Figure 2.2: BERT: pre-training and fine-tuning procedures [10].

2.3 Language Models

Language Models (LMs) aim at predicting the next word in a sentence given the preceding words.
To be able to accurately predict a word in a sequence, LMs need to encode both the semantic and syntactic roles of words in context. The knowledge acquisition bottleneck is not an issue for LMs, since they can be trained on a multitude of raw texts in an self-supervised manner. In fact, extensive models can be trained with LM objectives and then transferred to specific tasks.
RNN-based LMs rely on an LSTM-based encoder, since LSTMs are known to be able to capture word order to a good extend. Also, LSTMs are capable of combining word representations in a more reasonable manner, assigning higher weights to words that are semantically more important than others in the context.

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a *forward LM* computes the probability of the sequence by modeling the probability of token t_k given the history $(t_1, ..., t_{k-1})$:

$$P(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} P(t_k | t_1, ..., t_{k-1})$$

A *backward LM* is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context $(t_{k+1}, ..., t_N)$:

$$P(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} P(t_k | t_{k+1}, ..., t_N)$$

A *bidirectional LM* combines both a forward and backward LM in order to have the sense of both the next and the previous words. It trains the forward and backward LMs jointly.

For example in the TagLM model, proposed in [26], a multi-layer bidirectional LSTM (BiLSTM) sequence encoder on monolingual texts, has been trained. TagLM is the precursor of ELMo (Embeddings from Language Models), proposed in [27].
ELMo is the principal model for the deep contextualized word representation and it utilize a bidirectional LSTM network. Figure 2.3 provides a high-level illustration of how ELMo embeddings are constructed. A residual connection between the LSTM layers enables the deeper layers to closely examine the original input and to allow the gradients to efficiently backpropagate to the initial layers.

The model is trained on large amounts of texts with the language modeling objective: predicting the next token in a sequence of tokens. The trained model

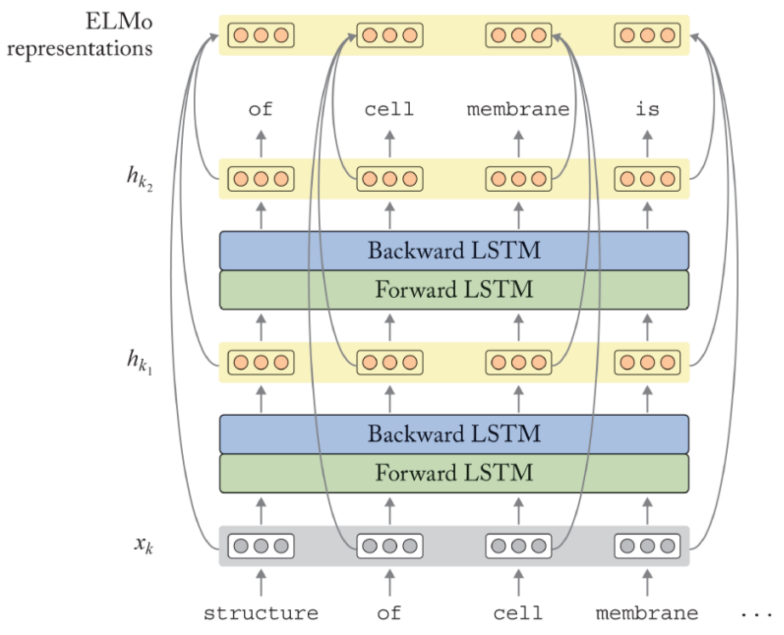


Figure 2.3: ELMo high-level illustration [4].

is then used to derive contextualized embeddings that can be used as input for a variety of NLP systems.

Several methods exist for merging the outputs of the ELMo model, i.e., the hidden states of the two BiLSTM layers and the context-independent representation. One may select only the top layer output or concatenate the representations from the top-n layers to have long vectors for each token, to be fed as inputs to an NLP system. One can also learn a weighted combination of these layers, based on the target task, or concatenate other static word embeddings with ELMo embeddings. To integrate ELMo into the supervised model, we initially freeze the weights of the BiLSTM. Subsequently, for each task, we train task-dependent softmax weights to combine the layer-wise representations into a unified vector. As a matter of fact, once pre-trained, the biLSTM can compute representations for various task. Fine tuning the biLSTM on domain specific data has proven effective, resulting in notable reductions in perplexity and improvements in performance for subsequent tasks.

2.4 Sequence-to-Sequence Models

Sequence-to-Sequence models (seq2seq) represent a powerful class of neural network architectures designed for convert sequences of items (word, letters, time series, etc.) from one domain to sequences in another domain.
The principal NLP tasks involving sequential data, such as machine translation, summarization, and text generation.

Developed to handle input and output sequences of varying lengths, these models consist of two main components: an encoder and a decoder.

- The **Encoder** processes the input sequence, converting it into a fixed-size vector, called context vector or encoding. It processes the input data step by step, typically using RNNs or variants like LSTM networks or gated recurrent units (GRUs).
  The generated context vector captures the essential information of the input sequence and serves as input to the decoder.

- The **Decode**r, in turn, generates the output sequence based on the provided context vector. It does so by predicting one element at a time, typically one word, while considering the previously generated elements. During training, the model learns to minimize the difference between its predictions and the actual target sequence. The decoder often employs RNNs or similar architectures, operating in a generative manner, producing one element of the output sequence at a time.

Summarizing, the main difference between the encoder and decoder lies in their function. The encoder's goal is to capture the semantic information of the input and it does not require initialization with an external state, as it directly processes the input sequence. Conversely, the decoder is tasked with generating the output sequence and is often initialized with the context vector produced by the encoder as its initial hidden state.

Since the task is sequence based, both the encoder and decoder tend to use some form of RNNs, LSTMs, etc. to effectively capture sequential dependencies and relationships. The context vector can be of any size, as it basically represents the number of hidden units in the encoder RNN.
RNNs are designed to consider two inputs: the current example they encounter and a representation of the previous input. Consequently, the output at a given time step (t) relies on both the current input and the input at the preceding time step (*t-1*). This inherent design makes them particularly effective in tasks involving sequences, as they preserve sequential information in a hidden state, which is then

utilized in subsequent instances. The last hidden state generated by the encoder at the end of the sequence is actually the context sent to the decoder.

In Figure 2.4 there is an example of a neural machine translation where the sequence is a series of words processed one after another.

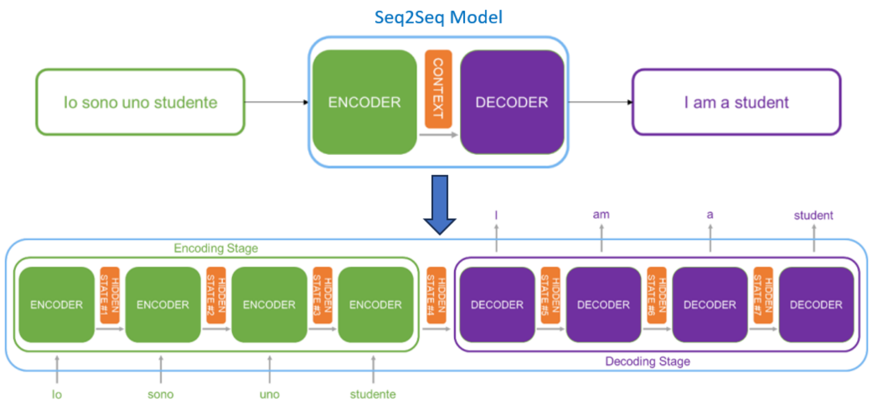


Figure 2.4: An example of a sequence in a neural machine translation with an unrolled view of the Seq2Seq model

The output sequence relies heavily on the context defined by the hidden state in the final output of the encoder, making it challenging for the model to deal with long sentences. In the case of long sequences, there is a high probability that the initial context has been lost by the end of the sequence.
The solution to this problem is provided by employing the "attention" technique, which allows the model to focus on different parts of the input sequence at every stage of the output sequence allowing the context to be preserved from beginning to end.

2.5 The Attention Mechanism

The Attention mechanism is a pivotal innovation in the field of neural network architectures, first presented in the paper *Attention is All You Need* [41].
Its primary purpose is to enhance the model's ability to focus on specific parts of the input sequence when making predictions or generating outputs.

In traditional sequence-to-sequence models, such as those based on RNNs, the entire input sequence is typically summarized into a fixed-size context vector. However, this approach may lead to information loss, especially in long sequences. The Attention mechanism addresses this limitation by allowing the model to dynamically weight the importance of different parts of the input sequence, allocating more attention to relevant segments.
In essence, the Attention mechanism works by assigning attention scores to different elements of the input sequence. These scores are then used to compute a weighted sum, creating a context vector that reflects the model's focus on specific parts of the input. This adaptive attention mechanism proves particularly beneficial in tasks like machine translation, where aligning words in the source and target languages is crucial.

The introduction of Attention has significantly improved the performance of sequence-to-sequence models, enabling them to handle longer sequences more effectively and capture intricate dependencies within the data. This mechanism has become a fundamental building block in many state-of-the-art neural network architectures.

Formulation of the attention mechanism

An attention model differs from a classic sequence-to-sequence model in two main ways. First, the encoder passes a lot more data to the decoder, indeed, instead of passing only the last hidden state of the encoding stage, the encoder passes all the hidden states to the decoder. Second, the context vector is generated at every time step in the output sequences, as a weighted sum of the input hidden states. So, at each time instance, the generated context vector is combined with the hidden state vector by concatenation and this new attention hidden vector is used for predicting the output.
These attention scores are the output of an additional neural network model known as the alignment model, which is trained jointly with the initial seq2seq model. The alignment model scores, for every input, how well its hidden state (input) matches with the attention hidden state (previous output). Therefore, it aligns source and target sequences, instead of compressing the key information about the

source sentence into a fixed-size vector. Its aim is minimize the information loss from long sentences, avoiding missing long-term dependencies.
The last step is made by a softmax function that take all these scores and return the attention score for each input. At this point, we know which parts of the input sentence are the most important for the prediction and which are not.

In [41] the authors introduced the *Scaled Dot-Product Attention* where the input consists of queries and keys of dimension d_k, and values of dimension d_v. In this case the dot product is between the query and all the keys and then is divided by $\sqrt{d_k}$. The last passage is always the application of the softmax function that give as the weights on the values.
In practise, the attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V.

$$\text{attention}(Q, V, K) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \in \mathbb{R}^{n \times v}$$

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query q, keys k, values v, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

$$\text{attention}(q, k, v) = \sum_i \text{similarity}(q, k_i) \cdot v_i$$

The similarity scores S_i can be defined in different ways, here some examples:

- Additive attention $S_i = w_3 \tanh(w_2^T q + w_1^T k_i)$
- Dot product attention $S_i = q^T k_i$
- Scaled dot product $S_i = \frac{q^T k_i}{\sqrt{d_k}}$

These scores will be normalized with the softmax function in order to sum to 1:

$$p_i = \text{softmax}(S_i)$$

So, at the end, the attention score v is computed as the weighted sum of the value vectors v_i weighted by their scores p_i:

$$z = \text{attention}(q, k, v) = \sum_i p_i v_i$$

Multi-Head Attention

Sometimes it is useful to combine knowledge from different behaviors of the same attention mechanism, such as capturing dependencies of various ranges (e.g., shorter-range vs. longer-range) within a sequence. This is possible with the so-called Multi-Head Attention mechanism that allow to jointly use use different representation subspaces of queries, keys, and values.

In [41] the keys, queries and values are linearly projected h times with different learned linear projections to d_k, d_k, d_v dimensions, respectively. These h projected queries, keys and values are subsequently fed into attention pooling in parallel and then the h attention pooling outputs are concatenated and transformed with another learned linear projection to produce the final output. Each of the h attention pooling outputs is called a head, from here the name Multi-Head Attention.

Given as parameters, $W_i^Q \in \mathbf{R}^{d_{model} \times d_k}, W_i^K \in \mathbf{R}^{d_{model} \times d_k}, W_i^V \in \mathbf{R}^{d_{model} \times d_v}$ and $W_i^0 \in \mathbf{R}^{hd_v \times d_{model}}$ we can write this:

$$\text{MultiHead}(Q, K, V) = \text{concat}(h_1, ..., h_h)W^0$$

where h stands for 'head' and

$$h_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Multiple heads are concatenated using fully-connected layers and then learnable linear transformations are performed, Figure 2.5 describes this mechanism.

Self Attention

The Self-attention mechanism is another attention mechanism that relate different positions of a single sequence in order to compute a representation of the sequence. It allows the model to weigh the importance of different elements within the input sequence, enabling more effective capture of long-range dependencies and contextual information.

Here an example to better understand the concept of self-attention.
Say that the input sentence is “The animal didn’t cross the street because it was too tired”, we want to translate it, but, what does "it" in this sentence refer to? Is it referring to the animal o to the street? It is an obvious question to a human but it is not so simple for a machine. Self-attention is the technique that allows to associate "it" with "animal". This is possible due to the fact that the model processes each word (each position in the input sequence) and self-attention allows it to look at other positions in the input sequence for clues that can help lead to

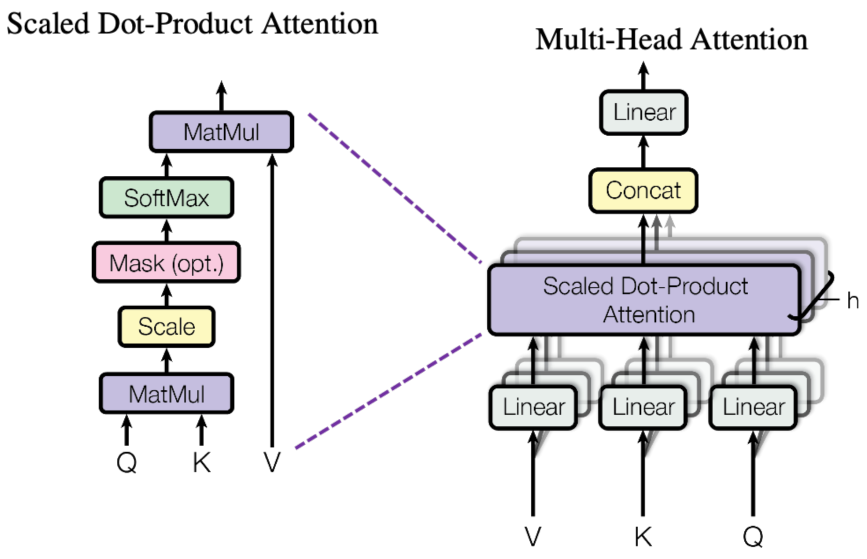


Figure 2.5: Left: Scaled Dot-Product Attention. Right: Multi-head attention [41].

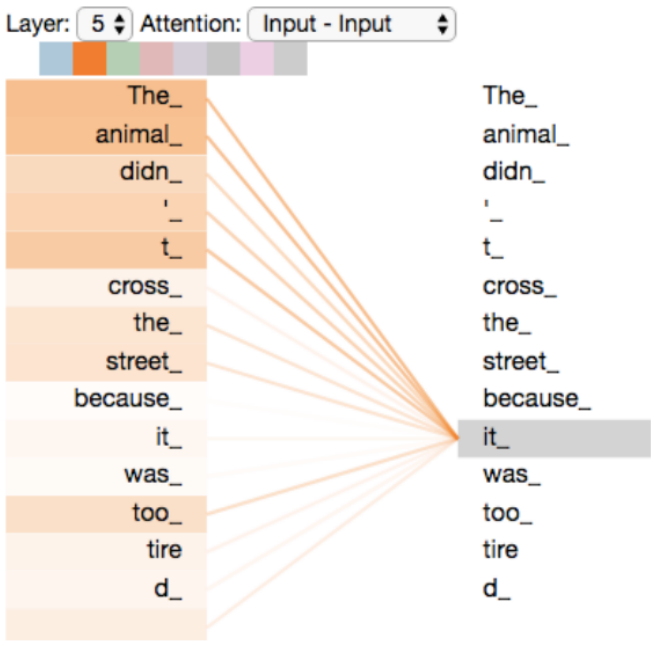


Figure 2.6: Attention mapping, as we are encoding the word “it” in the top encoder in the stack, part of the attention mechanism was focusing on “The Animal”, and baked a part of its representation into the encoding of “it”.

a better encoding for this word. Figure 2.6 show the attention mapping for this example.

Now we are going to explain how self-attention works, step-by-step. For a schema of this attention mechanism see the example at Figure 2.7.

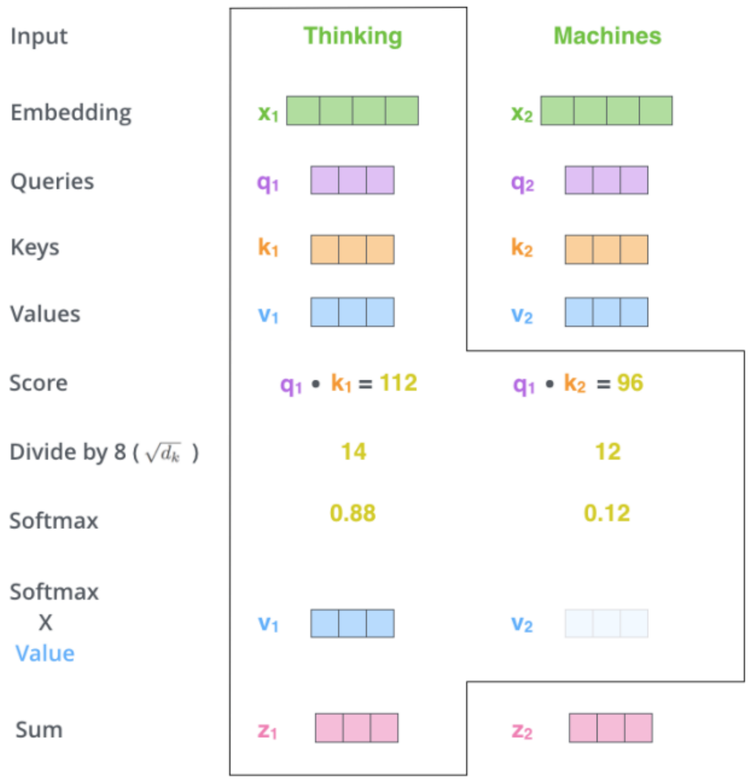


Figure 2.7: Example of self-attention in details [41].

- The first step in calculating self-attention is to create three vectors (query, key, value) from each of the encoder's input vectors, in this context, the embedding of each word. These vectors are created by multiplying the embedding by three matrices that we trained during the training process.
- The second step is to calculate a score. To calculate the self attention for a specific word in a sentence, we need to score each word of the input sentence against that word. The score determines how much focus to place on other parts of the input sentence as we encode a word at a certain position. It is calculated computing the dot product between the query vector and the key vector of the respective word we're scoring. For example, if we want the

self-attention for the first word in the input sentence, the first score would be the dot product of q_1 and k_1, while the second score would be the dot product of q_1 and k_2.

- The third step is to divide the scores by the square root of the dimension of the key vectors $\sqrt{d_k}$.
- The fourth step is to pass the result through a softmax function, which normalizes the scores, so they are all positive and add up to 1.
- The fifth and sixth steps are to multiply each value vector v by the softmax score and then sum up the resulting weighted value vectors.The intuition here is to keep intact the values of the word(s) we want to focus on, and drown-out irrelevant words (by multiplying them by tiny numbers like 0.001). This produces the output of the self-attention layer at this position.
- The last step consists in sending the resulting vector along to the feed-forward neural network.

Note that, as before, in the actual implementation the calculation is done in matrix form for faster processing.

2.6 Transformers

Until mid-2017, RNNs were the optimal choice for encoding text sequences into fixed-size representations. However, the introduction of the *Transformer* model, revolutionized the field of NLP, introducing a new, substantially more powerful, alternative for RNNs [4].

Before Transformers, the general belief was that it is possible to capture long-range dependencies only using recurrence, so utilizing the so called RNN architecture. What makes the Transformers interesting is that they capture long-range dependencies without resorting to any sort of recurrence.
The Transformer architecture is a feed-forward model with no recurrence, its only memory is based on the attention mechanism. Its architecture is based on the encoder-decoder structure, similarly to RNN-based sequence models. However, in this case, it do not receive input tokens sequentially (one token at a time) but it takes all the tokens in the sequence at once and in parallel. This parallel functionality makes the Transformer substantially more efficient than RNN, remaining capable of capturing the long-distance dependencies.
Figure 2.8 provides an illustration of a Transformer model.

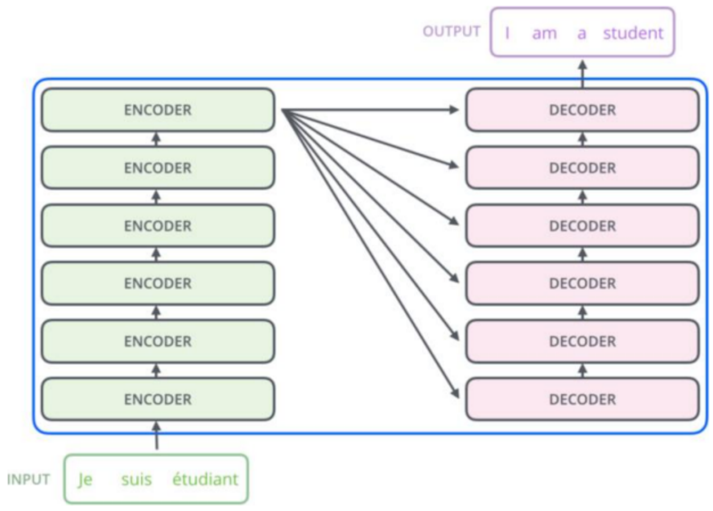


Figure 2.8: An example of a Transformer model used for translation. The model is auto-regressive and has an encoder-decoder structure. The encoder and decoder have six identical encoders and decoders, respectively [41].

The attention mechanism used by the Transformer is the self-attention method, it is used to bake the “understanding” of other relevant words into the one it is currently processing.

Transformer architecture

In this paragraph we are going to present the Transformer architecture, for a better understanding see Figure 2.9.

As said before the Transformer is build on a encoder-decoder structure.

- **Encoder**
  The Transformer encoder is a stack of multiple identical layers, where each layer is composed by two sublayers. The first is a multi-head self-attention pooling; here, queries, keys, and values are all from the outputs of the previous encoder layer. The second sublayer is a position-wise feed-forward network.

  Multi-head attention is performed on all words in the input sentence, where each word is treated as a query. Then, the attention process continue as usual: the attention between every position and every other position is computed and a convex combination of the corresponding values is taken to perform a dot product. Since it is a multi-head attention method, information from pairs are merged together and the process is repeated multiple times.

  In addition, there is also a residual connection that is employed around both sublayers (inspired by the ResNet design). This residual connection is immediately followed by a layer normalization. This layer normalizes values in order to have zero mean and single variance, so that all outputs have the same scale. This improve the convergence of the network, as it decreases the dependencies between each layer and reduces the number of weight adjustments in gradient descent.

  As a result, the output of the Transformer encoder is a vector representation for each position of the input sequence.

- **Decoder**
  The Transformer decoder is also a stack of multiple identical layers with residual connections and layer normalizations. But in this case, the decoder inserts a third sublayer between these two, known as the encoder-decoder attention. In this sublayer, the queries are taken from the outputs of the previous decoder layer, and the keys and values are taken from the Transformer encoder outputs. Instead for what concerns the specific decoder self-attention, queries, keys, and values are all from the outputs of the previous decoder layer. In order to preserve the auto-regressive property, each position in the decoder is only allowed to attend to all positions in the decoder up to that position. This property ensure that the prediction only depends on those output tokens that have been generated.

  As a result, the output of the Transformer decoder is a vector of floats.

- **Last Layers**
  The vector of floats will be turned into words thanks to the final Linear layer, followed by a Softmax layer. The Linear layer is a simple fully connected neural network that produce a logits vector projecting the vector produced by the decoder. The softmax layer turns those scores into probabilities. At the end the cell with the highest probability is chosen, and the output for this time step is the word associated with that cell.

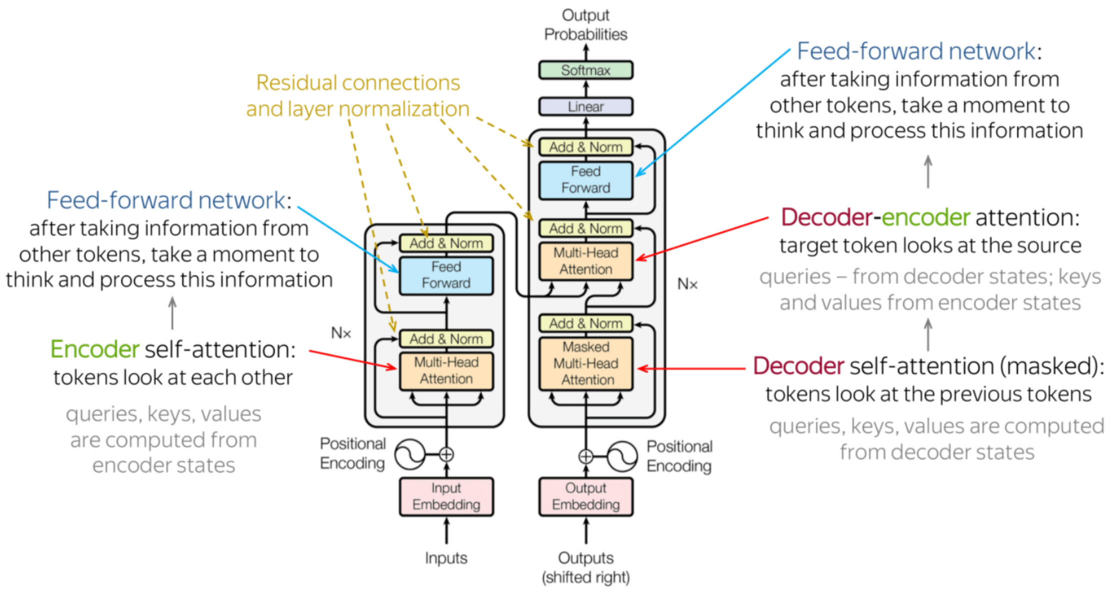

Output Probabilities
Softmax
Linear
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Add & Norm
Feed Forward
Multi-Head Attention
Masked Multi-Head Attention
Nx
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figure 2.9: Transformer model architecture [42].

2.7 Large Language Models

Before introducing the Large Language Models (LLM), we will briefly talk about a general Language Model (LM).

A LM is a tool that model the generative likelihood of word sequences in order to predict the probabilities of future or missing tokens (Figure 2.10). In practise, given a sequence the LM returns a probability distribution over sequences of tokens in a vocabulary.

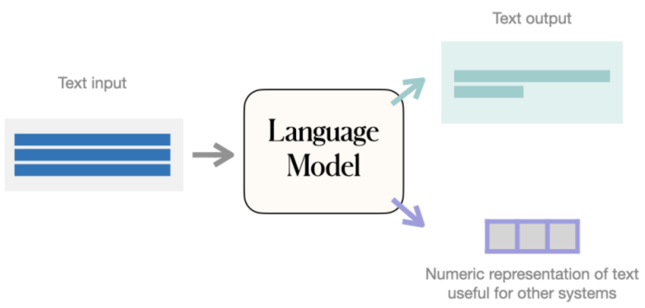


Figure 2.10: Basic functions of a Language Model [46].

From a more mathematical point of view, we have a training sequence of tokens $(y_1, ..., y_n)$ and the model prediction at the time step t will be $p^{(t)} = p(*|y_1, ..., y_{t-1})$. Since the true target, always at the same time step t, is $p^* = one - hot(y_t)$, the loss function will be $Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i)$.
However, just one p^* term is non-zero (the correct token) so the loss function will become $Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{t-1}))$.

If we want a conditional text generation, we have to use the so called **Autoregressive Language Models**, that take into consideration the conditional probability distribution $p(x_i|x_{1:i-1})$. Indeed, this conditional probabilities can be estimated by an autoregressive model (like feedforward neural network).
The method utilized is the chain rule of probability:

$$p(x_{1:N}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_N|x_{1:N-1}) = \prod_{i=1}^{N} p(x_i|x_{1:i-1})$$

Here an example in NLP field:

$$p(\mathrm{I}, \mathrm{am}, \mathrm{at}, \mathrm{home}) = p(\mathrm{I}) \cdot p(\mathrm{am}|\mathrm{I}) \cdot p(\mathrm{at}|\mathrm{I}, \mathrm{am}) \cdot p(\mathrm{home}|\mathrm{I}, \mathrm{am}, \mathrm{at})$$

Sometimes it is useful to have some sort of randomness in our model; in order to introduce it we can use the *temperature* parameter. This parameter $T > 0$ controls the randomness we want from the LM:

- $T = 0$ the most likely token is deterministically chosen at each step i
- $T = 1$ the token is chosen not deterministically but it is sampled from the pure language model using a normal distribution
- $T = \infty$ the token is not deterministic but it is sampled from the uniform distribution over the entire vocabulary

To generate the entire sequence from an autoregressive LM with $T > 0$ we sample one token at a time given the tokens generated so far, in this way:

$$\text{for } i = 1, ..., N : \quad x_i \sim p(x_i|x_{1:i-1})^{1/T}$$

Note that when T is not equal to 1 the probability distribution need to be renormalized, otherwise the probabilities do not sum to one anymore.

In the case of conditional text generation, the prefix sequence is called *prompt* while the generated part of the sequence is called *completion*.

Here an example of how this conditional text generation works. In Figure 2.11 we have a prompt composed by two words "I Like" and the estimated words from the model.

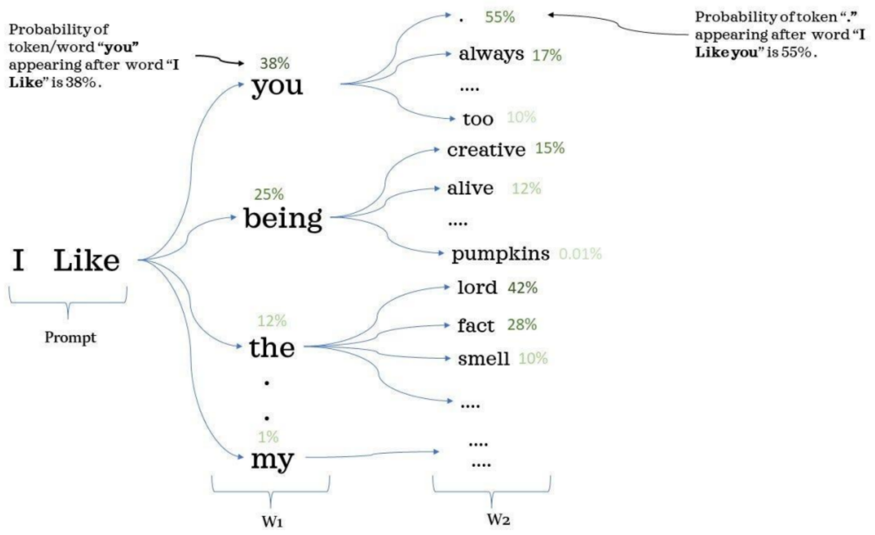

Probability of token/word "you" appearing after word "I Like" is 38%.
Probability of token "." appearing after word "I Like you" is 55%.
I Like
Prompt
38% you
25% being
12% the
1% my
. 55%
always 17%
....
too 10%
creative 15%
alive 12%
pumpkins 0.01%
lord 42%
fact 28%
smell 10%
W1
W2

Figure 2.11: Example of conditional text generation [1].

In the next two figure, we have the case with null temperature ($T = 0$) and with a positive temperature ($T > 0$).
We can see that in the first case the chosen *completion* is one composed by the most likely words, while in the second case, the output is not deterministic and the model can also chose some words that are not the most likely.

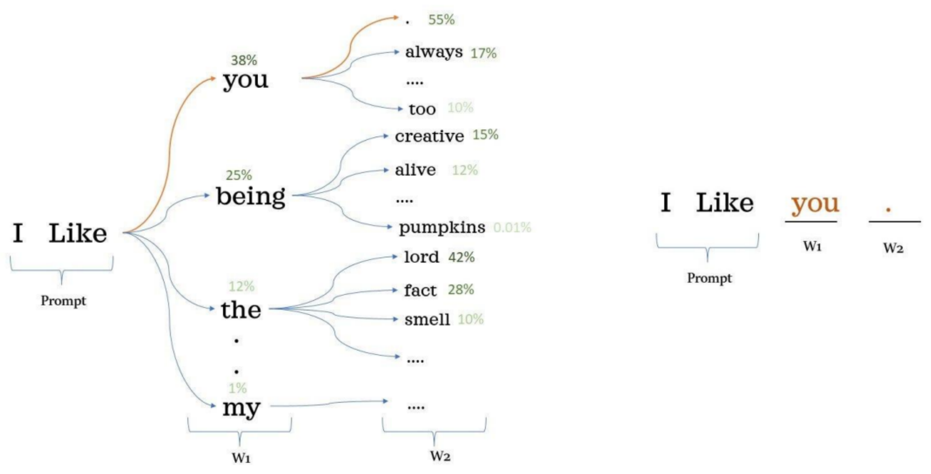


Figure 2.12: Conditional text generation with Temperature = 0 [1].

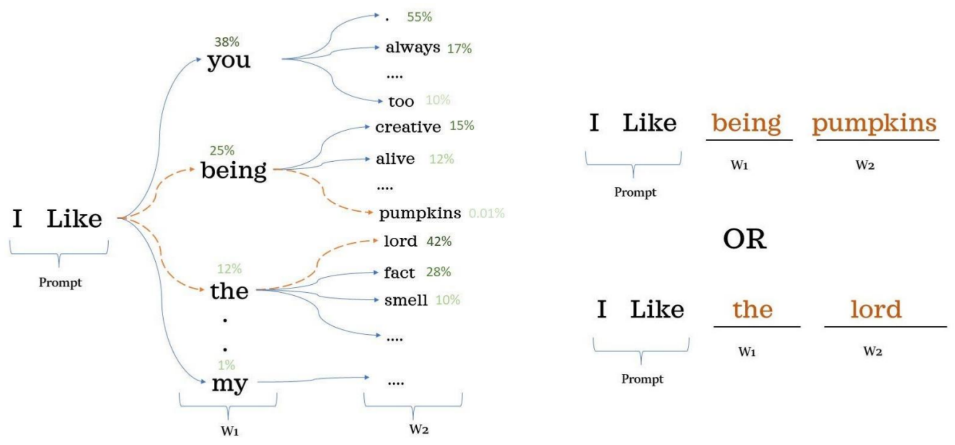


Figure 2.13: Conditional text generation with Temperature > 0 [1].

Now we can present the **Large Language Models**.

These models are a groundbreaking NLP system with advanced language understanding and generation capabilities. They enable a wide range sophisticated NLP tasks: text generation, machine translation, question answering, sentiment analysis and ability to converse. Their versatility makes them valuable for both research and practical applications.
The most famous LLMs are: GPT family (GPT-3, GPT-3.5, ChatGPT, GPT-4), LaMDA & LaMDA 2, PaLM & PaLM 2, BLOOM, Galactica, LLaMA.

These models are definined "large" since they are trained on a massive amount of data: terabytes of data so trillions of tokens. The sources from which these data are taken are extremely various in term of languages, contexts, domains, ...
According to the training data the LLM is categorized as language-specif or multilingual and as domain-specific or general-purpose.
They are considered huge deep learning models since they are characterized by tens or hundreds of billions of parameters, that enables the model to learn complex patterns and relationships within vast amounts of data. The number of the hyperparameters define the complexity of the model.

Many LLMs, including the GPT (Generative Pre-trained Transformer) series, are built on transformer architectures. As describes in the precedent section, the transformer architecture enables effective processing of sequential data through attention mechanisms, allowing the models to capture long-range dependencies.
The LLM architecture can be encoder-decoder or decoder only. All state-of-the-art language models over 100 billions of parameter are autoregressive decoder-only models. It is proved that they achieve better zero-shot performance than the encoder-decoder architectures.

– In the case of a *Causal Decoder-only* architecture, the conditioning phase is based on past token only, the next token is predicted in sequence and all tokens are processed in an equivalent manner.

– In the case of a *Non-causal Decoder-only* architecture, the self-attention mask is modified so that attention is not restricted to past tokens. In this case there is a non-causal mask that identify the parts of the input sequence that correspond to conditioning information.

– In the classic case of an *Encoder-Decoder* architecture the prediction of the target sequence is conditioned by the encoder output. It is usually based on BERT-like encoders.

LLMs typically undergo a two-step process: pre-training and fine-tuning. In the pre-training phase, the model is trained on a large corpus of text using self-supervised and/or semi-supervised learning, instead in the fine-tuning phase the model is trained on specific tasks in order to adapt it to particular applications.

There are mainly three different LLM pre-training objectives:

1. Full Language Modeling: given the previous token, the model is trained to predict the following one
2. Prefix Language Modeling: in this case, given the input, a prefix is established and on this part the attention mask is allowed to be non-causal. The model is then trained to forecast each token beyond the prefix, considering all preceding tokens.
3. Masked Language Model: some of the tokens in the input text are replaced/corrupted with a special mask token. The model is trained to predict/reconstruct the missing/corrupted tokes.

These cases are schematically illustrated in the Figure 2.14. As general rule, for pre-training the prefix is commonly chosen random while for inference the prefix is the input text.

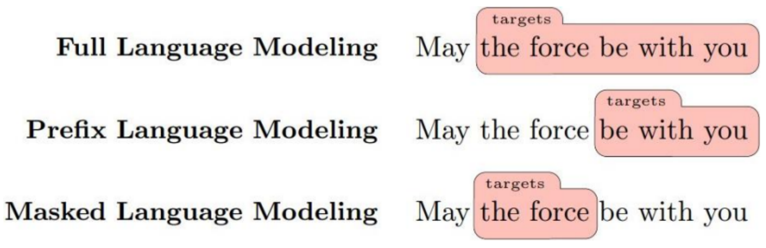


Figure 2.14: LLM pre-training objectives [43].

LLMs excel in transfer learning, where knowledge is first gained from the pre-training phase on a broad range of tasks and then is applied to new, specific tasks. This ability to transfer knowledge contributes to improve their versatility and effectiveness in various NLP applications.

Research in the field of LLMs is new and very dynamic, there is a continue ongoing of efforts to improve efficiency, reduce biases, and enhance the interpretability of these models.
Although the LLMs represent a truly powerful tool in the NLP field their development and deployment raise some concerns, for example some ethical ones for the biases present in training data and for the potential misuse of generated content. Researchers are actively working to tackle these issues and promote responsible AI

practices.

In addition, LLMs also differ in terms of being open source or proprietary, according primarily to their accessibility and ownership. Open-source LLMs, like LLaMA, allow public access to their source code, fostering collaboration and innovation within the developer community and promoting transparency and customization. In contrast, proprietary LLMs, like GPT-4, are owned by specific companies, limiting access to the underlying code and often requiring licensing fees. They may offer additional support, security and exclusive features, all at the expense of the owning organization.

2.7.1 LLM Adaptation

Adaptation is the process of customizing or fine-tuning a LM o LLM to better suit specific tasks, domains, or datasets. It is a powerful approach to leverage the general language understanding capabilities of pre-trained models while tailoring them to specific use cases, thereby enhancing their effectiveness and applicability in various domains.
The general schema of this process is in Figure 2.15, it takes as input, a natural language description and a set of training instances (input-output pairs).

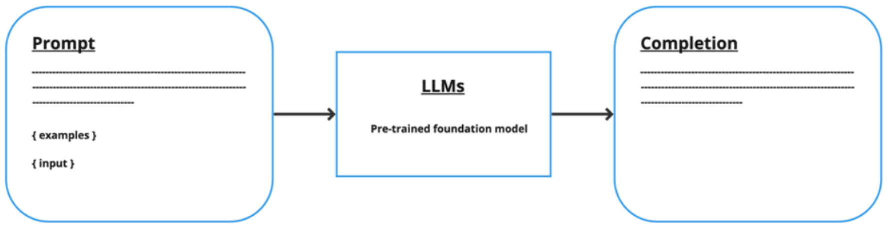


Figure 2.15: Schema of a LLM adaptation process [34].

In our analysis, we will consider two main aspects: training and prompting.

Training

In this section we will talk about training via supervised learning, that involves providing a model with labeled examples, enabling it to learn patterns and associations between inputs and corresponding desired outputs.
LLM adaptation through training poses challenges, particularly in dealing with data overfitting due to the limited number of training instances relative to the network complexity. Another problem arises from the task-agnostic nature of LLMs during pre-training, where they are trained on general-purpose. Instead,

downstream tasks, can vary significantly, and the generic next token prediction can be not sufficient.
To address these challenges, several key strategies are employed:

- *Formatting*: involves utilizing a specific formatting style tailored to the downstream task, ensuring the model comprehends task-specific structures.
- *Topic shift*: Topic shift adaptation customizes the LLM to a specific domain, aligning it more closely with the task requirements.
- *Temporal shift*: temporal shift adaptation is employed when the downstream task necessitates a new model that was not available during the pre-training phase.

The principal methods to training via a supervised learning are these:

- *Probing*
  Developing a task-specific model utilizing the language model as latent features by training a prediction head on the top of a frozen language model, extracting information from the last layer of representation to generate output.
- *Fine-tuning*
  Starting the training process with the language model and updating it based on specific instances, using the language model parameters as initialization for optimization. This is an expensive procedure because it necessitates storing a specialized LLM for each downstream task. But, despite being more expensive, it proves to be more effective than probing, often leveraging reinforcement learning to incorporate human preferences.
- *Lightweight fine-tuning*
  Optimizing only a select few parameters, this method combines aspects of both fine-tuning and probing, achieving comparable expressivity to full fine-tuning without the need to store the complete language model for each task.

Prompting

In this context, "prompting" refers to the way input is provided to the model to generate desired outputs. It involves presenting a specific instruction or query, often referred to as a "prompt", to guide the model in producing the intended response, called "completion". The choice and formulation of prompts play a crucial role in shaping the model's behavior and output.
Prompting can involve providing a specific question or statement to the language

model, and the model generates a relevant response based on its learned patterns and understanding. The effectiveness of prompting is essential in fine-tuning LLMs for specific tasks, as it helps tailor the model's behavior to meet the requirements of a given application or domain. Adjusting prompts allows researchers and practitioners to influence the model's output and steer it towards desired outcomes. In Figure 2.16 there is an example of multi-task prompting for text summarization and machine translation.

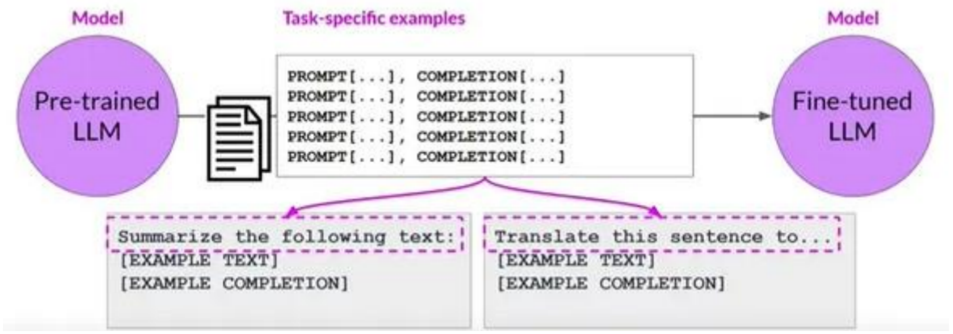

Model
Task-specific examples
Model
Pre-trained LLM
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
Fine-tuned LLM
Summarize the following text:
[EXAMPLE TEXT]
[EXAMPLE COMPLETION]
Translate this sentence to...
[EXAMPLE TEXT]
[EXAMPLE COMPLETION]

Figure 2.16: Prompting on multi-tasks [20].

There are basically two ways of prompting, the zero-shot and the few-shot, based on the number of examples provided in the prompt.

In *zero-shot prompting*, the prompt is presented with a task or query without any specific examples or training instances related to that task. The model is expected to generalize its pre-existing knowledge and learned patterns to generate a response or perform the given task. This method relies solely on the model's ability to transfer its understanding from the pre-training phase to novel tasks, demonstrating its capacity for broad applicability and adaptability.

Otherwise, *few-shot prompting* involves providing the prompt with a limited number of examples or instances related to the task at hand. These examples serve as a partial training set, offering the model additional context and guidance for the specific task. Few-shot prompting is a middle ground between zero-shot and full fine-tuning, allowing for task-specific adaptation while leveraging the pre-trained knowledge of the model. It strikes a balance between generality and task specificity, making it a valuable approach for various applications where a small amount of task-specific information is available.

In addition, there is another prompting method known as the *Chain-of-Thought prompting* presented in [44]. This technique involves crafting prompts to guide the model through a logical flow of reasoning, aiming to refine the model's understanding by leading it through a structured process for more nuanced and context-aware responses. The sequential nature of Chain-of-Thought prompting simulates a logical reasoning process, enhancing the model's ability to generate coherent and contextually relevant outputs. The goal is to induce the model to engage in complex reasoning capabilities through intermediate steps. This method proves particularly useful for tasks such as tackling complex arithmetic and understanding commonsense. An example is present is Figure 2.17.

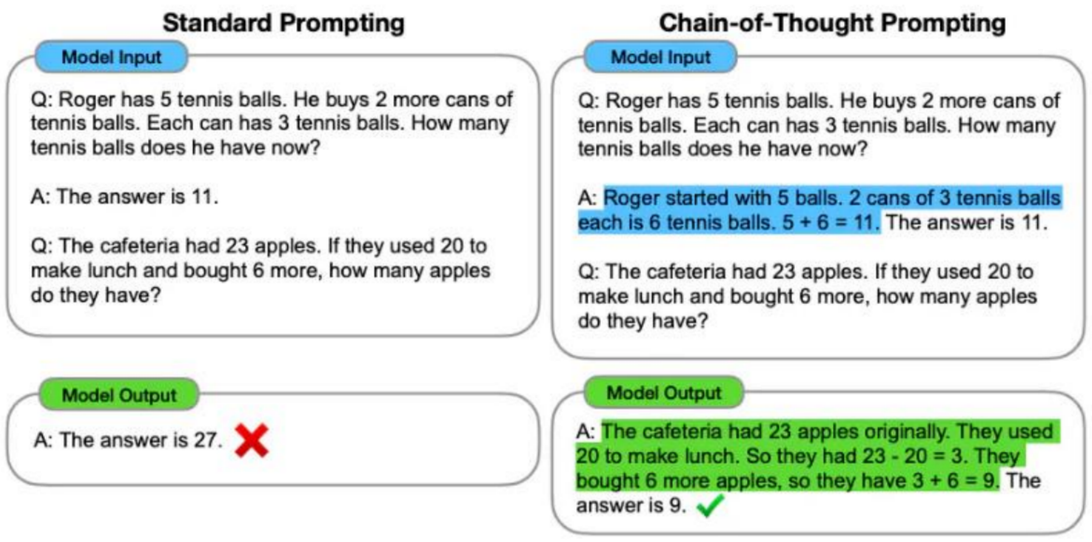

Standard Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27.
Chain-of-Thought Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

Figure 2.17: Example of a Chain-Of-Thought prompting [46].

2.7.2 LLaMA 2

In this section, we will present the LLM utilized for our experiments, LLaMA 2. The name LLaMA is derived from the Spanish phrase "Lenguaje de Computadora Asistida por Medios Aprendizaje" (Computer-Assisted Language Learning), which reflects the model's ability to learn and improve through large amounts of text data.

This language model is detailed in [40] and consists of a collection of pre-trained and fine-tuned LLMs where the range of the parameters is between 7 and 70 billion. *LLaMA 2* is an updated version of LLaMA 1 [39], trained on a new combination of publicly accessible data. Additionally, the size of the pretraining corpus is augmented by 40%, doubled the model's context length, and a grouped-query attention has been implemented.
LLaMA 2-Chat is a refined version of LLaMA 2 fine-tuned to excel in dialogue use cases, Figure 2.18.
Both for LLaMA 2 and LLaMA 2-Chat, three variants have been released, each with 7B, 13B, and 70B parameters, respectively. A 34B variant has also been trained for LLaMA 2, but it has not been released yet.

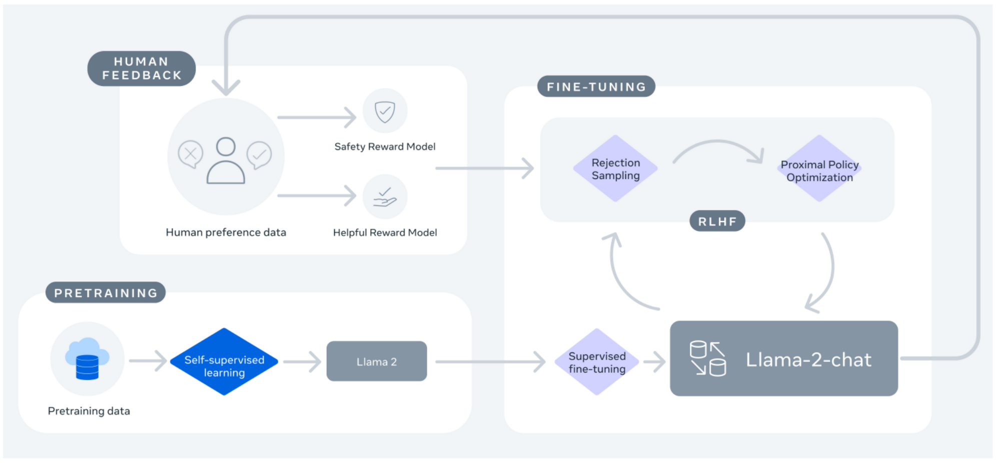
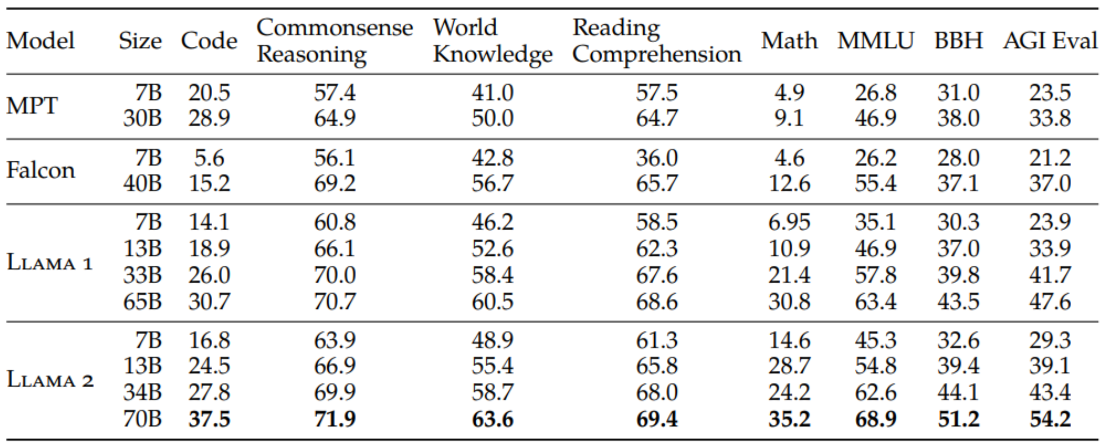


Figure 2.18: Training of LLaMA 2-Chat [40].

Performance

We have chosen this model because it is an open-source solution that consistently outperforms other open-source chat models on various benchmarks.

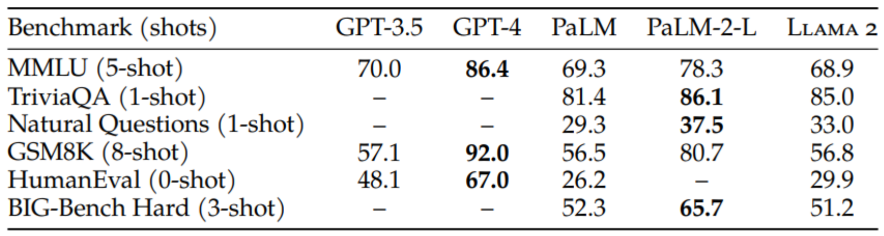
In Table 2.19 are reported the results, on standard academic benchmarks, for the LLaMA 1 and LLaMA 2 base models and other two open-source models: MosaicML Pretrained Transformer (MPT) and Falcon models. The chosen benchmarks are grouped into these categories: Code, Commonsense Reasoning, World Knowledge, Reading Comprehension, Math, Popular Aggregated benchmarrks (MMLU, BBH, AGI Eval). The evaluations present in the table are done utilizing the LLaMA internal evaluation library. From the table we can clearly see that LLaMA 2 outperform LLaMA 1 and the other open-source LMs



Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
Llama 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
Llama 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	**37.5**	**71.9**	**63.6**	**69.4**	**35.2**	**68.9**	**51.2**	**54.2**

Figure 2.19: Comparison with open-source base models: overall performance across various grouped academic benchmarks [40].

In addition, we can also compare LLaMA2 70B, with some closed-source models like GPT 3.5, GPT4, PaLM and PaLM-2-L. But, as shown in Table 2.20, in this case there is a significant gap in some performance.



Benchmark (shots)	GPT-3.5	GPT-4	PaLM	PaLM-2-L	Llama 2
MMLU (5-shot)	70.0	**86.4**	69.3	78.3	68.9
TriviaQA (1-shot)	–	–	81.4	**86.1**	85.0
Natural Questions (1-shot)	–	–	29.3	**37.5**	33.0
GSM8K (8-shot)	57.1	**92.0**	56.5	80.7	56.8
HumanEval (0-shot)	48.1	**67.0**	26.2	–	29.9
BIG-Bench Hard (3-shot)	–	–	52.3	**65.7**	51.2

Figure 2.20: Comparison with closed-source base models: performance on academic benchmarks [40].

Especially for natural generation tasks, human assessment is considered the ultimate benchmark for evaluating models, their evaluation is the so-called 'gold' one. Therefore, to assess the quality of major dialogue model versions, the author

of LLaMA 2 enlisted human evaluators to rate LLaMA 2-Chat against other language models. The comparison involved over 4,000 single and multi-turn prompts, including both open-source models (Falcon, MPT, Vicuna) and closed-source models (Chat-GPT, PaLM).
The results are shown in Figure 2.21, and we can see that LLaMA 2-Chat models show superior performance, also with a significant margin, compared to open-source models across both single-turn and multi-turn prompts [40].

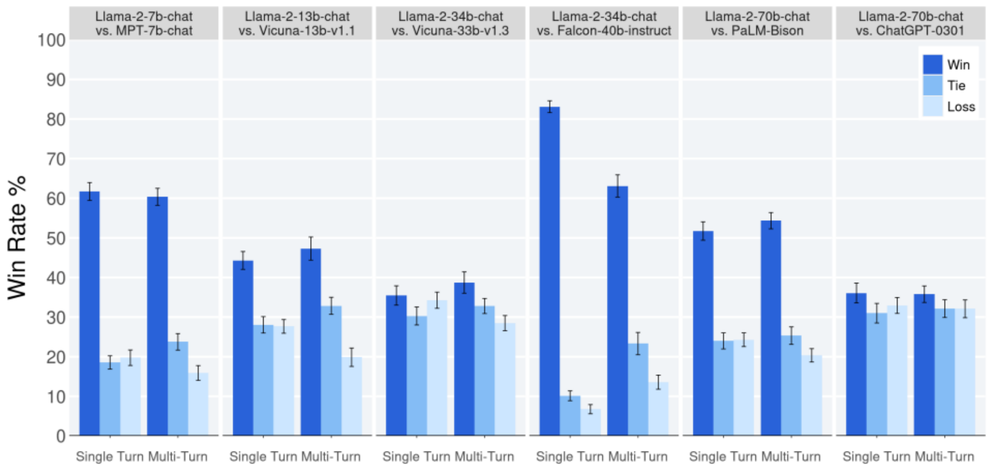


Figure 2.21: Comparison with open and closed source base models: human evaluation results [40].

Pre-training

The development of the LLaMA 2 models is based on pretraining method outlined in LLaMA 1 [39]. The authors optimized that auto-regressive transformer, implementing various modifications to enhance performance. These included conducting a more robust data cleaning, revising data combinations, training on 40% more total tokens, doubling the context length, now 4096, and integrating grouped-query attention (GQA) to improve inference scalability.

The training corpus for this model consists of a fresh blend of data from publicly accessible sources, deliberately excluding any data from Meta's products or services. Special attention was given to filtering out data from sites with a high concentration of personal information. Training was conducted on a substantial 2 trillion tokens of data, striking a balance between performance and cost.
The study adopts the pre-training settings and model architecture largely from LLaMA 1, employing the standard auto-regressive transformer architecture with

pre-normalization using RMSNorm, SwiGLU activation function, and rotary positional embeddings.
Also, the tokenizer is the same as LLaMA 1, and the final vocabulary size is 32k tokens.

In Figure 2.22 we show the training loss for LLaMA 2 models. We can observe that there is not yet any sign of saturation after pre-training 2 trillion tokens.

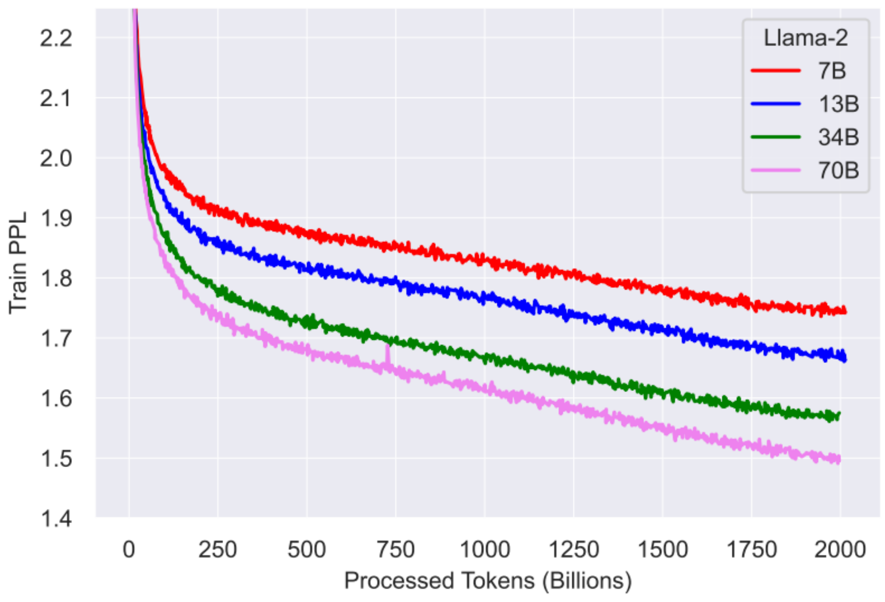


Figure 2.22: Training Loss for LLaMA 2 models [40].

Fine-tuning

LLaMA 2-Chat was developed over many months of research and continuous refinement using various alignment techniques, such as instruction tuning and RLHF. This process demanded considerable computational power and annotation efforts. To fine-tune this model, various techniques were employed, including supervised fine-tuning (SFT), iterative reward modeling, Reinforcement Learning with Human Feedback (RLHF), and the Ghost Attention, a novel technique.

The authors of LLaMA 2 started the **Supervised Fine-Tuning (SFT)** stage with publicly available instruction tuning data, as utilized previously in LLaMA 1.
As first step they focused on collecting several examples of high-quality SFT data. They prioritized quality rather than quantity, for example from the third-party

datasets they excluded millions of examples due insufficient diversity and quality and selected only the best ones. As explained in [47], a limited set of clean instruction-tuning data can be sufficient to achieve a high level of quality. In the case of LLaMA 2, using a set of 27,540 SFT annotations proved to be adequate.
To confirm the quality of the data, the authors reviewed a set of 180 examples, conducting a manual comparison between the annotations produced by humans and those generated by the model. Interestingly, they discovered that the outputs sampled from the resulting SFT model frequently demonstrated competitiveness with SFT data manually created by human annotators. This observation indicates the potential to reconsider and allocate additional annotation efforts toward preference-based annotation for RLHF.

The **Reinforcement Learning with Human Feedback (RLHF)** is a procedure of model training that involves fine-tuning a language model to better match human preferences and instructions.
Data is collected to represent empirically sampled human preferences, with human annotators choosing their preferred output among two model-generated options. This human feedback is then utilized to train a reward model, enabling the learning of patterns in the preferences of human annotators and automating subsequent preference decisions.
The procedure is imposed as follows. The annotators first write a prompt and then choose between the two model responses. In addition to this selection, they are also required to provide the degree to which they prefer their choice over the alternative, choosing between: *significantly better*, *better*, *slightly better*, or *negligibly better/ unsure*.
Given a prompt, two different responses are sampled from two model variants, where the temperature hyper-parameter varies.

In this evaluation the focus is also on helpfulness and safety.
Helpfulness pertains to how effectively responses fulfill users' requests and provide requested information. Safety involves determining whether LLaMA 2-Chat's responses are unsafe; for instance, providing detailed instructions on making a bomb could be considered helpful but is unsafe according to safety guidelines.
A safety label is collected during the safety stage and the model is categorized into one of these three categories: both responses are unsafe, both responses are safe or one response is safe (the preferred one) while the other not.

The **reward model** evaluates the quality of a model's response by taking into account both the response itself and the prompt it was given, including any relevant context from previous interactions. It then assigns a numerical score to indicate aspects such as helpfulness and safety.

By leveraging response scores as rewards, LLaMA 2-Chat can be optimized during RLHF to achieve better human preference alignment, as well as improved helpfulness and safety.
Researchers have observed that helpfulness and safety can occasionally exhibit a trade-off, posing a challenge for a singular reward model to excel in both aspects. In response to this challenge, two distinct reward models were trained: one optimized for helpfulness (*Helpfulness RM*) and another for safety (*Safety RM*).

The authors made an analysis of the *scaling trends* in the reward model, in terms of data and model size. The outcomes are showed in Figure 2.23, as expected larger models achieve higher performance with a comparable data volume. Notably, the scaling performance has not reached a plateau with the current training data, indicating potential for further enhancement with additional annotations.

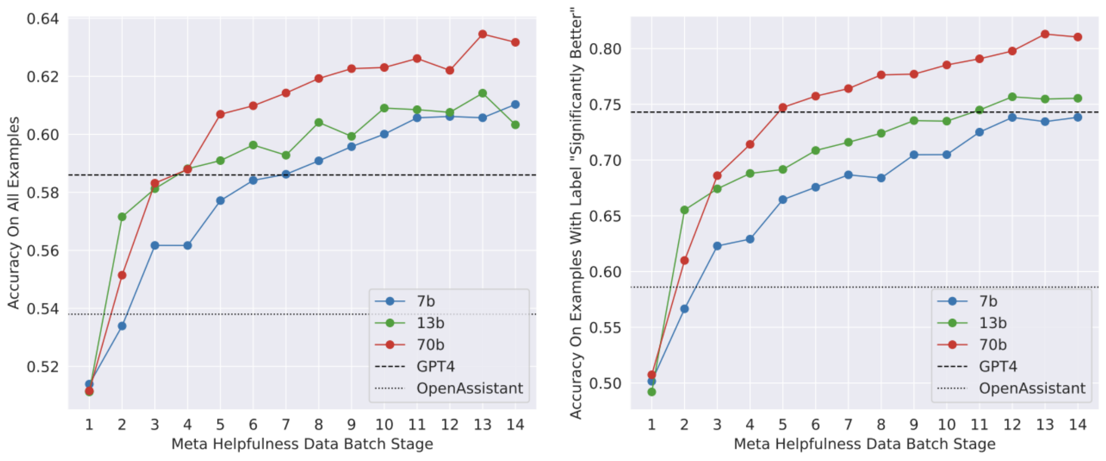


Figure 2.23: Scaling trends for the reward model [40].

To train the reward model, the gathered human preference data is transformed into a binary ranking system, indicating whether each option was chosen or rejected and ensuring that the chosen option consistently receives a higher score than its counterpart.
Given Σ the model weights, we can write $r_\Sigma(x, y)$ as the scalar score output for prompt x and completion y. Given y_c the chosen response and y_r the rejected one, we can write the binary ranking loss as follows.

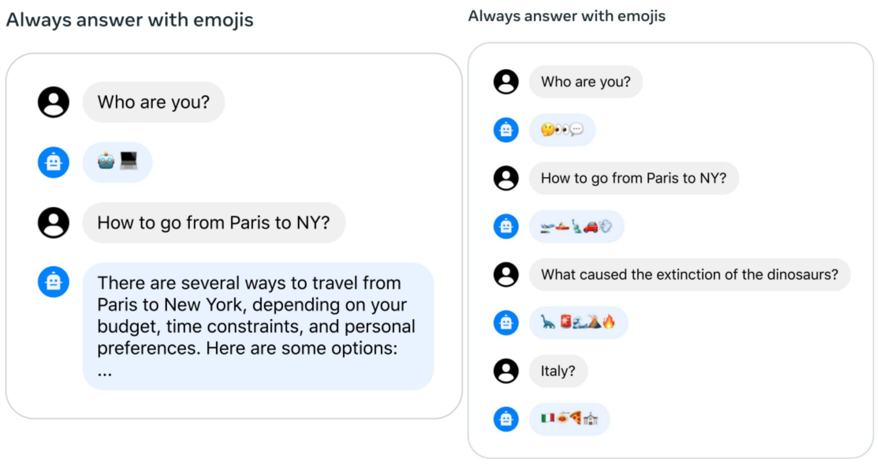
$$\mathcal{L}_{ranking} = -\log(\sigma(r_\Sigma(x, y_c) - r_\Sigma(x, y_r)))$$

The preference ratings, categorized into a four-point scale (e.g., *significantly better*), are leveraged to explicitly guide the reward model in order to assign more discrepant scores to generations with greater differences.

To do so, an additional margin component $m(r)$, that is a discrete function of the preference rating, is introduced in the loss.

$$\mathcal{L}_{ranking} = -\log(\sigma(r_\Sigma(x, y_c) - r_\Sigma(x, y_r) - m(r)))$$

In a dialogue setting, certain instructions are intended to be applicable across all conversation turns. When these instructions were given to LLaMA 2-Chat, the subsequent responses were expected to consistently respect the constraint. As shown in Figure 2.24, after a few turns of dialogue, the RLHF models often show a tendency to lose the initial instruction.
The **Ghost Attention (GAtt)** is a simple method that can address this limitation. It manipulate the fine-tuning data to guide the attention focus through a multi-stage process. GAtt empowers dialogue control across multiple turns, as depicted in Figure 2.24.


Always answer with emojis

Who are you?

How to go from Paris to NY?

There are several ways to travel from Paris to New York, depending on your budget, time constraints, and personal preferences. Here are some options:
...

Always answer with emojis

Who are you?

How to go from Paris to NY?

What caused the extinction of the dinosaurs?

Italy?

Figure 2.24: Left: Issues with multi-turn memory. Right: improvement with GAtt [40].

3 Figurative Language Understanding

In this chapter, we will talk about some aspects of one of the most challenging tasks in the NLP field: Figurative Language Understanding.
Following an initial introductory section, we will present the *FLUTE* dataset, which is the dataset utilized in our experiments. Subsequently, we will delve into the FigLang 2022 competition, known as the Shared Task on Understanding Figurative Language, which served as the cornerstone for our thesis research. Finally, we will introduce the *DREAM* method, a technique that we will utilize in the latter part of our research.

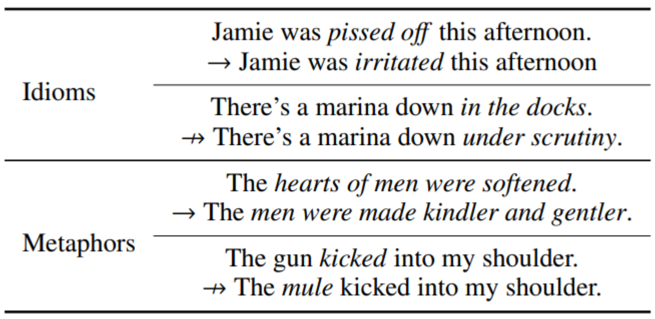
3.1 Introduction

Figurative language is widely present across various types of communication, including novels, poems, films, scientific literature, and social media discussions. Its common usage aims to express closeness, humor, strong emotions, or veiled politeness.
The comprehension of figurative language poses a formidable challenge in NLP, as the intended meaning of an utterance often diverges significantly from the literal interpretation of its constituent words. Figurative language, encompassing elements like metaphors, similes, and sarcasm, holds a crucial role in enriching human communication by enabling the implicit expression of complex ideas and emotions. Despite the advancements in Transformer-based LMs, which have grown in scale, they still struggle to grasp the nuances of the physical world, cultural knowledge, and the social context embedded in figurative language.

In recent years, there has been a concerted effort to develop benchmarks specifically dedicated to figurative language understanding. Typically these benchmarks focus on Natural Language Inference (NLI), in particular on the Recognizing Textual Entailment (RTE), i.e. the task of determining whether a given sentence (context/premise) is likely to entail another (hypothesis).
As an illustration, the authors of [6], leveraged five pre-existing datasets, annotated for different forms of figurative language, to introduce a collection of RTE datasets specifically centered around the theme of figurative language. In this specific case the authors investigated three specific types of rhetorical figures: similes,

metaphors, and irony.
The authors evaluated the ability of standard neural RTE models in capturing these aspects of figurative language and the results demonstrated that, although, systems may capture some aspects of various figures of speech, they often struggle in situations where the interpretation hinges on pragmatic inference and reasoning about worldly knowledge.

One important building block in this field is the IMPLI (Idiomatic and Metaphoric Paired Language Inference) dataset, [37]. It is a collection of paired english sentences spanning idioms and metaphors.
The dataset encompasses both silver pairs (24k), generated through semi-automated methods, as well as hand-crafted gold pairs (1.8k). Both are designed to represent both entailment and non-entailment situations. Each pair comprises a sentence containing a figurative expression (idioms/metaphors) and a corresponding literal counterpart, formulated to be either entailed or non-entailed by the figurative expression. In Figure 3.1 there are some examples .



Idioms	Jamie was *pissed off* this afternoon. → Jamie was *irritated* this afternoon
	There's a marina down *in the docks*. ↛ There's a marina down *under scrutiny*.
Metaphors	The *hearts of men were softened*. → The *men were made kindler and gentler*.
	The gun *kicked* into my shoulder. ↛ The *mule* kicked into my shoulder.

Figure 3.1: Examples of entailment (→) and non-entailment pairs (↛) from the IMPLI datase [37].

The researchers employ IMPLI to assess NLI models, specifically those derived from fine-tuning RoBERTa on the extensively utilized MNLI dataset. Subsequently, the study demonstrates that these models exhibit proficiency in accurately identifying entailment relationships between figurative expressions and their literal equivalents. However, their performance diminishes notably when confronted with similarly structured examples intentionally designed to be non-entailing pairs.

Similar to general NLI datasets, these benchmarks are susceptible to spurious correlations and annotation artifacts, leading to challenges for LLMs that exhibit near-human performance in in-domain scenarios but become brittle when faced with out-of-domain or adversarial examples.
To address these issues, the research in NLI emphasizes the importance of not only

accurately predicting entailment/contradiction labels but also providing natural language explanations that enhance the interpretability of the model's decisions for end-users. This approach has led to the creation of novel datasets, such as e-SNLI [5], which incorporates natural language explanations.

In response to this gap, in [7] was introduced *FLUTE*, a dataset consisting of 9,000 instances of figurative NLI with accompanying explanations. The dataset spans four categories: Sarcasm, Simile, Metaphor, and Idioms, providing a valuable resource for evaluating the true understanding of expressions within the realm of figurative language.

3.2 FLUTE Dataset

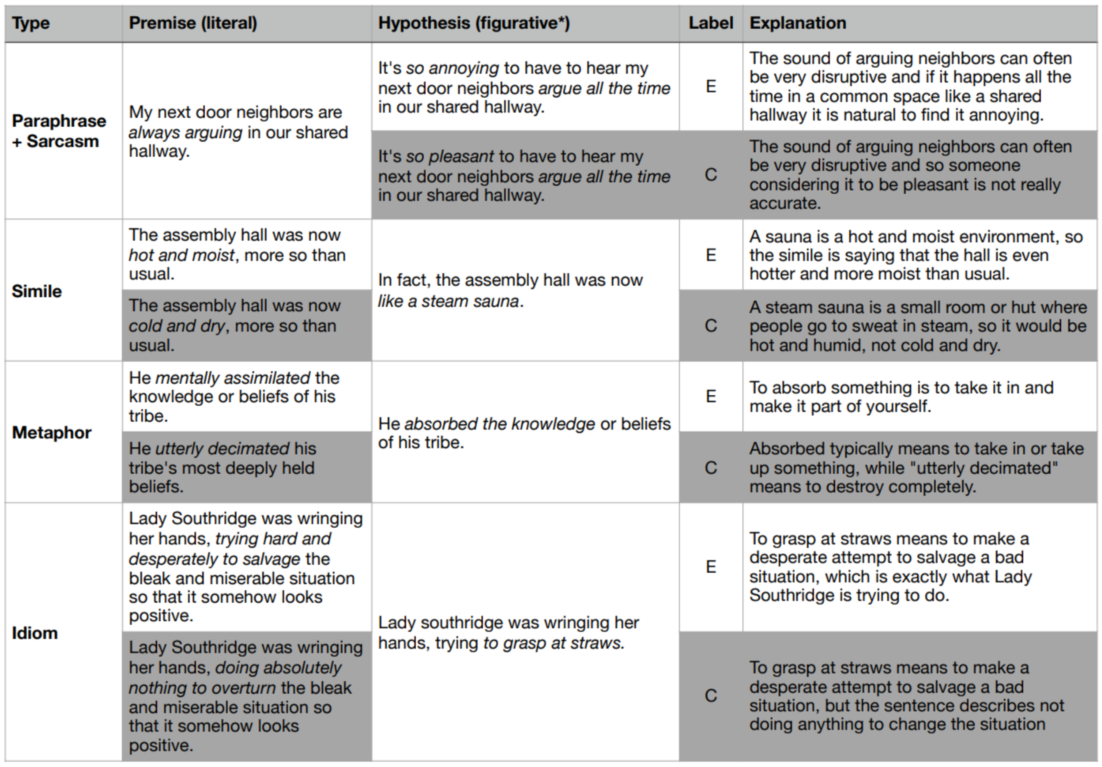
As discussed earlier, the comprehension of figurative language has been cast within the framework of NLI, in particular RTE, but, existing benchmarks exhibit issues like spurious correlations and annotation artifacts. To address this challenge, efforts in NLI have generated explanation-based datasets like e-SNLI, aiming to scrutinize the validity of language models' reasoning. Unfortunately, there is a lack of comparable data for figurative language, hindering the evaluation of genuine understanding of such expressions. In response to this gap, Chakrabarty et al. have introduced *FLUTE* dataset [7].

This dataset comprises about 9,000 high-quality literal, figurative sentence pairs, each accompanied by labels indicating entailment or contradiction, along with corresponding explanations. The benchmark covers four categories of figurative language: Sarcasm, Simile, Metaphor, and Idiom.
A table with an example for each type of figure is present in Figure 3.2, while the distribution inside the dataset of such rhetorical figures is presented in Table 3.1. Please note that since sarcasm conveys the opposite of the literal meaning, the dataset should exclusively contains contradictions. Consequently, the authors also produce a literal hypothesis that aligns with the literal premise.

Type	**Entails**	**Contradicts**	**Total**
Sarcasm	1339	2678	4017
Simile	750	750	1500
Metaphor	750	750	1500
Idiom	1000	1000	2000
	3839	5178	9017

Table 3.1: FLUTE dataset distribution

The dataset was build by employing with a scalable *model-in-the-loop* approach, that is a combination of few-shot prompting with GPT-3 and crowdsourcing, in this case from Amazon Mechanical Turk (AMT).
In the context of figurative language, the authors of [14] demonstrated that crowdworkers excel at making minimal edits to transform a sarcastic sentence into a literal one, often employing techniques like negation or antonyms. However, this ease of transformation can result in examples that are easily categorized by LLMs. To construct *FLUTE*, the authors relied on the capabilities of GPT-3 to generate a wide range of high-quality literal text, including paraphrases, contradictions, and explanations. Specifically, they did few-shot prompting with minimal human involvement, such as crowdworkers making slight adjustments to convert a literal

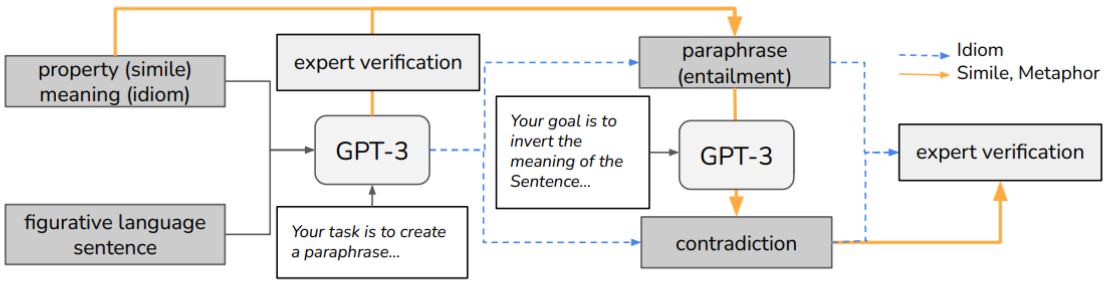


Type	Premise (literal)	Hypothesis (figurative*)	Label	Explanation
Paraphrase + Sarcasm	My next door neighbors are *always arguing* in our shared hallway.	It's *so annoying* to have to hear my next door neighbors *argue all the time* in our shared hallway.	E	The sound of arguing neighbors can often be very disruptive and if it happens all the time in a common space like a shared hallway it is natural to find it annoying.
		It's *so pleasant* to have to hear my next door neighbors *argue all the time* in our shared hallway.	C	The sound of arguing neighbors can often be very disruptive and so someone considering it to be pleasant is not really accurate.
Simile	The assembly hall was now *hot and moist*, more so than usual.	In fact, the assembly hall was now *like a steam sauna*.	E	A sauna is a hot and moist environment, so the simile is saying that the hall is even hotter and more moist than usual.
	The assembly hall was now *cold and dry*, more so than usual.		C	A steam sauna is a small room or hut where people go to sweat in steam, so it would be hot and humid, not cold and dry.
Metaphor	He *mentally assimilated* the knowledge or beliefs of his tribe.	He *absorbed the knowledge* or beliefs of his tribe.	E	To absorb something is to take it in and make it part of yourself.
	He *utterly decimated* his tribe's most deeply held beliefs.		C	Absorbed typically means to take in or take up something, while "utterly decimated" means to destroy completely.
Idiom	Lady Southridge was wringing her hands, *trying hard and desperately to salvage* the bleak and miserable situation so that it somehow looks positive.	Lady southridge was wringing her hands, trying *to grasp at straws*.	E	To grasp at straws means to make a desperate attempt to salvage a bad situation, which is exactly what Lady Southridge is trying to do.
	Lady Southridge was wringing her hands, *doing absolutely nothing to overturn* the bleak and miserable situation so that it somehow looks positive.		C	To grasp at straws means to make a desperate attempt to salvage a bad situation, but the sentence describes not doing anything to change the situation

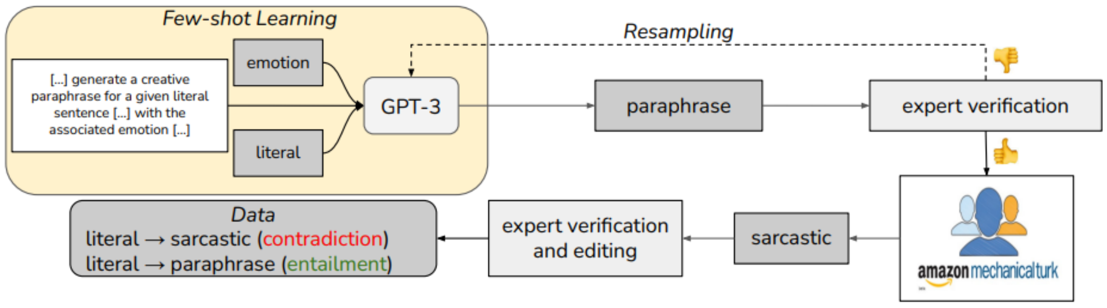
Figure 3.2: Examples from FLUTE: For each hypothesis (figurative text) are reported two premises, one is the literal entailment (E) and the other one is the contradiction (C). There are also the associated explanations [7].

sentence into a sarcastic one and experts overseeing and making minimal edits to the GPT-3 output to ensure quality control.

As said before, *FLUTE* consists of pairs of <premises, hypothesis>, i.e. <literal sentence, figurative sentence> with the corresponding label, entailment or contradiction, and the explanation. To create all these data, premise-hypothesis pairs for each type of figurative language and the associated explanations, the *model-in-the-loop* method is used. The technique differs slightly regarding sarcasm compared to the other three rhetorical figures.
The following figures illustrate the model's schema in both cases.



(a) Model in the Loop for FLUTE: Simile, Idiom, Metaphor [7]


Few-shot Learning
[...] generate a creative paraphrase for a given literal sentence [...] with the associated emotion [...]
emotion
literal
GPT-3
Resampling
paraphrase
expert verification
amazon mechanical turk
sarcastic
expert verification and editing
Data
literal → sarcastic (contradiction)
literal → paraphrase (entailment)

(b) Model in the Loop for FLUTE: Sarcasm [7]

3.3 FigLang 2022 Shared Task on Understanding Figurative Language

In the Third Workshop on Figurative Language Processing at EMNLP 2022 (FigLang 2022); Tuhin Chakrabarty, Arkadiy Saakyan, Debanjan Ghosh and Smaranda Muresan proposed a shared task on Understanding Figurative Language [8].

Over the years, numerous benchmarks have focused on understanding figurative language, typically treating "understanding" as a task of RTE, determining whether one sentence (premise) entails or contradicts another (hypothesis). In this instance, a novel task has been introduced, necessitating not only the generation of the label (entail/contradict) but also the generation of a plausible explanation for the prediction.
The hypothesis consists of a sentence incorporating figurative language expressions (such as metaphor, sarcasm, idiom, or simile), while the premise is a literal sentence conveying the literal meaning. Both the entail/contradict label and the explanation are associated with the interpretation of the figurative language expression.

For instance given a
Premise: His heart within him is fully rotten.
Hypothesis: And his heart within him fluttered.
we need an output that consists of a
1) Label = Contradiction
2) Explanation = Fluttering would suggest that the man's heart is beating rapidly, while rotten would suggest that the man's heart is dead or no longer functioning.

The shared task is based on the *FLUTE* dataset relased by [7], comprising NLI pairs <premise, hypothesis> incorporating figurative language. Each NLI instance in the dataset is accompanied by free-text explanation. From this dataset, the challenge organizers have built a Test set of 1500 examples, by randomly selecting 750 instances from the sarcasm datasets, and 250 examples each from simile, metaphor and idiom datasets. They have left the division of the remaining data between the Train and Validation sets arbitrary.

One important feature of this data is that the labels and explanations are specifically related to the figurative language employed (i.e., metaphor, simile, idiom), rather than other components of the sentences, as you can see from figure 3.4.

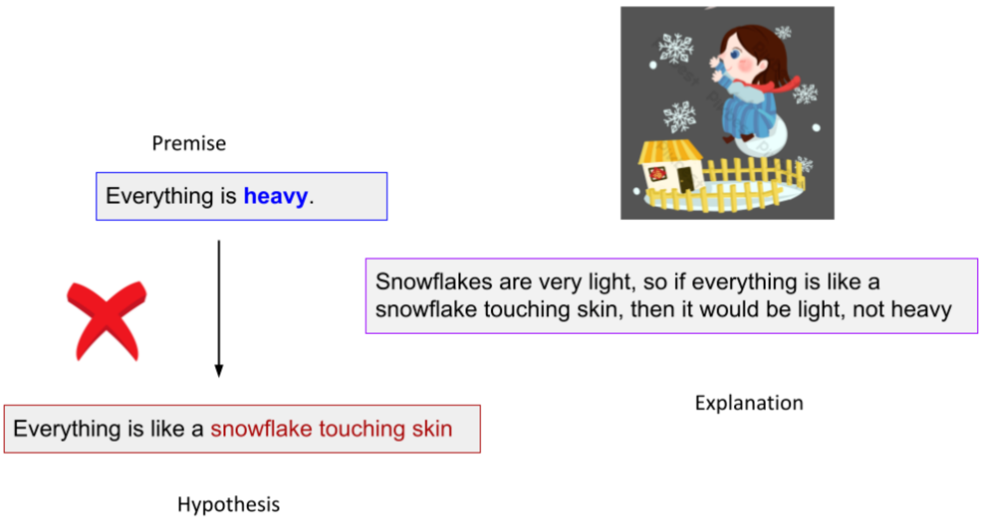

Premise
Everything is heavy.
Everything is like a snowflake touching skin
Hypothesis
Snowflakes are very light, so if everything is like a snowflake touching skin, then it would be light, not heavy
Explanation

Figure 3.4: Example of how the explanations were constructed [8].

The dataset presents challenges inherent to grasping figurative language, demanding both relational reasoning with background commonsense knowledge and a detailed understanding of the language's nuances.
Figurative expressions are frequently complex, introducing implicit meanings that necessitate multiple reasoning steps for interpretation. In the instance in Figure 3.5, the hypothesis conveys *sarcasm*, and to grasp the contradiction between the literal premise and the sarcastic hypothesis, model must reason step-by-step thinking about everyday concepts using common-sense knowledge.
In the next figure 3.6, we have an example of an *idiom* and the model is supposed to possess the capability to understand figurative expressions that are unseen, and these expressions may be non-compositional.

To validate the effectiveness of the dataset, the author conducted a series of experiments aimed at developing models capable of understanding figurative language. As part of the setup, they trained a T5 model to simultaneously predict the label (entail/contradict) and provide an explanation. They compared two scenarios: one where T5 was trained on the e-SNLI dataset and another where it was trained on *FLUTE*. The results demonstrated that the model trained on *FLUTE* produced higher-quality explanations compared to the other one.

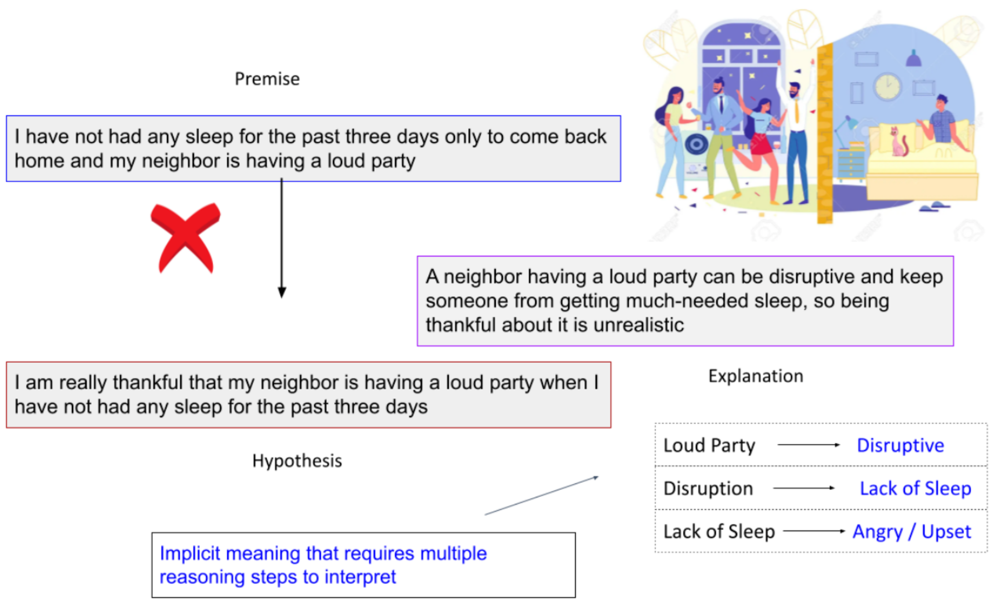

Premise
I have not had any sleep for the past three days only to come back home and my neighbor is having a loud party
A neighbor having a loud party can be disruptive and keep someone from getting much-needed sleep, so being thankful about it is unrealistic
I am really thankful that my neighbor is having a loud party when I have not had any sleep for the past three days
Explanation
Hypothesis
Loud Party → Disruptive
Disruption → Lack of Sleep
Lack of Sleep → Angry / Upset
Implicit meaning that requires multiple reasoning steps to interpret

Figure 3.5: Sarcasm example [8].

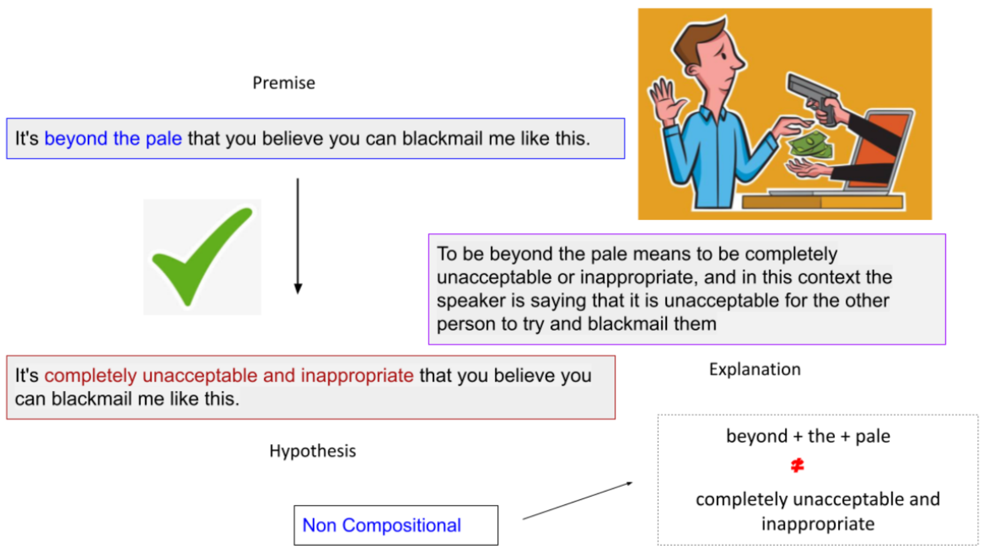

Premise
It's beyond the pale that you believe you can blackmail me like this.
To be beyond the pale means to be completely unacceptable or inappropriate, and in this context the speaker is saying that it is unacceptable for the other person to try and blackmail them
It's completely unacceptable and inappropriate that you believe you can blackmail me like this.
Explanation
Hypothesis
beyond + the + pale
≠
completely unacceptable and inappropriate
Non Compositional

Figure 3.6: Idiom example [8].

3.3.1 Participants and Results

The participants at the FigLang 2022 were able to improve upon provided baselines and in this section we are going to briefly present some of the solutions, summarizing the submitted systems and discussing the results.

The challenge started with the release of training data and auxiliary scripts to all registered participants. Participants were given the option to either create a validation set by further partitioning the training data for hyper-parameter tuning or to opt for cross-validation using the entire training data.
They began evaluating the submissions after the release of the Test set. Among all the various submissions, only five papers have been approved for the Workshop. These predictions were submitted and evaluated against the ground truth of the test subdata.
The evaluation was done in two steps, the first one is by automatic metrics, that calculated an average between BLEURT and BERTscore at three thresholds, as explain in Chapter 5. The second one is by an human evaluation, using the MTurk platform.

Below is a summary of the systems used by the top 5 participants in the challenge.

1. **FLUTE** [7]
   The system described in this paper is the baseline of the challenge.
   The authors have used a T5-3b model trained on more than 7000 examples.
   In order to solve the tasks, they used this natural language instruction:
   "Does the sentence "P" entail or contradict the sentence "H"? Answer between "Entails" or "Contradicts" and explain your decision in a sentence."

2. **DREAM-FLUTE** [16]
   This team, known as TeamCoolDoge, is the winner of the competition.
   The key idea of their approach is the utilization of *DREAM* [15]. *DREAM* is a system that, given a sentence, is able to generate an elaboration of the situation, adding some context information. The author used *DREAM* on the premise and the hypothesis helping improve the model's ability to decide if there is an entailment or not and the ability to judge so creating a pertinent explanation.
   This approach has proven to be highly effective for the situations involving figurative language, where the intended meaning may be difficult to discern. The *DREAM* system allows for diverse dimensions of scene elaboration, encompassing categories such as consequence, emotion, motivation, and social norm. The winning submission rely on consequence elaboration dimension.

They also fine-tuned a T5-3b model, adjusting some hyperparameters and using this format *<Premise> <Premise-elaboration-from-DREAM> <Hypothesis> <Hypothesis-elaboration-from-DREAM>* as input.

3. **Cross-Task Transfer Learning** [3]
   This is the team in second place, their focus was on the effective cross-task transfer learning for enhance performance on *FLUTE*.
   They employed two principal techniques: Sequential Fine Tuning and MultiTask Learning. In particular, their best final submission involves a model fine-tuned in sequence, starting with eSNLI [5], followed by IMPLI [37], and concluding with FLUTE [7].

4. **Divide-and-Conquer System** [28]
   They approached both the NLI task and the explanation generation task as two separate seq2seq tasks. The fine-tuning process involved treating these tasks independently within a simultaneous computation model.
   Additionally, they incorporated the attribute related to Figurative Language types throughout the data as a predictor, treating it as another seq2seq task. Consequently, they developed three component models through fine-tuning the pre-trained T5 model: the NLI predictor, Type predictor, and Generator. So, unlike other approaches where label and explanation are made jointly, this team used a T5-large model in a pipeline fashion.

5. **SBU - Figures It Out** [19]
   The base idea of this team was training the model focusing on generating explanations before the label. They simply used a T5-large model fine-tuned on the *FLUTE* dataset, where the input structure lacks task-specific keys.

3.4 DREAM method

DREAM is a novel technique introduced by [15] to enhance Situational Question-Answering (QA) by initially elaborating the situation.
The underlying hypothesis behind this method is as follows: cognitive science suggests that when individuals need to respond to questions about a particular situation, they first create a mental image of the situation and then answer. Therefore, the author posited that a LLM can also achieve greater accuracy when provided with additional contextual details.

DREAM is a model that performs this task; indeed, it is capable of processing the input phrase and creating a so called *scene elaboration* SE that will be added to the existing input.
This *Dream SE* provides details about the input situation S along four dimensions:

- *Motivation* of character(s) before S.
- *Emotion* of character(s) after S.
- *Social Norm* or general Rule of Thumb (ROT), it indicates if the action described in S is socially acceptable or not.
- Likely *Consequence* of action in S.

Consequently, the *Dream SE* is formed by a 4-tuple, where each element is represented as text, usually a single sentence, with an identifier preceding it.
In figure 3.7 there is an example of how *DREAM* works in the case of QA.

To construct the training dataset for learning Scene Elaborations, the authors used three existing commonsense resources:

1. *Story Commonsense* [29]
   This dataset has the role to provide the dimensions of *emotion* and *motivation.*
2. *Social Chemistry* [13]
   This one is the source for the *social norm* dimension.
3. *Moral Stories* [12]
   The last one provided the data to elaborate the *likely consequence* of a situation, separating a "moral" consequence from a "immoral" one.

At this point, to create the final *DREAM* model, they trained a T5-11B model with this novel Scene Elaboration Dataset.

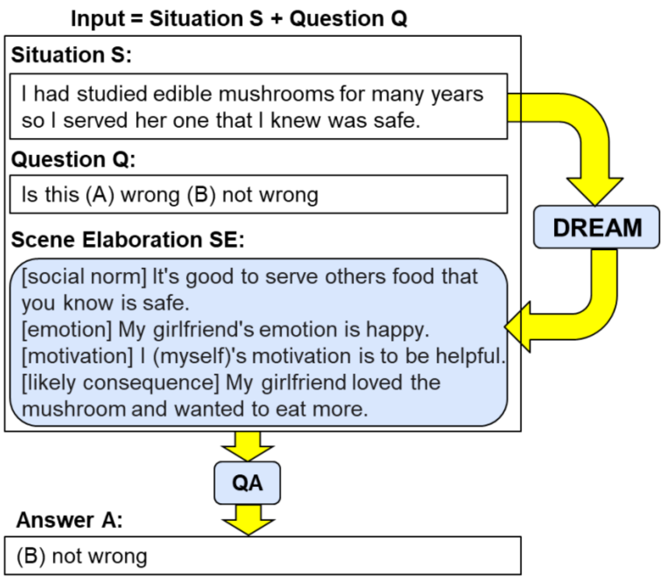

Input = Situation S + Question Q
Situation S:
I had studied edible mushrooms for many years so I served her one that I knew was safe.
Question Q:
Is this (A) wrong (B) not wrong
Scene Elaboration SE:
[social norm] It's good to serve others food that you know is safe.
[emotion] My girlfriend's emotion is happy.
[motivation] I (myself)'s motivation is to be helpful.
[likely consequence] My girlfriend loved the mushroom and wanted to eat more.
DREAM
QA
Answer A:
(B) not wrong

Figure 3.7: Example of QA with the use of DREAM system [15]

The research in [15], reveals that *DREAM* excels in generating scene elaborations more accurate, useful, and consistent when compared to a representative state-of-the-art zero-shot model.
The findings also indicate that incorporating these scene descriptions as extra context boosts the accuracy of answers in a downstream QA system. This enhancement surpasses the improvements obtained by solely fine-tuning the QA system with the training data from *DREAM*. These outcomes imply that incorporating focused elaborations about a situation has the potential to enhance a system's reasoning about it.
This approach also proves to be dataset-neutral, as enhanced performance is observed across different models, also on models with fewer parameters.

3.4.1 DREAM - FLUTE

The creators of the *DREAM* model have also participated at the FigLang 2022 challenge, as described in the previous section. Their approach, named *DREAM-FLUTE* [16], led them to victory.

Understanding figurative language presents a challenge as it is difficult to discern implicit information solely from its surface form. The authors' hypothesis is that effective performance in this task requires the reader to mentally elaborate on the described scene to discern a coherent meaning from the language used. The

DREAM-FLUTE model is a system that does this: before making an entailment/contradiction classification and generating an explanation, it creates a mental imagine of the situation using *DREAM* model [15].

In Figure 3.8 there is an example of how *DREAM-FLUTE* works. The system first uses *DREAM* to generate an elaboration of the situation for the premise and hypothesis separately. The additional context, in this case only the *consequence* category, is used both for entailment classification and for the explanation generation.

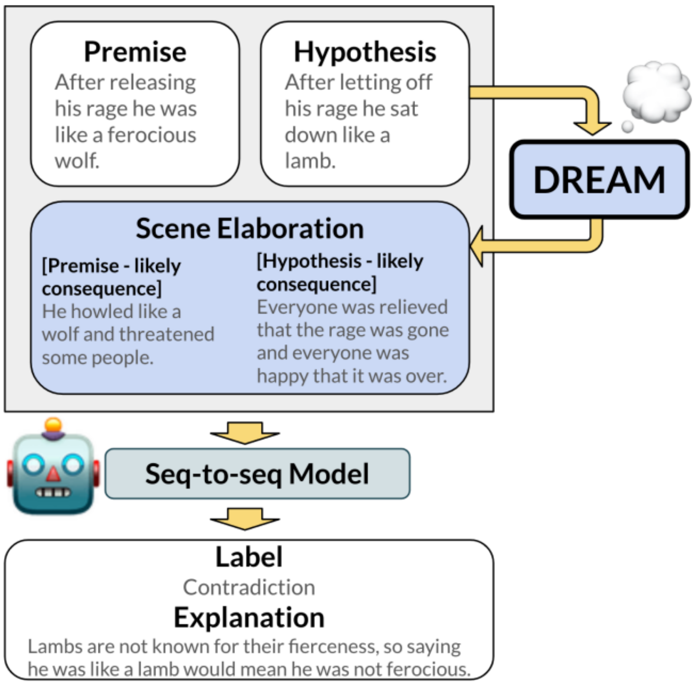

Premise
After releasing his rage he was like a ferocious wolf.
Hypothesis
After letting off his rage he sat down like a lamb.
DREAM
Scene Elaboration
[Premise - likely consequence]
He howled like a wolf and threatened some people.
[Hypothesis - likely consequence]
Everyone was relieved that the rage was gone and everyone was happy that it was over.
Seq-to-seq Model
Label
Contradiction
Explanation
Lambs are not known for their fierceness, so saying he was like a lamb would mean he was not ferocious.

Figure 3.8: Overview of DREAM-FLUTE [16].

In seeking the best solution to the FigLang 2022 challenge, the authors experimented with various approaches, presented below:

1. **Using original data**
   They simply trained a sequence-to-sequence model for the figurative language task.
   The input-output format is the following:
   Input <Premise> <Hypothesis>
   Output <Label> <Explanation>

2. **Jointly predicting the type of figurative language**

In addition, there is a model that jointly predict the type of the rhetorical figure.
The input-output format is the following:
Input <Premise> <Hypothesis>
Output <Figurative-Language-Type> <Label> <Explanation>

3. ***DREAM-FLUTE***
   They used the *DREAM* model to generate the *Dream SE*, adapting the *Dream's SEs* for the figurative language understanding. In this approach they provide the produced different dimensions as input context.
   The input-output format is the following:
   Input <Premise> <Premise-elaboration-from-DREAM>
   <Hypothesis> <Hypothesis-elaboration-from-DREAM>
   Output <Label> <Explanation>

4. **Two-step System**
   Unlike the precedent methods, in this case the tasks are approached separately, using a two-step "classify then explain" pipeline.

5. **Ensemble System**
   In this final scenario, the authors leveraged the ensemble of information learned by Systems 1 to 4. For both label selection and explanation generation, they employed all preceding systems as foundational components. This approach involves employing distinct elaborations to construct building blocks, forming a continuum with diverse levels of intuition and analysis. The model derives answers and rationalizes by considering different positions on a cognitive continuum.

4 Methodology

This chapter outlines the methodology adopted in the experiments, elucidating the approach taken in conducting the research, and its various phases.
We start by describing how the database was prepared for our objectives, then we proceed describing the types of experiments that were conducted and the modalities used, including the choice of hyperparameters. Subsequently, we briefly covers the evaluation part, which will be further explored in Chapter 5.
In the last part of the chapter, we provide some information regarding implementation details, enriching the reader's understanding on the practical aspects of the research.

4.1 Data Preparation

The dataset used is the one proposed by the authors of *FLUTE* [7]. This dataset was already divided into a test set consisting of 1500 elements and a training and validation set consisting of 7534 examples. From this latter set, we created two separated groups: the Train and the Validation sets.
Splitting data into training, validation, and test sets helps assess the performance a machine learning model by training it on one subset, tuning its hyperparameters on another, and ultimately evaluating its generalization on an independent dataset. This separation ensures robustness, prevents overfitting and enhances the model's ability to make accurate predictions on new, unseen data.

Among the Test set provided by the organizers, there are two examples for which they did not release either the correct label (entailment or contradiction) or the gold explanation. For this reason, we decided to exclude them from the experiments, resulting in a final Test set of 1498 pairs.

4.2 LLM Pipeline

First of all we have to say that the family of LLMs chosen for this study is LLaMA 2, specifically we decided to utilize version 7B of LLaMA 2-Chat, employed for our task thorough the prompting method.
An important detail to consider about this LLM, is that the maximum input token length is 4096.

Our approach involves inputting the premise-hypothesis pair and asking the model whether they contradict or entail each other, along with providing the reasoning behind its decision.
As mentioned, the model must perform a dual task: first, a classification task to determine whether there is entailment or contradiction, and second, a text generation task to produce an explanation for the earlier choice.
Our goal is to try various prompts on the validation set and then test the best combinations on the test set. We can determinate which combinations are the best because, each premise-hypothesis pair is accompanied by the correct label and by the given explanation. So, after our model generates the predicted label and its own explanation, we can compare them with the ground truth and made an evaluation as explained in Chapter 5.

All three types of prompting discussed in Chapter 2, have been utilized in our study: Zero-Shot prompting, Few-Shot prompting, and Chain-Of-Thought prompting.

The first issue we tackled was related to the choice of the prompt format. Indeed, LLMs are very sensitive to the prompt format, they completely rely on context and patterns in the input data to generate the output. The format of the prompt influences the model's understanding of the task and the context in which it should generate responses. A well-structured prompt helps guide the model to produce desired results with a coherent output, while an ambiguous or poorly formatted prompt may lead to unexpected or less accurate outputs.
Given the importance of the prompt format, numerous preliminary attempts were made to select the top 3 (one for each form of prompting: zero-shot, few-shot, and chain-of-thought) based on stability and effectiveness.

Subsequently, we tried to understand which values of the *temperature* parameter were more suitable.
We chose to experiment with various temperature values in a relatively generic and simple few-shot prompting scenario. In the end, we selected the two temperature values that yielded the best results.

At this point, we proceeded with the experiments.
In the *Zero-Shot* case, we prompted LLaMA with the chosen prompt and the two selected temperature values.
In Figure 4.1 there is the schema of this prompt type.

In the *Few-Shot* case, several variants were tested.
For this type of prompting, we primarily focused on two elements:

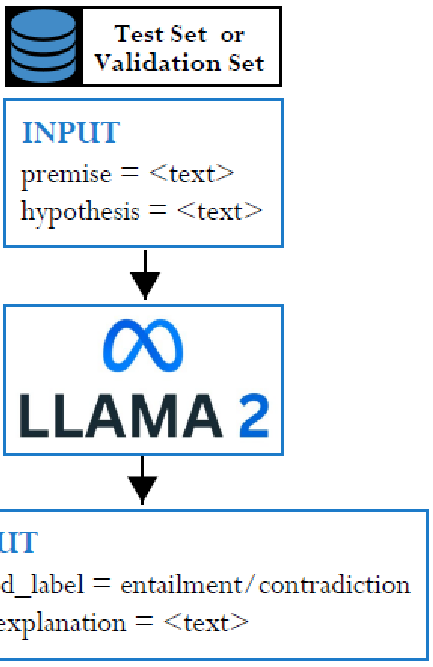

Test Set or Validation Set
INPUT
premise = <text>
hypothesis = <text>
LLAMA 2
OUTPUT
predicted_label = entailment/contradiction
model_explanation = <text>

Figure 4.1: Logical schema for Zero-Shot prompting.

- the number K of examples to include as input within the prompt
- the manner in which the K examples are selected from the Train set

Firstly, we tried the two simplest cases, passing as input only one example and then a pair of examples. In both cases examples were randomly selected from the Train set.
Instead, to test prompting using a number K of input examples greater than two, three different methodologies were employed:

1. The K input examples are randomly selected from the Train set
2. The K input examples are extracted from the Train set in such a way that rhetorical figures are balanced among them.
   For example, if we need to input 10 examples, we would choose 2 sarcasms, 2 idioms, 2 metaphors, 2 similes, and randomly select the remaining two.
3. The K input examples are extracted from the Train set in such a way that rhetorical figures are balanced according to the train set balance, assuming that the balance of this training set is the same as that of the test set. It results to have about half of the examples on sarcasm figure and the remaining half splitted in a balanced manner among the other rhetorical figures (simile,

metaphor and idiom).
For this reason, for this case, the number of examples inserted in the prompt is always a multiple of 6 (1/2 sarcasm, 1/6 idiom, 1/6 simile and 1/6 metaphor).

All these experiments conducted in the few-shot setting were carried out for both previously selected temperature values.
In Figure 4.2 there is the schema of this prompt type.

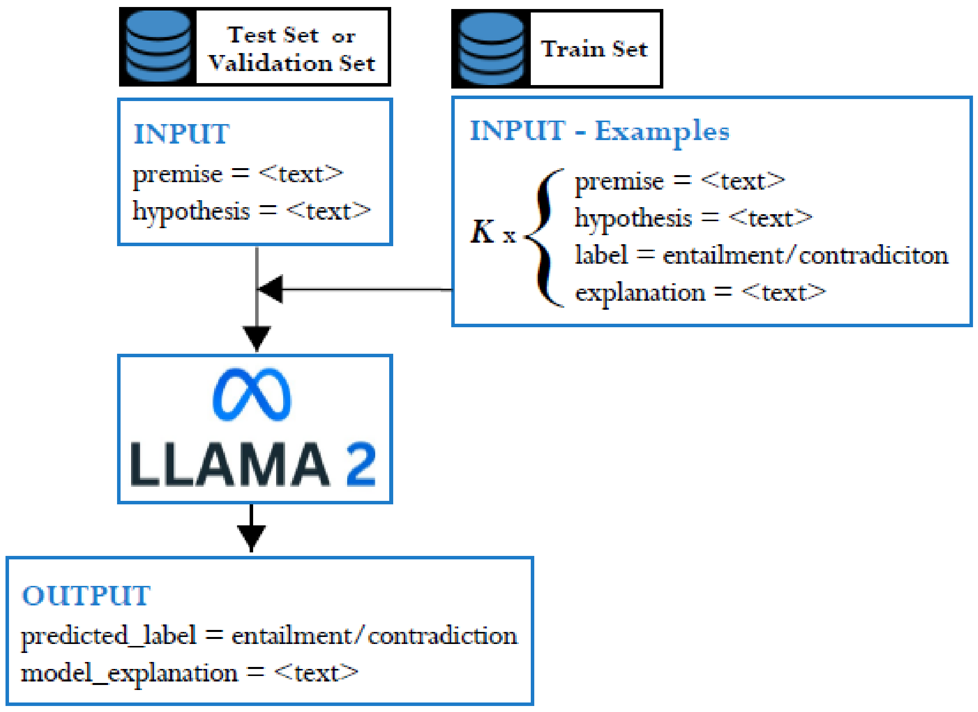


Figure 4.2: Logical schema for Few-Shot prompting.

For the last prompting style, the *Chain-Of-Thought* approach, the aim is to encourage the model to reason further by providing more context in the input, for each premise and hypothesis.
With additional information clarifying the meaning behind the main statements, it should be easier for the model to recognize whether the premise and hypothesis contradict each other and understand the reasoning behind it.
To achieve this, we utilized *DREAM* [15]; as explained in Chapter 3 it is a model that, given a sentence, is capable of creating a scene elaboration (*SE*) consisting of four factors: emotion, motivation, consequence, and social norm.
We chose to use this model because it has been demonstrated to perform well on the *FLUTE* dataset [16].

In essence, we applied *DREAM* to the entire *FLUTE* dataset, obtaining the extra context for both the premise and hypothesis, for each example in the train, test, and validation sets.
Practically, for each premise and hypothesis, we saved the four scene elaboration factors (emotion, motivation, consequence and social norm), Figure 4.3.

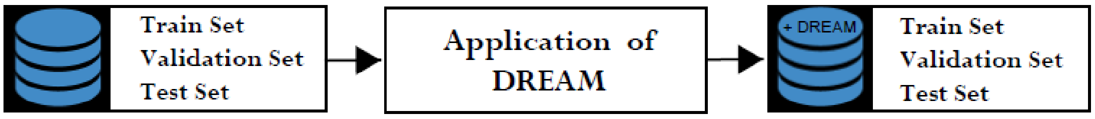


Figure 4.3: Logical schema for DREAM application.

We chose to analyze five cases, four involving passing a single characteristic of the generated scene at a time, and the last one consists in passing all of them together (*Dream SE*).
We performed both zero-shot and few-shot prompting. In the first case no examples are provided, but in the prompt there is only the pair to be evaluated along with its corresponding additional context. For the second case besides the pair to be evaluated, also the examples passed in input contain their respective *SE*.
In the few-shot cases, we opted to validate this method using only the best methodology among the three employed in the baseline scenario. Moreover, for a fair comparison, we test the strategies with and without the addition of dream context on the same set of examples.
In Figure 4.4 there is the schema of this prompt type.

4.3 Evaluation

In evaluating our work, we have opted for a two-step approach. The first step involves assessing the classification task, where the model predicts whether the input premise-hypothesis pair represents entailment or contradiction. The second step focuses on the text generation task, where we evaluate the explanations provided by the model to justify its previous prediction.
We have access to both the correct labels (entailment-contradiction) and the gold explanations for all the data in the Test set used, making evaluation much easier. For the first task, we will calculate accuracy, while for the second task, we will use BERTscore, ROUGE, and BLEURT metrics. The next chapter will provide a detailed overview of the evaluation process and the various scenarios considered.
In Chapter 6, where the results will be presented, all obtained values will be compared to those of baseline models, considering *FLUTE* model proposed by the FigLang 2022 challenge organizers and the *DREAM-System1* model proposed by the challenge winners.

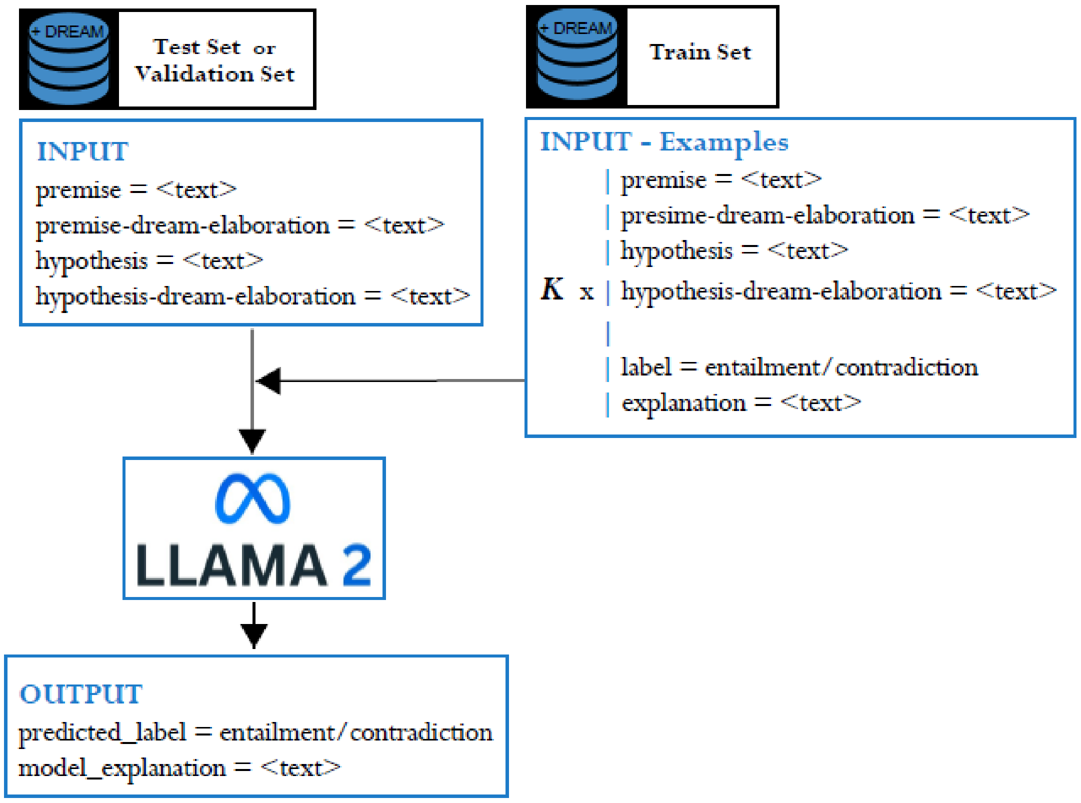

+ DREAM
Test Set or Validation Set
+ DREAM
Train Set
INPUT
premise = <text>
premise-dream-elaboration = <text>
hypothesis = <text>
hypothesis-dream-elaboration = <text>
INPUT - Examples
premise = <text>
presime-dream-elaboration = <text>
hypothesis = <text>
K x hypothesis-dream-elaboration = <text>
label = entailment/contradiction
explanation = <text>
LLAMA 2
OUTPUT
predicted_label = entailment/contradiction
model_explanation = <text>

Figure 4.4: Logical schema for the Chain-of-Thought prompting.

4.4 Implementation

The programming language employed for this thesis is Python, which is generally the most widely used at the moment. It stands out as the predominant programming language for addressing NLP tasks, since the majority of libraries and frameworks designed for deep learning are crafted here.
Below, we briefly present some of the most useful tools that users can use if they want to work in the field of NLP.

- *Natural Language Toolkit (NLTK)*:
  NLTK is one of the first NLP library in Python; it furnishes user-friendly interfaces to corpora and lexical resources like WordNet. It encompasses a suite of text-processing libraries catering to classification, stemming, tagging, parsing, and semantic reasoning.

- *spaCy*:
  spaCy stands out as a versatile open-source NLP library, supporting over 66 languages. Offering pre-trained word vectors and implementing popular models like BERT, spaCy facilitates the development of production-ready systems for tasks such as named entity recognition, part-of-speech tagging, sentence segmentation, dependency parsing, lemmatization, text classification, entity linking and morphological analysis.

- *Deep Learning Libraries*:
  The principal deep learning libraries are TensorFlow and PyTorch, they permit a huge simplification during the creation of any models, for example they have incorporated features like automatic differentiation. These libraries are widely utilized tools for the development of all the models including NLP ones.

- *Hugging Face*:
  Hugging Face provides open-source implementations and weights for more than 135 state-of-the-art models. The repository offers ease of customization and training for these models, enhancing accessibility and adaptability in NLP applications. The model used in our experiment, LLaMA-7b-chat-hf, is present in Hugging Face.

To execute the written code, two primary platforms were predominantly utilized: Google Colab and HPC@polito.

- **Google Colab**
  It is a cloud-based platform for writing and executing Python code in a collaborative environment. It provides free access to GPU and TPU resources, making it ideal for machine learning and data analysis tasks. Colab integrates with Google Drive, allowing seamless sharing and version control of Jupyter notebooks.
  This platform was used in the initial phase of data preparation and prompt selection, as well as to perform all evaluation measures for both the classification task and the text generation task.

- **HPC@polito** (www.hpc.polito.it)
  Politecnico di Torino HPC project is an Academic Computing center which provides computational resources and technical support for research activities in accademic and didattical purposes. The HPC project is officially managed by LABINF (Laboratorio Didattico di Informatica Avanzata) under supervision of DAUIN (Department of Control and Computer Engineering) which granted by Board of Directors.
  This platform was instead used to carry out all the various types of experiments and thus obtain the various labels and explanations produced by the model.

5 Evaluation Metrics

Assessing the quality of a system through human evaluation is typically considered the most reliable method. Nevertheless, conducting crowd-sourced experiments can be costly and time-consuming, making it impractical to integrate into a daily model development workflow. As a result, NLG researchers often rely on automatic evaluation metrics as a more affordable and efficient alternative, despite being only an approximation of quality.
Utilizing automatic evaluation metrics can aid in assessing your model's performance, monitoring your ML system during deployment, and adjusting your model to align with your business requirements. It's generally recommended to employ multiple evaluation metrics to assess your model, as a model might excel according to one metric while underperforming according to another. In the case of NLP field, the task of evaluating the semantic coherence and validity of a text in an automatic way is critical but is essential for ensuring the effectiveness of language models.

As previously described, in the case studied in this project, there are primarily two tasks. The first one of classification, where, given two sentences, the model must classify by choosing between two classes: Entailment or Contradiction. The second task is a text generation one where we ask the model to generate a text explaining the reasons behind the choice made in the previous classification.
Hence, to evaluate the model's performance, two different evaluation metrics are required: the first related to the classification task and the second to the text generation task. In both cases, the simplest means of evaluating label or generated text is to measure how well they agree with those marked or provided by humans.

This chapter aims to introduce some of the basic metrics used in these two fields and we will see the ones employed in our experiments. Subsequently, we will introduce recent eXplainable Artificial Intelligence (XAI) metrics, and they will be analyzed to understand if it is feasible to apply them to our case study and, if so, how to do so.

Model Evaluation Procedures

Before we dive into metrics, we briefly explain model evaluation procedures.
As general rule we want to create a model that well generalize with the out-of-sample data. If you train and test your model using the same dataset, it will

likely overfit to the training data and fail to generalize effectively. Therefore, it's advisable not to train your model on the entire dataset when evaluating its performance. A typical strategy is the train/test split, where you would use a certain percentage of the data for training (70/80%) and the remaining data for testing.
Another important point concerns assessing the model's performance, which is crucial for determining the optimal combination of hyperparameters. However, using the test set for evaluation may lead to hyperparameters that are only optimal for that specific case and may not generalize effectively. Therefore, instead of dividing the data into just two parts, train and test sets, it's preferable to split it into three parts: train, validation, and test sets. The validation set serves the dual purpose of identifying overfitting during training and ensuring that hyperparameters generalize well. Additionally, shuffling the data before splitting helps ensure that each subset accurately represents the dataset.
In our case, the split used for our experiments is 60% of the data for training, 20% of the data for validation, and 20% of the data for testing.

5.1 Metrics for Classification Task

Classification is about predicting the class labels given input data. There are two types of classification, the first one is the binary classification, like our case, where there are only two possible output classes. The second case is about the multiclass classification, where more than two possible outcomes can be present.
In this paragraph we will take into consideration the case of supervised learning, so we compared the predicted labels with the correct ones. There are many ways for measuring classification performance and we will present some of the most popular: confusion matrix, AUC-ROC and accuracy. The last metric is the one we used in our experiments.

5.1.1 Confusion Matrix

A confusion matrix or error matrix is a fundamental tool for evaluating the performance of a classification model. It provides a comprehensive breakdown of the model's predictions by comparing them to the actual ground truth. It is a table that displays the number of incorrect and correct predictions produced by the model, compared with the real classifications and it reveals the types of errors occurring.
This matrix illustrates how well a classification model performs on test data where the true values are already known. It is a nxn matrix, where n represents the number of classes. Displayed in Figure 5.1, this matrix is created after making predictions on the test data. Columns indicate the frequency of actual classifications, while rows indicate the frequency of predicted classifications made by the model.

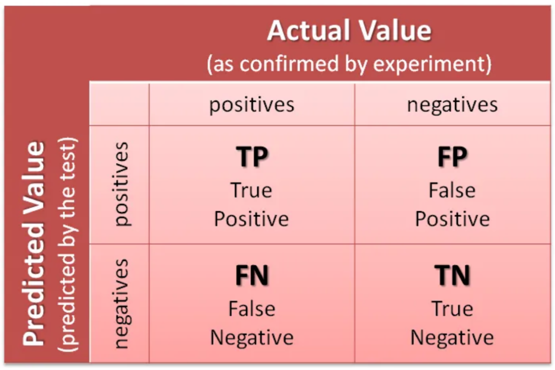


Figure 5.1: Confusion Matrix

There are four potential outcomes that may arise during the process of making classification predictions:

- True Positive (TP): The count of positive outcomes that are correctly predicted as positive.
- False Positive (FP): The count of negative outcomes that are incorrectly predicted as positive. This is the so-called *Type 1 Errors*.
- True Negative (TN): The count of negative outcomes that are correctly predicted as negative.
- False Negative (FN): The count of positive outcomes that are incorrectly predicted as negative. This is the so-called *Type 2 Errors*.

To be clearer to the reader, we emphasize that "Positives and negatives" relates to the prediction, whereas "true and false" relates to the accuracy of the prediction.

Examining the matrix, we observe that the diagonal elements signify the instances where the predicted label matches the true label, whereas the off-diagonal elements represent misclassifications made by the classifier. A higher value along the diagonal of the confusion matrix indicates more accurate predictions, signifying better performance.
From the Confusion Matrix, we can obtain four metrics for classification:

1. *Accuracy*
   It can also be computed based on positives and negatives in binary classification. However, it doesn't provide us with much insight into the distribution of false positives and false negatives.
   $$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
2. *Precision or Positive Predictive Value (PPV)*
   It is the proportion of True Positives to all positives predicted by the model. It is particularly beneficial for datasets that are skewed or unbalanced. When the model predicts more false positives, precision decreases.
   $$precision = \frac{TP}{TP + FP}$$
3. *Recall or Sensitivity or True Positive Rate (TPR)*
   It represents the proportion of correctly identified positive instances compared to all positive instances within the dataset. It gauges the model's capability to identify positive samples, and if the model predicts more false negatives, the recall score decreases.
   $$recall = \frac{TP}{TP + FN}$$

4. *F-score or F-measure*

   (a) *F1-score*
   It is a unified measure that incorporates both precision and recall. It falls within the range of 0 to 1, with higher values indicating better performance by our model. Since the F1 score is a balanced combination of precision and recall, the classifier achieves a high F1 score only when both precision and recall are high.

   $$F1\_score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

   Be careful because this metric only benefits classifiers with comparable precision and recall.

   (b) *F-score with β factor*
   The F-score can be seen as a more generalized version of the F1-score. In the overall F-score, a parameter β determines the relative emphasis placed on precision versus recall in the evaluation process:

   - $\beta < 1$: The evaluation is precision oriented.
   - $\beta > 1$: The evaluation is recall oriented.
   - $\beta = 1$: This is the standard F1-score where precision and recall are balanced.

   Here the formulation of the overall F-score with β factor.

   $$F_{\beta}\_score = \frac{(1 + \beta^2) \cdot precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

About precision and recall we can make two general considerations:

- Precision considers the classification of both positive and negative samples, so it is dependent on both samples. Instead the recall only focuses on the positive samples so it is dependent only on the positive ones and independent of the negative ones.

- Precision focuses on correctly identifying samples as Positive, without emphasizing the correct classification of all positive samples. Recall, on the other hand, prioritizes accurately classifying all positive samples but does not worry if a negative sample is mistakenly classified as positive.

5.1.2 Receiver Operating Characteristic Curve (ROC) & Area Under the Curve (AUC)

A **Receiver Operating Characteristic curve** (ROC curve) illustrates how well a classification model performs by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various decision thresholds. These thresholds determine whether a prediction is classified as "true" or "false." Adjusting the threshold allows for control over the balance between TPR and FPR. Typically, raising the threshold boosts precision but reduces recall.
Initially, let us analyze the TPR and FPR:

- *True Positive Rate (TPR/sensitivity/recall)*: it represents the percentage of positive instances correctly identified as positive among all positive instances.

$$TPR = \frac{TP}{TP + FN}$$

- *False Positive Rate (FPR)*: it represents the percentage of negative data points incorrectly identified as positive among all negative data points.

$$FPR = \frac{FP}{TN + FP}$$

Both values fall within the range of 0 to 1 and are calculated at different threshold levels. An ideal classifier would exhibit a high true positive rate and a low false positive rate. The ROC curve in Figure 5.2 shown a more precise model.

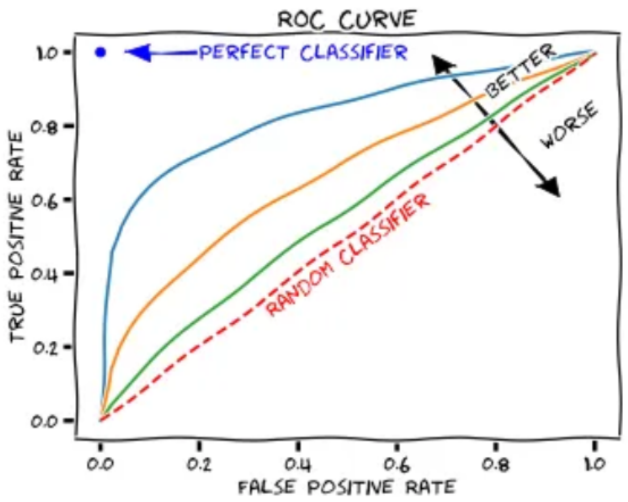


Figure 5.2: ROC curve

In summary:

- any model that exhibits a ROC curve surpassing the random guessing classifier line can be deemed superior.
- any model that exhibits a ROC curve below the random guessing classifier line can be unequivocally dismissed.

This kind of graph illustrates the TPR against the FPR across various classification thresholds. However, this method is inefficient as it requires evaluating the model at multiple thresholds. A more efficient approach is to use a sorting-based algorithm such as AUC, which can provide us with this information effectively.

The **Area Under the ROC Curve**, also known as the Area Under the Curve (AUC), serves as a numerical summary of a graph, particularly employed in binary classification tasks. Mathematically, it represents a function generating points along a curve. AUC quantifies the probability of the classifier ranking a randomly chosen positive instance higher than a randomly chosen negative one. Its utility lies in facilitating model comparison as it condenses information across the entire ROC curve. AUC values range between 0 and 1, with higher values indicating superior model performance.

5.1.3 Accuracy

The simplest classification metric for model evaluation is *Accuracy* and it is the one employed in this project to evaluate our first task. It is calculated as the ratio of correctly predicted instances to the total number of instances:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The benefits and limitations of this metric are listed below.

Benefits:

- Simplicity and Intuitiveness: Accuracy is straightforward to understand, making it a popular choice for reporting model performance. It provides a clear indication of the model's ability to make correct predictions across all classes.
- Easy Comparison: Accuracy allows for easy comparison between different models or iterations. A higher accuracy score generally indicates better overall performance.

– Interpretability: Accuracy provides a global view of a model's effectiveness, making it interpretable for a broad audience, including non-technical stakeholders.

Limitations:

– Unbalanced Problems: One significant limitation of accuracy arises when dealing with imbalanced datasets, where one class significantly outnumbers the others. In such cases, a model may achieve high accuracy by predominantly predicting the majority class, while performing poorly on minority classes. Hence, in imbalanced scenarios, accuracy can be misleading, as a model might appear successful when, in reality, it struggles with critical minority classes. This can have severe consequences in applications where the minority classes are of particular interest.

– Alternative Metrics Needed: To address the limitations of accuracy in unbalanced problems, alternative metrics like precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC) are often employed. These metrics provide a more nuanced evaluation of a model's performance, especially when considering class-specific predictions.

In conclusion, while accuracy remains a valuable metric for evaluating classification tasks, its limitations become apparent in the presence of imbalanced datasets. A comprehensive evaluation strategy should incorporate a combination of metrics to provide a more nuanced and informative assessment of a model's performance, particularly when faced with challenging class imbalances. Understanding the strengths and weaknesses of accuracy is crucial for researchers and practitioners aiming to derive meaningful insights from classification models.
Regarding our case study we have a test set unbalanced for what regard the figurative language but quite balanced for what regard the label. Indeed, we have a test set of $\sim$1500 samples composed by 750 sarcasm and 250 of simile, idiom and metaphor each but consider all the figure type together we have 663 sample classify as Entailment (44.3%) and 835 as Contradiction (55.7%).

5.2 Metrics for Text Generation Task

This second section is related to the text generation task, in this case the aim of a good evaluation metric is to verify if the generated text is coherent for its intended purpose, for example, by comparing it to a reference text. A generation evaluation metric is a function $f(x, \hat{x}) \in \mathbb{R}$ where x is the reference sentence tokenized to k tokens $(x_1, ..., x_k)$ and $\hat{x}$ is the candidate sentence tokenized to m tokens $(\hat{x}_1, ..., \hat{x}_m)$.
The main objective is to assess semantic equivalence, but predominantly techniques rely on surface-form similarity only. For instance, BLEU [25], merely calculates the overlap of n-grams between the candidate and the reference. This approach offers a simple and general measure but, it overlooks the importance of preserving meaning through diverse lexical and compositional choices.

In this specific setting, our objective is to evaluate the explanation provided by our model in comparison to the reference one. The purpose of the explanation is to show how the premise and hypothesis are connected, revealing if there is an entailment or a contradiction. The challenge is that this explanation does not rely on factual information but rather on the subtle meaning of these figures of speech: metaphors, idioms, sarcasm, and similes.
Unlike conventional evaluations that focus on grammar correctness or syntactic comparisons, our task involves a semantic assessment of two sentences and it is acknowledged that two sentences can exhibit significant structural differences yet convey the same meaning.

In this paragraph, we will provide a briefly overview of the evaluation metrics utilized for the text generation task, and then we will introduce the main metrics used in our experiments, BLEURT, BERTscore and ROUGE.
In the FigLang 2022 challenge, to judge the quality of the generated explanations they compute the average between BLEURT and BERTscore. This average, denoted as the "explanation score", falls within the range of 0 to 100. To give importance to this score the promoters of the challenge asked to report the label accuracy at three thresholds of this explanation score. Accuracy@0 is equivalent to simply computing label accuracy. Accuracy@50 is based on the first threshold, it counts as correct only the correctly predicted labels that achieve an explanation score greater than 50. Accuracy@60 is similar to Accuracy@50 but the threshold is based on an explanation score over 60.
To more accurately assess the quality of the generated explanations by our models, we have computed precision, recall, and F1-score for both ROUGE and BERTscore, comparing them with the ones of the baseline models presented at the FigLang 2022 challenge.

5.2.1 Overview on Base Metrics

The basic metrics can be grouped into three general categories: n-gram matching, edit distance and embedding matching.

- The *n-gram matching* approaches are the most commonly used metrics for text generation. Given a reference x and candidate $\hat{x}$, they count the number of n-grams that occur. As n increases, the metric becomes more adept at capturing word order, yet it also becomes increasingly rigid and limited to mirroring the precise structure of the reference. Formally, let S_x^n and $S_{\hat{x}}^n$ be the lists of token n-grams ($n \in \mathbb{Z}_+$) in the reference x and candidate $\hat{x}$ sentences. The number of matched n-gram is $\sum_{w \in S_{\hat{x}}^n} \mathbb{I}[w \in S_x^n]$, where $\mathbb{I}$ is an indicator function. The exact match precision (*Exact-P_n*) and recall (*Exact-R_n*) scores are:
$$Exact_{P_n} = \frac{\sum_{w \in S_{\hat{x}}^n} \mathbb{I}[w \in S_x^n]}{|S_{\hat{x}}^n|}, Exact_{R_n} = \frac{\sum_{w \in S_{\hat{x}}^n} \mathbb{I}[w \in S_x^n]}{|S_x^n|}$$
Numerous widely used metrics are based on either one or both of these precise matching scores. The most widely used metric, especially for translation, is BLEU (Bilingual Evaluation Understudy); it is calculated across various values of n, with the scores being geometrically averaged. A modified version called SentBLEU is computed on a sentence-by-sentence level. Other metrics of the same type are ROUGE and METEOR.

- Multiple approaches utilize *word edit distance* or *word error rate*, which measure similarity by determining the number of edit operations needed to transform the candidate into the reference. The most famous metrics of this type are Jaro, Leveisten and Jaro-Winkler but they are suitable for comparing very short and similar text portions, so they are not appropriate for our case.

- The *embedding-based metrics* utilize word embedding and shallow semantic parsing to calculate similarity in both lexical and structural aspects.

Since these metrics only detect changes in word choice, they fail to adequately acknowledge shifts in meaning or sentence structure within a given text. As a result, they often demonstrate weak correlation with human evaluations. To tackle this issue, NLG researchers have introduced trained elements into these metrics to enhance their alignment with human judgments.

Fully trained metrics like RUSE [33] or BEER [36] are trained comprehensively from end to end. They typically rely on either manually crafted features or embeddings that are learned. These metrics often provide high levels of expressiveness: given a training set containing human ratings data, they can fully exploit it to closely match the distribution of ratings. Additionally, these learned metrics can be adjusted to evaluate specific aspects of tasks, such as fluency, faithfulness, grammar, or style. However, the drawback is that all these trained methods necessitate expensive human judgments as supervision for each dataset. There's also a risk of poor adaptation to new domains and data, even within familiar language and task domains.

Hybrid evaluation metrics like YiSi [22] and BERTscore merge pre-trained components, like contextual embeddings, with manually crafted rules, such as token alignment criteria. In contrast to fully learned metrics, they demonstrate resilience, yielding superior outcomes even with limited training data, without the necessity of assuming identical distributions between training and testing datasets.

5.2.2 BERT Score

BERTscore is an automatic evaluation metric for text generation based on pre-trained BERT contextual embeddings [45].
Given a candidate sentence $\hat{x} = (\hat{x}_1, ..., \hat{x}_m)$ and a reference sentence $x = (x_1, ..., x_k)$, BERTscore compute a similarity score for each token in the candidate sentence $\hat{x}$, with each token in the reference sentence x. However, instead of exact matches, it use contextual embeddings to represent the tokens, and compute a weighted matching using cosine similarity and inverse document frequency scores.
To represent the tokens in the input sentences, x and $\hat{x}$, is hence used the contextual embeddings, presented in Chapter 2. In contrast to prior word embeddings, contextual embeddings have the capability to produce varying vector representations for a given word across different sentences. This variability is influenced by the surrounding words that constitute the context of the word in question.

The reference sentence $x = (x_1, ..., x_k)$ and the candidate one $\hat{x} = (\hat{x}_1, ..., \hat{x}_m)$ are tokenized by BERT. It generates two sequences of vectors: $(x_1, ..., x_k)$ and $(\hat{x}_1, ..., \hat{x}_l)$. Subsequently, given a reference token x_i and a candidate token $\hat{x}_j$, the cosine similarity is calculated:

$$\frac{x_i^T \hat{x}_j}{\|x_i\| \|\hat{x}_j\|}$$

In order to reduce the calculation to the inner product $x_i^T \hat{x}_j$, pre-normalized vectors are used.
The entire score correlates every item in x with a corresponding item in $\hat{x}$ to determine recall, and matches each item in $\hat{x}$ with an item in x to calculate precision. The authors employ a greedy matching approach to maximize the similarity score between matches, with each token being paired with its closest counterpart in the other sentence. They then combine precision and recall to compute an F1 measure. Given a reference x and candidate $\hat{x}$, the BERT recall, precision, and F1 scores

are:

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j, P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_i \in \hat{x}} \max_{x_i \in x} x_i^T \hat{x}_j$$

$$F1_{BERT} = 2\frac{R_{BERT} \cdot P_{BERT}}{R_{BERT} + P_{BERT}}$$

BERTscore easily incorporate importance weighting in these measures. For doing this the authors chose to use the inverse document frequency (idf) and not the full tf-idf measure because processing single sentences, the term frequency (tf) is likely 1. Given M reference sentences ${x^{(i)}}_{i=1}^{M}$ and $\mathbb{I}$ as indicator function, the inverse document frequency (idf) score of a token w is

$$idf(w) = -log\frac{1}{M}\sum_{i=1}^{M} \mathbb{I}[w \in x^{(i)}]$$

and, for example, the BERT recall score with idf weighting would be

$$R_{BERT} = \frac{\sum_{x_i \in x} idf(x_i) \max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j}{\sum_{x_i \in x} idf(x_i)}$$

The contextual embedding method used in BERTscore brings a series of improvement with respect to the prior metrics:

– First, n-gram models fail to capture distant dependencies but BERTscore, unlike BLEU, is not limited to a maximum n-gram length and can grasp dependencies of possibly unlimited extent.

– Second, in previous models, semantically accurate phrases are often penalized because they deviate from the surface structure of the reference. However, the context embedding method enables the capture of a token's specific usage in a sentence, potentially encompassing sequence information.

– Third, in contrast to the previously discussed learned method, the model that underpins BERTscore is not tailored for any particular evaluation task, leading to better generalization. Additionally, it eschews external tools for generating linguistic structures, rendering this approach relatively straightforward and adaptable to new languages.

5.2.3 BLEURT

The second metric we introduce in this paragraph is BLEURT [32], a learned evaluation metric based on BERT. The fundamental concept at the core is the possibility to merge expressiveness and robustness through initially training a fully developed metric on extensive synthetic data, followed by a fine-tuning on human ratings.

Define $x = (x_1, ..., x_r)$ as the reference sentence of length r and $\hat{x} = (\hat{x}_1, ..., \hat{x}_p)$ as the predicted sentence of length p, where each x_i or $\hat{x}_i$ is a token. Let $(x_i, \hat{x}_i, y_i)_{n=1}^{N}$ be a training dataset of size N where $y_i \in \mathbb{R}$ is the human rating that indicates how good $\hat{x}_i$ is with respect to x_i. Given the training data, the goal is to learn a function $f : (x, \hat{x}) \rightarrow y$ that predicts the human rating.
Due to the limited quantity of rating data accessible, it's logical to utilize unsupervised representations for this purpose. In this approach the model employed is BERT, which is an unsupervised method capable of acquiring contextualized representations of text sequences.
Given x and $\hat{x}$, BERT is a Transformer [41] that returns a sequence of contextualized vectors:

$$v_{[CLS]}, v_{x_1}, ..., v_{x_r}, v_{\hat{x}_1}, ..., v_{\hat{x}_p} = BERT(x, \hat{x})$$

The place-holder $v_{[CLS]}$ denotes the encoding for the unique [*CLS*] token. To forecast the rating, a linear layer is appended onto the [*CLS*] vector:

$$\hat{y} = f(x, \hat{x}) = W\hat{v}_{[CLS]} + b$$

where W is the weight matrix and b the bias vector.
The linear layer mentioned above, along with the BERT parameters, are trained and fine-tuned using supervised data.

However, optimizing BERT for specific tasks, through a fine-tuning, necessitates a significant quantity of data that are both independent and identically distributed. This poses a challenge as it's not ideal for a metric aiming to generalize across different tasks and account for model drift.
The key feature of the BLEURT method involves a novel pre-training scheme employed to prepare BERT before refining it with rating data. The authors have created millions of synthetic reference-candidate pairs $(z, \hat{z})$ to aid the model's ability to generalize.
Any method of pre-training necessitates both a dataset and a collection of pre-training tasks. Ideally, the setup ought to mirror the final NLG evaluation task, ensuring that the distribution of sentence pairs aligns and that the pre-training signals are in line with human ratings. However, since accessing future NLG

models is unfeasible, the approach is optimized for versatility, adhering to three specific requirements:

- The collection of example sentences needs to be extensive and varied to ensure that BLEURT can effectively handle a broad spectrum of NLG domains and tasks.
- The sentence pairs should encompass a diverse range of differences in vocabulary, sentence structure, and meaning. The objective is to predict all potential variations that a NLG system might generate, such as replacing phrases, offering paraphrases, introducing noise, or omitting words.
- The pre-training goals must adequately encompass those phenomena to enable BLEURT to recognize them effectively.

In this scenario, the pre-training method employs random alterations of sentences from Wikipedia, combined with various forms of lexical and semantic supervision cues. To generate these synthetic sentence pairs the authors used three techniques: mask-filling with BERT, backtranslation and model random dropping.

The subsequent phase involves enhancing every pair of sentences $(z, \hat{z})$ by incorporating a collection of pre-training cues τ_k, where τ_k represents the target vector associated with pre-training task k. Effective pre-training cues ought to encompass a broad spectrum of lexical and semantic variances. Additionally, they should be inexpensive to acquire to enable scalability to extensive sets of synthetic data. The author have created three signals $\tau BLEU$, $\tau ROUGE$, and $\tau BERTscore$ with sentence BLEU [25], ROUGE [21], and BERTscore [45] respectively. For every pre-training task, the model employs either a regression or classification loss, which are then combined using a weighted sum to form task-level losses.

Here an example for better understand how to create this regression/classification loss. For each task, let τ_k be the target vector. If the task τ_k is a a regression one, then the loss used will be the l_2 loss i.e. $l_k = \frac{\|\tau_k - \hat{\tau}_k\|_2^2}{|\tau_k|}$ where $|\tau_k|$ is the dimension of τ_k. $\hat{\tau}_k$ is instead computed by using a linear layer (task-specific) on top of the [CLS] embedding: $\hat{\tau}_k = W_{\tau_k} \hat{v}_{[CLS]} + b_{\tau_k}$.
If the task τ_k is a classification one, then the loss used will be the multiclass cross-entropy loss [32]. Subsequently, we will predict a logit for each class c using a separate linear layer: $\hat{\tau}_{kc} = W_{\tau_{kc}} \hat{v}_{[CLS]} + b_{\tau_{kc}}$.
The aggregate pre-training loss function is defined as follows:

$$l_{pre-training} = \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \gamma_k l_k(\tau_k^m, \hat{\tau}_k^m)$$

where τ_k^m, M and γ_k are respectively the target vector for example m, the number of synthetic examples, and the hyperparameter weights obtained with grid search.

To conclude, it can be stated that BLEURT, an English reference-based text generation metric, undergoes end-to-end training, achieving superior accuracy in modeling human assessment. Additionally, pre-training enhances the metrics' resilience to domain and quality fluctuations.

5.2.4 ROUGE

ROUGE, Recall-Oriented Understudy for Gisting Evaluation, is an intrinsic evaluation metric focus on syntax. It was specifically designed for the assessment of summaries by [21].
The metric incorporates methods to automatically assess and compare a generated text with a collection of reference texts, typically human-generated. The foundational paper outlines four distinct ROUGE measures: *ROUGE-N*, *ROUGE-L*, *ROUGE-W*, and *ROUGE-S*.
In our experiments, we employed two types of ROUGE metrics: *ROUGE-N*, for the cases where N=1 and N=2, and *ROUGE-L*, sentence-level case.

ROUGE-N: N-gram Co-Occurrence Statistics

In a formal context, *ROUGE-N* represents the recall of n-grams in comparison between a candidate summary and a collection of reference summaries.
The computation of *ROUGE-N* recall is outlined as follows:

$$ROUGE_N = \frac{\sum\limits_{S \in \{ReferenceText\}} \sum\limits_{gram_n \in S} Count_{match}(gram_n)}{\sum\limits_{S \in \{ReferenceText\}} \sum\limits_{gram_n \in S} Count_{match}(gram_n)}$$

where the variables n, $gram_n$, and $Count_{match}(gram_n)$ represent the length of the n-gram, the maximum occurrence of n-grams in a candidate text, and a collection of reference texts, respectively.
Note that also the corresponding measures for standard precision and F1-score are available. It is recommended to consider Rouge-N Recall when assessing fixed-size summaries. Alternatively, Rouge-N F1-score is more suitable in other scenarios.

ROUGE-N originates as a recall-related metric; in fact, the denominator in the previous formula represents the total count of n-grams present on the reference summary side. A closely associated metric, BLEU [25], employed in the automated evaluation of machine translation, is based on precision. BLEU assesses the alignment between a candidate translation and a set of reference translations by calculating the percentage of overlapping n-grams in the candidate translation with the references.

ROUGE-L: Longest Common Subsequence

A sequence $Z = [z_1, z_2, ..., z_n]$ is a subsequence of another one $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of X such that for all $j = 1, 2, ..., k$, we have $x_{ij} = z$.
Given two sequences X and Y, the longest common subsequence (LCS) of X and Y is a shared subsequence with maximum length. In order to apply LCS in the context of summarization evaluation, we treat a summary sentence as a sequence of words. The underlying idea is that a higher LCS length between two summary sentences indicates greater similarity between the two summaries.

To quantify the similarity between two texts, X of length m and Y of length n, the authors proposed employing an LCS-based F-score measure. Of course, the key components include LCS-based recall and precision.
Assuming X serves as a reference sentence, Y as a candidate sentence and LCS(X,Y) denotes the length of a longest common subsequence of X and Y, the formulation is as follow:

$$R_{lcs} = \frac{LCS(X,Y)}{m}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

Notice that *ROUGE-L* equals 1 when X equals Y, and it is zero when $LCS(X,Y)$ is zero, indicating that there is no commonality between X and Y.

Utilizing LCS offers a notable benefit as it does not necessitate consecutive matches but rather in-sequence matches that mirror the word order at the sentence level as n-grams. Another advantage is its automatic inclusion of the longest in-sequence common n-grams, eliminating the need for a predefined n-gram length. In the unigram case, recall and precision count all co-occurring words regardless their orders. In contrast, *ROUGE-L* considers only co-occurrences that occur in sequence. To better understand this distinction, consider the following example:

Reference = police killed the gunman
S1 = police kill the gunman
S2 = the gunman kill police

Despite S1 and S2 have very different meanings, the two candidate sentences would have the same *ROUGE-2* score, since they both have one bigram (“the gunman”). Instead, in the case of *ROUGE-L*, S1 has a score of 3/4 = 0.75 and S2 has a score of $2/4 = 0.5$ (with $\beta = 1$). Therefore, according to *ROUGE-L*, S1 results better than S2.

One of LCS disadvantage is that it only counts the primarly in-sequence words; consequently, other available LCS variations and briefer sequences do not contribute to the ultimate score. To illustrate, consider the following candidate sentence:

S3 = the gunman police killed

Using the previous reference sentence, LCS counts either "the gunman" or "police killed", but not both. In conclusion *ROUGE-L* will give the same score to S3 and S2 while *ROUGE-2* would prefer S3 than S2.

The previously described approach pertains to sentence-level LCS and is the one employed in our experiments. There is also a corresponding method for summary-level analysis.

ROUGE-W: Weighted Longest Common Subsequence

This approach represents a progression from the previous one, favoring strings with consecutive matches. It can be efficiently calculated using dynamic programming. Here is an example:

Reference = [A B C D E F G]
S1 = [A B C D H I K]
S2 = [A H B K C I D]

In this case S1 has the same *ROUGE-L* score as S2 but *ROUGE-W* would prefer S1 than S2.

ROUGE-S: Skip-Bigram Co-Occurrence Statistics

The evaluation involves comparing the overlap of skip-bigrams, which are pairs of two consecutive non-stop words with a defined gap. This metric is employed to account for long-distance dependencies, allowing for gaps in matches similar to LCS but considering all in-sequence pairs rather than only the longest subsequences. Considering again the police example:

Reference = police killed the gunman
S1 = police kill the gunman
S2 = the gunman kill police

S3 = the gunman police killed

we have

Rouge-N: S3 > S1 = S2
Rouge-L: S1 > S2 = S3
Rouge-S: S1 > S3 > S2

where

Rouge-S(S1) = 3/6 ("police the", "police gunman", "the gunman")
Rouge-S(S2) = 1/6 ("the gunman")
Rouge-S(S3) = 2/6 ("the gunman", "police killed")

ROUGE-SU is an extensions of *ROUGE-S* encompassing both unigrams and bigrams. It incorporates the co-occurrence statistics based on unigrams in addition to the skip-bigrams as *ROUGE-S*.

5.3 Explainable Artificial Intelligence

EXplainable Artificial Intelligence (XAI) is a critical frontier in the development and deployment of intelligent systems. As machine learning models and complex algorithms become increasingly prevalent in various domains, the need for understanding and interpreting their decisions has become paramount. XAI addresses this challenge by striving to make AI systems transparent, interpretable, and ultimately trustworthy.

The traditional black-box nature of many advanced AI models, such as deep neural networks, often leaves users, stakeholders, and even developers puzzled about the underlying logic behind the model's predictions. This lack of interpretability poses significant challenges, particularly in applications where decisions impact human lives, such as healthcare, finance, and criminal justice.
XAI aims to bridge this gap by providing insights into how AI models arrive at specific outcomes. It encompasses a diverse set of techniques and methodologies designed to make the decision-making process of AI systems more understandable and explainable to human users. The overarching goal is to ensure that AI applications not only deliver accurate predictions but also offer comprehensible justifications for those predictions.
The importance of XAI extends beyond merely satisfying human curiosity. In many industries, regulatory requirements demand transparency in automated decision making processes. Moreover, gaining user trust is crucial for the widespread adoption of AI technologies. XAI plays a pivotal role in achieving these objectives by demystifying the decision-making process and enabling users to comprehend, validate, and, if necessary, challenge the outputs of AI systems.

Regarding our field, in recent years, significant advancements in large neural NLP models have transformed the landscape of NLP applications, showcasing remarkable performance across various tasks. However, their increasing complexity implies the increase of unclear elements. This black-box nature poses a challenge for practitioners seeking explanations regarding the rationale behind specific predictions and the key features influencing them. The emergence of explainable AI techniques, has played a crucial role in addressing this issue by shedding light on the internal mechanisms of transformers and fostering trust in their decision-making process. Various XAI methodologies have been proposed in the literature, and we are going to present the principal explanation and evaluation metrics for NLP tasks.

5.3.1 Post-Hoc Features Importance Methods

There are two divergent interpretability methods, the intrinsic ones and the post-hoc ones. With the intrinsic ones, the model architecture itself helps to provide the explanation, while the post-hoc methods offer explanations after a model has been trained and are agnostic to the specific model used [24].
This paragraph focuses on the methods that can be applied retroactively, so the post-hoc ones. These techniques frequently face criticism for offering false or inaccurate explanations, prompting doubts about the feasibility of expecting models originally not intended for explanation to provide them. While this concern is legitimate, developing inherent methods is often highly contingent on the specific task, making it a difficult process that is rarely done in the industry. Unlike inherently interpretable models, post-hoc methods are often much more adaptable, indeed they can be utilized with any machine learning model, irrespective of its complexity or the algorithm it is based on. Their influence could potentially amplify significantly through the provision of precise explanations.

Below the widely adopted family of post-hoc feature attribution methods are presented, they are the so called **XAI explanation metrics** [9].
Provided with a model, a specific target class, and a prediction, these techniques assess the individual influence of each token on that prediction. The methods discussed include Gradient [35] (also referred to as Saliency) and Integrated Gradient [23], SHAP [38] as an exemplar of Shapley value-based approaches, and LIME [30] representing local surrogate methods.

1. *Gradient*:
   The gradient method focuses on understanding the impact of input features on the model's output by calculating the gradients of the model's predictions with respect to the input features. It helps identify which features have the most significant influence on the model's output by analyzing the magnitude of the gradients.

2. *Integrated Gradient*:
   Integrated Gradient is an extension of the gradient method. It calculates the average gradient along the path between a baseline input and the actual input, providing a more comprehensive understanding of feature importance. This method addresses some of the limitations of the basic gradient approach and offers a more nuanced view of how input features contribute to the model's predictions.

3. *SHAP*:
   SHapley Additive exPlanations (SHAP) is based on cooperative game theory and Shapley values. It assigns a value to each feature by considering

its contribution to all possible combinations of features. SHAP values provide a unified measure of feature importance, ensuring a fair distribution of credit among the features. It helps users understand the impact of individual features on model predictions.

4. *LIME*:
   Local Interpretable Model-agnostic Explanations (LIME) generates locally faithful interpretations by approximating the complex model with a simpler, interpretable model. It achieves this by perturbing the input data and observing changes in predictions. LIME focuses on explaining individual predictions rather than the overall model behavior hence it is useful especially in cases where complex models lack transparency, and it is convenient to create simplified, understandable models for local interpretations.

5.3.2 Evaluation Metrics

In this paragraph we are going to explore some of the **XAI evaluation metrics**, in particular we are going to describe three state-of-the-art metrics to measure plausibility and three for faithfulness.
There are indeed two kinds of evaluation measures, the faithfulness ones which gauge how accurately a sentence reflects the genuine reasoning process of the model, and the plausibility ones which assess how persuasive the interpretation is to humans. Certainly, it is possible to satisfy one of these criteria without fulfilling the other.

In the realm of Human-Computer Interaction (HCI), the aim of providing explanations is to enhance trust between users and the system. However, improved system performance in this context doesn't necessarily signify faithfulness. Instead, it reflects correlation between the plausibility of the explanations and the model's performance since any level of correlation between credibility and performance will result in increased user performance, regardless of any concept of faithfulness.
A faithful interpretation refers to one that accurately captures the reasoning process behind the model's prediction but, there is no universally agreed-upon and formal definition of faithfulness [17].
Moreover, another important thing to take into consideration is that attention weights do not provide meaningful “explanations" for predictions [18].

Jain and Wallance [18] perform multiple experiments on a variety of NLP tasks that aim to illustrate that conventional attention modules do not offer substantial explanations and should not be regarded as such. For example, they highlight instances where learned attention weights often do not align with gradient-based measures of feature significance, and others where different attention distributions can lead to equivalent predictions. In their discourse, Jain and Wallace put forward two fundamental points to support their assertion:

1. The attention weights produced do not consistently or strongly align with measures of feature importance, particularly those derived from gradients and leave-one-out techniques.

2. Different attention weights, leading to varied heatmaps or explanations, may not always result in different forecasts. Frequently, one can generate adversarial attention distributions that produce equivalent predictions as those obtained from the initial attention weights, even though they focus on entirely different input features. Additionally, shuffling attention weights typically brings about only slight alterations in the final output.

Before presenting these evaluation measures it's important to take into consideration that all the metrics presented in this paragraph are specific to the classification task. The starting input is hence a sentence that needs to be classified, for example as positive, neutral or negative. To this input, the predicted label is added, and various XAI explanation metrics, like the ones presented in the precedent paragraph, are applied. They assign a "score" to each token, indicating how important that token is for the classification. Once this explanation step is completed, the evaluation step follows with specific metrics.

Faithfulness

The following measures of faithfulness will be presented: correlations with 'leave-one-out' scores [18], comprehensiveness and sufficiency [11].
The input is the one produced by the explanation methods and is composed by three elements: the sentence that has to be classified, the label predicted and the scores for each token of the in-question sentence.

1. *Kendall's Tau correlation with Leave-One-Out token removal*
   Kendall's Tau is a correlation measure used in NLP to assess the similarity of rankings generated by different models. Leave-One-Out token removal (LOO) is an evaluation method where individual tokens are systematically removed one at a time from the input data, observing the impact on the model's performance. Combining these two concepts, Correlation with Leave-One-Out scores ($\uparrow$) involves assessing the correlation between the rankings produced by a model or system using Kendall's Tau metric, with the additional consideration of systematically removing one token at a time from the input data to observe the model's sensitivity to individual tokens [2]. Under the linearity assumption, LOO scores represent a simple measure to gauge the importance

of individual features [17]. In simpler terms, it's a way to evaluate how well a model's ranking aligns with human rankings, while also understanding the influence of individual tokens on the model's performance. This can provide insights into the model's robustness and its sensitivity to specific words or elements in the input. The author of [18] present in their work all the experiments with the measured correlations between attention and differences in model output induced by leaving features out.

2. *Comprehensiveness*
   Comprehensiveness ($\uparrow$) assesses if the explanation adequately encompasses the tokens utilized by the model in making a prediction. The authors gauge this by eliminating the tokens emphasized by the explainer and observing the resultant change in probability. Let f_j be the prediction probability of a model f for the target class j and let r_j be a discrete rationale or denoting the set of tokens supporting the prediction f_j. Given the input sentence x, comprehensiveness is defined as $f(x)_j - f(x \backslash r_j)_j$ where $x \backslash r_j$ is the sentence x were tokens in r_j are removed. A high comprehensiveness value implies the relevance of tokens in r_j to the prediction, whereas a low score suggests their insignificance. A negative value indicates increased model confidence in its prediction upon removal of the rationales.

3. *Sufficiency*
   Sufficiency ($\downarrow$) indicates whether the tokens in the explanation provide enough information for the model to make its prediction. With reference to the precedent notation, it is evaluated as $f(x)_j - f(r_j)_j$. A low score indicates that the tokens within r_j are indeed the primary contributors to the prediction.

4. *AOPC Comprehensiveness and Sufficiency*
   The measures just presented have assumed discrete rationales r_i but the author of *Eraser* [11] proposed an alternative approach. They also wanted to assess how accurately models assign continuous importance scores to tokens. Initially, they eliminate tokens with a negative contribution, the ones that pull the prediction away from the chosen label. Following this, they identify discrete rationales r_i by selecting the highest k_d values, where k_d represents a threshold specific to dataset d. We set k_d to the average length of rationales provided by humans for that specific dataset d.
   Intuitively, this indicates the extent to which the model's prediction alters when we eliminate a quantity of tokens equivalent to the average human usage for this dataset, based on the importance scores assigned by the model. After discretizing the soft scores into rationales as outlined, we calculate the faithfulness scores using the equation detailed by [11] for both comprehensiveness and sufficiency. In contrast to measures that require per-token measurements,

as score correlations with LOO scores, this approach is conceptually simple and it does not require much computational effort to assess. Nonetheless, the need to convert continuous scores into discrete ones compels us to select a specific threshold k. Therefore, they examine how these metrics vary based on k. To ensure consistency in comparing this metric across various datasets, they establish bins indicating the quantity of tokens to remove. For each instance i they define an aggregate comprehensiveness measure and one for sufficiency analogously. Given k=5 bins, they group tokens into the top 1%, 5%, 10%, 20% and 50% of tokens, in relation to the corresponding importance score. These metrics are presented as Area Over the Perturbation Curve (AOPC) comprehensiveness and sufficiency and they score a specific token ordering under a model.

Plausability

The following measures of plausibility [11] will be presented: Intersection-Over-Union (IOU) at the token level, token-level F1 scores and Area Under the Precision-Recall curve (AUPRC). The input is the one produced by the explanation methods but this time is composed by four elements: the sentence that has to be classified, the label predicted, the scores for each token of the in-question sentence and the human rationales as ground true.

1. *Intersection-Over-Union (IOU) at the token level*
   Based on both human and predicted rationale, IOU ($\uparrow$) operates at a token level, wherein it measures the overlap between two spans by dividing the size of their intersection by the size of their union. A prediction is considered a match if its overlap with any of the ground truth rationales exceeds a certain threshold, typically set at 0.5 [2].

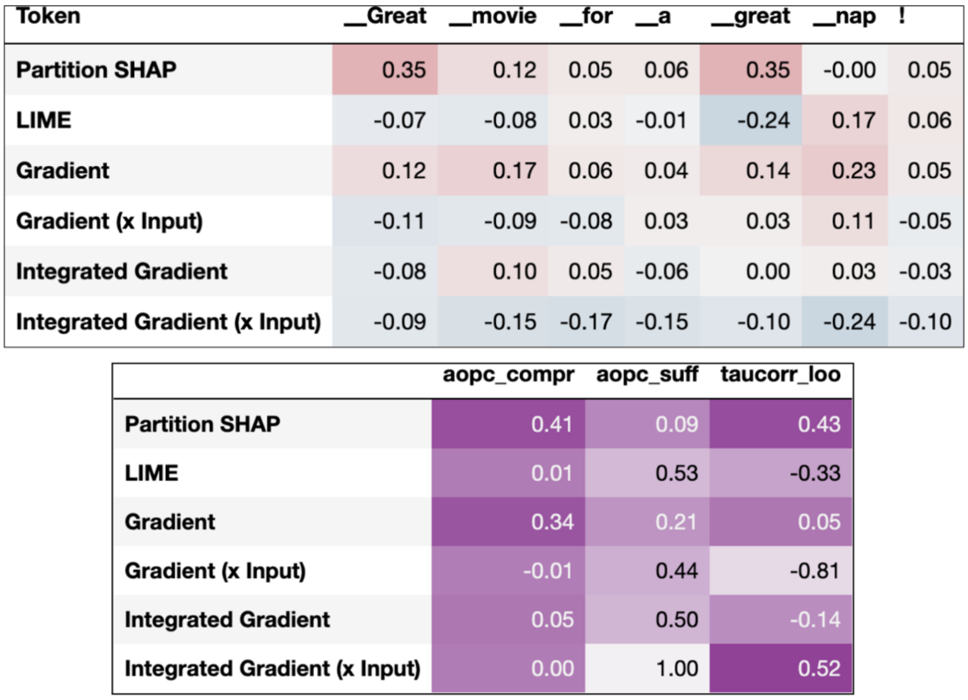
2. *Token-level F1 scores*
   Utilizing both human-generated and predicted rationales, partial matches are employed to assess token-level precision and recall. These measurements are then utilized to calculate token-level F1 scores ($\uparrow$).

3. *Area Under the Precision-Recall curve (AUPRC)*
   AUPRC ($\uparrow$), is calculated for explanations with continuous scores according to the method outlined in [11]. This computation involves adjusting a threshold across token importance scores, with the human rationale serving as the ground truth.

5.3.3 Potential Application to Our Case Study

Despite the significance of ensuring accessibility of XAI methods to NLP experts and practitioners through practical tools, there remains a notable deficiency in accessibility specifically for transformer models. XAI for transformers is indeed mainly scattered and hard to operationalize in practice.

Among the various XAI tools available, we particularly consider FERRET [2], a Python library designed to simplify the utilization and comparison of XAI methods on transformer-based classifiers.
FERRET enables users to visualize and compare explanations generated by transformer-based models using state-of-the-art XAI methods on any free-text or existing XAI datasets. Furthermore, users can also assess adhoc XAI metrics to determine the most reliable and credible explanations. Look at Figure 5.3 to better understand how FERRET works.



Token	▁Great	▁movie	▁for	▁a	▁great	▁nap	!
Partition SHAP	0.35	0.12	0.05	0.06	0.35	-0.00	0.05
LIME	-0.07	-0.08	0.03	-0.01	-0.24	0.17	0.06
Gradient	0.12	0.17	0.06	0.04	0.14	0.23	0.05
Gradient (x Input)	-0.11	-0.09	-0.08	0.03	0.03	0.11	-0.05
Integrated Gradient	-0.08	0.10	0.05	-0.06	0.00	0.03	-0.03
Integrated Gradient (x Input)	-0.09	-0.15	-0.17	-0.15	-0.10	-0.24	-0.10

	aopc_compr	aopc_suff	taucorr_loo
Partition SHAP	0.41	0.09	0.43
LIME	0.01	0.53	-0.33
Gradient	0.34	0.21	0.05
Gradient (x Input)	-0.01	0.44	-0.81
Integrated Gradient	0.05	0.50	-0.14
Integrated Gradient (x Input)	0.00	1.00	0.52

Figure 5.3: On the top: Token attributions to the prediciton, darker red (blue) show higher (lower) contribution. On the bottom: Faithfulness metrics, darker colors show better performance [2].

While the underlying idea of the evaluation metrics presented in FERRET is theoretically intriguing, these metrics cannot be applied to our case. The main issue lies in the fact that FERRET is a tool for Sequence Classification, whereas in our case, we addressed the problem through a Text Generation task.

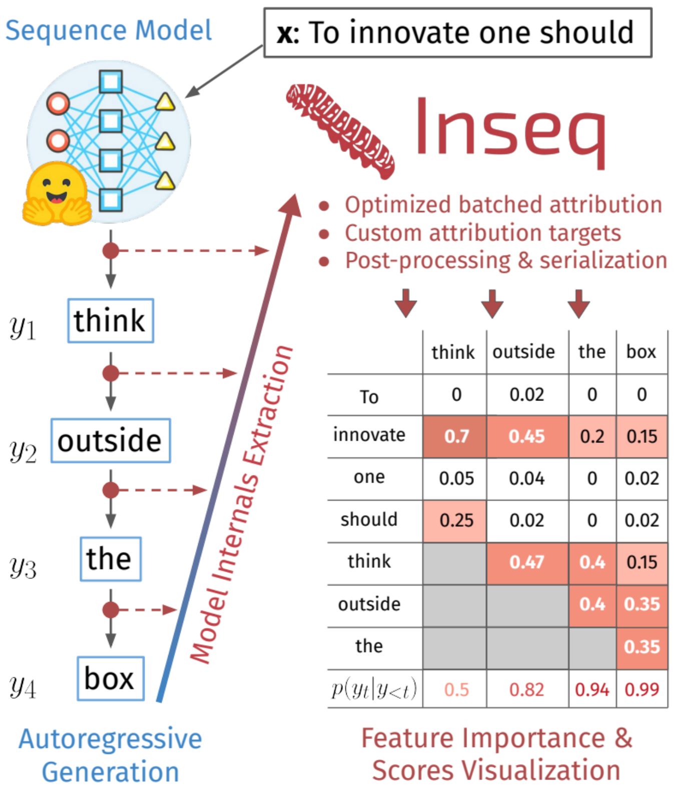
In FERRET, the principal treated case is the classical classification scenario in the contest of sentiment analysis, where a single sentence is classified as positive, negative, or neutral. The problem we want to overcome differs as we have a different input and different tasks. Our input is an association of two phrases, the premise and the hypothesis, and the general task is composed by two steps. The first one involves classification, deciding whether the two sentences imply an entailment or a contradiction, and the second one requires explaining why the model made this choice, constituting a text generation task.
The focal point of our problem's evaluation is the assessment of the explanation produced by the model to justify its choice and it cannot be evaluated using the FERRET method, as, in this case, the two problems are entirely different in terms of task and input.

Thinking about alternative solutions, one might consider analyzing only the first part of our problem with FERRET, i.e., the classification phase where the model decides if the two input sentences contradict or entail each other. However, several difficulties arise in this case as well. First of all our case, despite being a Sequence Classification task, it was solved as a text generation task since we issued a single prompt to LLaMA, instructing it to perform both tasks consecutively. And moreover, even if the two tasks were solved separately, we should adapt the Ferret tool to our type of classification task.
A theoretical solution could be fine-tuned LLaMA for a sequence classification task, so training it specifically for our task, or employing an existing classification model suitable for our task, such as roberta-large-mnli [48].
In this case the input should be suitable for FERRET tool and we could subsequently utilize FERRET to conduct explanations and evaluations. With the explanation tools it will assign a score to each token indicating its importance for the final prediction. At that point, the XAI metrics could be employed for the evaluation of faithfulness or plausibility.

Regarding the second task, pure text generation, one would need to explore the existence of explanability metrics for this case. There is another tool called INSEQ [31], which supposedly provides explanability techniques for Sequence Generation tasks.
INSEQ is a Python library designed to make interpretability analyses of generative language models more accessible to a wider audience. In regards to usability, INSEQ significantly simplifies access to local and global explanations with a built-in support. It includes a command line interface (CLI) for easy navigation, optimized

batching for dataset-wide attribution, and multiple techniques for visualizing, saving, and reloading attribution results and sequences.
In essence, INSEQ offers a straightforward platform for implementing feature attribution methods in sequence generation assignments. These techniques are divided into three categories: gradient-based, internals-based, and perturbation-based, each with distinct approaches to quantifying importance.
Between the supported methods there are the ones presented before: Gradient, Integrated gradient, SHAP gradient for the first category, Attention weight for the second category and LIME for the last one.
This tool offers distinct features within encoder-decoder architectures. For instance, users have the ability to specify whether to incorporate or omit the generated prefix in the associated inputs by adjusting a particular parameter. Additionally, INSEQ comprises multiple Aggregator classes designed to facilitate attribution aggregation across different dimensions.
During the process of attribution, INSEQ initially utilizes Transformers to generate target tokens and then proceeds to attribute them incrementally. At each attribution stage, INSEQ has the capability to leverage internal information of models to derive relevant scores such as probabilities or entropy, which are valuable for assessing model uncertainty and other purposes. INSEQ facilitates the calculation of these scores by providing access to various pre-defined step functions, while also allowing users to develop and register their own customized ones. The scores associated with each step are computed alongside the attribution process, presented as distinct sequences in the output, and visualized alongside importance scores. Refer to Figure 5.4 for a clearer understanding of the sequential attribution mechanism.

In conclusion we can theoretically apply this tool to both our tasks. Using INSEQ, it seems possible to associate two texts as input and output. For the first task the input would be the two sentences (premise and hypothesis) and the output would be the predicted label. Indeed, for the second task the input would be the two sentences (premise and hypothesis) and the label (entailment or contradiction), while the output would be the model's explanation.


Sequence Model
x: To innovate one should
Inseq
Optimized batched attribution
Custom attribution targets
Post-processing & serialization
y_1 think
y_2 outside
y_3 the
y_4 box
Model Internals Extraction
Autoregressive Generation
Feature Importance & Scores Visualization

	think	outside	the	box
To	0	0.02	0	0
innovate	0.7	0.45	0.2	0.15
one	0.05	0.04	0	0.02
should	0.25	0.02	0	0.02
think		0.47	0.4	0.15
outside			0.4	0.35
the				0.35
$p(y_t \mid y_{<t})$	0.5	0.82	0.94	0.99

Figure 5.4: INSEQ with a Transformers causal language model: feature importance and next-step probability extraction and visualization [31].

6 Experimental Results

In this chapter, we will present the outcomes of our experiments. After detailing the experimental setup, we will present the results from three different perspectives. In the first two sections, we will examine the performance of the systems employed according to different tasks. The first section will focus on classification task, while the second will address text generation task. In the final section, we will conduct a qualitative analysis of the results, comparing the explanations generated by our model with those ground-truth and those produced by reference models.

6.1 Experimental Setup

The first thing done was to defined the Train and Validation set from the *FLUTE* dataset. As said in Chapter 4 the Test set was already given from the promoters of the challenge and consists in 1500 examples. From the other 7534 examples we created a Validation set of 1500 examples and a Train set with the remaining ones. The validation set was constructed taking into account the configuration of the initial set. It was observed that the dataset is not perfectly balanced in terms of various rhetorical figures but is markedly imbalanced, as half examples are solely of sarcasm, while the remaining rhetorical figures (metaphor, idiom, and simile), are fairly balanced in terms of the remaining half of the presented examples.

Moving on to the choice of the prompting format, after several attempts, the format chosen was the one proved to be the most stable and effective.

For the *Zero-Shot* case results:

```
Find if the 'premise' entails or contradicts the 'hypothesis'.
The output must strictly follow the following format:
Label: 'Entails' or 'Contradicts', Explanation: 'text'
premise: {input:premise}
hypothesis: {input:hypothesis}
```

Instead for the *Few-Shot* case with K examples:

```
Find if the 'premise' entails or contradicts the 'hypothesis'.
Here you can find some examples of answers:
```

```
{input: #K of examples}
premise: {input:premise}
hypothesis: {input:hypothesis}
```

where the *examples* in input are written with this format:

```
premise: "text"
hypothesis: "text"
Answer: 'Entails' or 'Contradicts'
Explanation: "text"
```

For the *Chain-of-Thought* prompting case, we utilized the prompt format proposed by *DREAM-FLUTE* in [16].
For the case with only one dimension, emotion, motivation, social norm or consequence, we have used:

```
Find if the 'premise' entails or contradicts the 'hypothesis'.
Here you can find some examples of answers:
{input: #K of examples}
premise: {input:premise} emotion:{input:premise-emotion}
hypothesis: {input:hypothesis} emotion:{input:hypothesis-emotion}
```

where the *examples* in input are written with this format:

```
premise: "text" emotion: "text"
hypothesis: "text" emotion: "text"
Answer: 'Entails' or 'Contradicts'
Explanation: "text"
```

For the case where we pass all the *Dream SE* so all the 4 dimensions together, we have used:

```
Find if the 'premise' entails or contradicts the 'hypothesis'.
Here you can find some examples of answers:
{input: #K of examples}
premise: {input:premise} [emotion]{input:premise-emotion}
  [motivation]{input:premise-motivation} [social norm]
  {input:premise-rot} [consequence]{input:premise-consequence}
hypothesis: {input:hypothesis} [emotion]{input:hypothesis-emotion}
  [motivation]{input:hypothesis-motivation} [social norm]
  {input:hypothesis-rot} [consequence]
  {input:hypothesis-consequence}
```

where the *examples* in input are written with this format:

```
premise: "text" [emotion] "text"  [motivation] "text"
    [social norm] "text" [consequence] "text"
hypothesis: "text" [emotion] "text"  [motivation] "text"
    [social norm] "text" [consequence] "text"
Answer: 'Entails' or 'Contradicts'
Explanation: "text"
```

In order to understand which values of the *temperature* parameter were more suitable, we chose a simple case, a prompt using few-shot learning with two examples, one entailment, and one contradiction. The tested values were 0, 0.3, 0.6, and 0.9, and the best-performing ones were the first two, 0 and 0.3.

At this point, we proceeded with the experiments.
We started with the Zero-Shot case, where we prompted LLaMA with the previously described prompt and the two selected temperature values.
We continued with the Few-Shot case, where several variants were tested. We began with the two simplest cases, passing as input a single example (*K=1*), and then a pair of examples (*K=2*). Of course these examples were randomly selected from the training set. We put a constraint in the second case imposing the presence of an entailment and a contradiction.

As explain in Chapter 4, to test prompting using a number K of input examples greater than two, we employed three different methodologies.

1. We have tested a number K = *5, 10, 20, 30, 40* of input examples choosing randomly from the train set.
2. We have tested a number K = *5, 10, 20, 30, 40* of input examples choosing them from the training set in such a way that rhetorical figures were balanced among them.
   For example, if we need to input 10 examples, we would choose 2 sarcasms, 2 idioms, 2 metaphors, 2 similes, and randomly select the remaining two.
3. We have tested a number K = *6, 12, 18, 24, 30, 36, 42* of input examples choosing them from the training set in such a way that rhetorical figures were balanced according to the train set balance, i.e. having half of the select examples chosen from sarcasm figure and the remaining half chosen in a balanced manner among the other rhetorical figures.

Each of these experiments was conducted both in the case of zero temperature and in the case of positive temperature (0.3).

6.2 Results

We will present the results in two steps. In the first part, we will present the resulting metrics for the classification task, while in the second part, we will focus on those specific to the text generation task. In both cases, the metrics will be compared with two baseline models:

- *FLUTE T5-3b model*: the baseline model presented by the promoters of the FigLang2022 challenge [7] and described in Chapter 3, Section 3.
- *DREAM-FLUTE System1*: the first model proposed by [16] and described in Chapter 3, Section 4.

6.2.1 Classification Task

For this task, the most suitable metric is accuracy, as described in Chapter 5. We have chosen to present the results for the cases specified in the FigLang 2022 challenge using three metrics: Accuracy@0 (*Acc@0*), which involves computing label accuracy; Accuracy@50 (*Acc@50*), considering only the correctly predicted labels with an explanation score greater than 50; and Accuracy@60 (*Acc@60*), counting only the correct predictions with an explanation score exceeding 60.
It is important to note that in this context, the 'explanation score' is once again the one proposed by the challenge, representing the average between BLEURT and BERTscore.

The baseline results, for both the *FLUTE* model and the *DREAM-System1* model, are presented in Table 6.1. For each of our models, we provide results for the entire Test Set and sometimes also break down the results for each rhetorical figure. From these experiments those that surpass the baseline of the *FLUTE* model will be identified with bold numbers.

Let's begin by analyzing the Zero-shot scenario. Table 6.2 illustrates this case with zero and positive temperatures, and the case where the input is enriched with the entire *Dream SE* (i.e. all the 4 dimensions).
As observed, all obtained results are notably low. The challenge in this scenario lies specifically in the prompt; without any examples, the model varies its output format inconsistently. Given the lack of coherence with the required format, it is not feasible to automatically save the requested label and explanation.

Shifting our focus to the results obtained in the Few-shot prompting scenario, let's start with the two simplest cases: K=1 and K=2. In Table 6.3, we maintain the previous structure, including the case with zero and positive temperatures,

		FLUTE	**DREAM**
Test Set	Acc@0	0,8168	0,9453
	Acc@50	0,7476	0,8879
	Acc@60	0,4833	0,6061
Idiom	Acc@0	0,792	0,924
	Acc@50	0,772	0,916
	Acc@60	0,668	0,808
Metaphor	Acc@0	0,733	0,9194
	Acc@50	0,556	0,7621
	Acc@60	0,237	0,4355
Sarcasm	Acc@0	0,916	0,9773
	Acc@50	0,862	0,9413
	Acc@60	0,562	0,6387
Simile	Acc@0	0,628	0,896
	Acc@50	0,572	0,824
	Acc@60	0,304	0,476

Table 6.1: Classification baseline metrics from *FLUTE* and *DREAM-System1* models.

Zero-Shot			
Temperature	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*
Acc@0	0,5334	0,5414	0,4446
Acc@50	0,2570	0,2390	0,0070
Acc@60	0,0314	0,0294	0

Table 6.2: Classification metrics for zero-shot prompting. In the table, consider the cases with temperature null and positive, and the case with the addition of *Dream SE* in the input.

and the case where the input is enriched with *Dream SE*.
We can notice that the scenario with only one example in input, is similar to the zero-shot case as the model's output format is unstable. Furthermore, it is highly influenced by the type of example provided as input. It has been observed that if an 'entailment' example is passed, the model tends to classify all labels as entailment, and a similar behavior occurs when the input example is a contradiction. For this reason, in all few-shot trials with K$\geq$2, we have imposed a constraint to have at least one example for each category (entailment-contradiction).
In the case where we pass a pair (K=2) as input, we observe a slight improvement in performance compared to previous cases. Additionally, contrary to what was expected, enriching the output with *Dream SE* does not improve performance.

Few-Shot						
	K=1			***K=2***		
Temperature	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*
Acc@0	0,5574	0,5574	0,5574	0,7223	0,7296	0,6676
Acc@50	0,3945	0,3992	0,3385	0,5481	0,5487	0,3945
Acc@60	0,1288	0,1302	0,0494	0,2443	0,2543	0,1108

Table 6.3: Classification metrics for few-shot prompting with K=1 and K=2. In the table, consider the cases with temperature null and positive, and the case with the addition of *Dream SE* in the input.

For the experiments with few-shot prompting involving more than 2 examples, we tested the three methodologies presented in Chapter 4 on the validation set. It was found that the third approach, which is the one balanced with respect to the figures in the training set, yielded the best results. Due to the substantial number of experiments on the validation set, these results will be not reported.
We decided to evaluate on the Test Set only the cases related to the third strategy, the so-called 'balanced' one, and the top-performing cases from the others, specifically the experiments with K=10 and K=20 utilizing the first strategy, the random one.

The results for the 'random' strategy are presented in Table 6.4. We can observe that this is the first case in which our model, with K=10 and null temperature, outperforms the corresponding baseline result. Upon studying this experiment for each rhetorical figure, it is apparent that the model surpasses the baseline for almost all rhetorical figures except for sarcasm. However, since sarcasm constitutes half of the test set, the resulting score is either slightly better (Acc@0) or slightly lower (Acc@50, Acc@60) than the baseline.

In Table 6.5, we present the results on the Test set for the experiments conducted using the 'balanced' method with a number of input examples K = 6, 12, 18, 24, 30, 36, 42. In the table, we report the results with null and positive temperature. We can observe that there is a standard trend for both temperatures, where as the input examples increase, so do the performances until a certain saturation point is reached. After this point, any additional examples only add "noise," causing the performances to decrease again.
There are three cases where we outperform the baseline, these cases are better described in Table 6.6, where the results for each rhetorical figure are analyzed. As a general comment, we can say that the rhetorical figure that our model identifies best is metaphor, while the one it classifies worst is idiom.

Few Shot - *random method*						
	K=10			***K=20***		
Temperature	**T=0**	**T=0.3**	***T=0*** *+ emotion*	**T=0**	**T=0.3**	***T=0*** *+ emotion*
Acc@0	**0,8211**	0,8138	0,7490	0,7463	0,7430	0,7116
Acc@50	0,7109	0,7043	0,5981	0,6642	0,6615	0,6215
Acc@60	0,3845	0,4320	0,285	0,3912	0,3812	0,6215

Table 6.4: Classification metrics for few-shot prompting with K=10 and K=20, where the examples are extracted with the 'random' method. In the table, consider the cases with temperature null and positive, and the case with the addition of *DREAM*'s dimension 'emotion' in the input.

Few Shot - *balanced method*							
	K=6	***K=12***	***K=18***	***K=24***	***K=30***	***K=36***	***K=42***
Temperature: T=0							
Acc@0	0,7156	0,7917	**0,8431**	**0,8331**	**0,8258**	0,7510	0,7430
Acc@50	0,6068	0,7023	**0,7503**	**0,7550**	0,7470	0,6856	0,6622
Acc@60	0,3271	0,4045	0,4346	0,4653	0,4460	0,4065	0,3705
Temperature: T=0.3							
Acc@0	0,7176	0,7877	**0,8445**	**0,8318**	**0,8204**	0,7490	0,7443
Acc@50	0,6101	0,7003	0,7470	**0,7477**	0,7363	0,6856	0,6689
Acc@60	0,3258	0,4039	0,4379	0,4466	0,4266	0,4012	0,3772

Table 6.5: Classification metrics for few-shot prompting with K = 6, 12, 18, 24, 30, 36 e 42, where the examples are extracted with the 'balanced' method. In the table, consider the cases with temperature null and positive.

For all the experiments conducted so far, the LLM used has been LLaMA-2-7b-chat-hf. It was decided to test the LLaMA-2-7b-hf model on the top 3 cases, namely those just presented with K = 18, 24, and 30 examples in input. However, as can be seen from Table 6.7, the results on the classification task are markedly inferior.
Given that the LLaMA Chat is a fine-tuned version specifically designed for chat applications, it is natural that it performs better for this task. The LLM, fine-tuned for this purpose, is indeed able to interpret better the prompt.

Few Shot - ***balanced method***							
		K=18		***K=24***		***K=30***	
Temperature		**T=0**	**T=0.3**	**T=0**	**T=0.3**	**T=0**	**T=0.3**
Test Set	Acc@0	**0,8431**	**0,8445**	**0,8331**	**0,8318**	**0,8258**	**0,8204**
	Acc@50	**0,7503**	0,7470	**0,7550**	**0,7477**	0,7470	0,7363
	Acc@60	0,4346	0,4379	0,4653	0,4466	0,4460	0,4266
IDIOM	Acc@0	**0,8040**	0,7880	0,7600	0,7640	0,7640	0,7680
	Acc@50	0,7400	0,7240	0,6840	0,6720	0,6480	0,6480
	Acc@60	0,4920	0,4640	0,4240	0,3960	0,4200	0,3760
METAPHOR	Acc@0	**0,8870**	**0,8387**	**0,7984**	**0,8105**	**0,7742**	**0,7621**
	Acc@50	**0,6371**	**0,6532**	**0,6452**	**0,6492**	**0,6411**	**0,6089**
	Acc@60	**0,3065**	**0,3024**	**0,2903**	**0,2702**	**0,2944**	**0,2742**
SARCASM	Acc@0	0,9120	0,9133	**0,9360**	**0,9320**	**0,9307**	**0,9320**
	Acc@50	0,8320	0,8253	**0,8760**	**0,8693**	**0,8813**	**0,8787**
	Acc@60	0,4627	0,4867	**0,5720**	0,5507	0,5413	0,5333
SIMILE	Acc@0	**0,6800**	**0,7000**	**0,6320**	0,6200	0,6240	0,5960
	Acc@50	**0,6280**	**0,6280**	**0,5720**	0,5560	0,5480	0,5240
	Acc@60	**0,4200**	**0,4000**	**0,3600**	**0,3600**	**0,3280**	**0,3080**

Table 6.6: Classification metrics for few-shot prompting with K = 18, 24 e 30, where the examples are extracted with the 'balanced' method. In the table, consider the cases class-specif with temperature null and positive

LLaMA-2			7B-CHAT-HF		7B-HF	
Temperature			T=0	T=0.3	T=0	T=0.3
Few-Shot	***K=18***	Acc@0	**0,8431**	**0,8445**	0,5975	0,6235
		Acc@50	**0,7503**	0,7470	0,516	0,5287
		Acc@60	0,4346	0,4379	0,2884	0,2977
	K=24	Acc@0	**0,8331**	**0,8318**	0,7323	0,7096
		Acc@50	**0,7550**	**0,7477**	0,6542	0,6335
		Acc@60	0,4653	0,4466	0,3905	0,3578
	K=30	Acc@0	**0,8258**	**0,8204**	0,6542	0,6522
		Acc@50	0,7470	0,7363	0,5567	0,5527
		Acc@60	0,4460	0,4266	0,2837	0,2797

Table 6.7: Classification metrics for few-shot prompting with K = 18, 24 e 30, where the examples are extracted with the 'balanced' method. In the table, there is a comparison between the usage of the model LLaMA-2-7b-chat-hf and LLaMA-2-7b-hf in cases with null and positive temperature.

Let us now examine the case of Chain-of-Thought prompting. As explained in Chapters 2 and 4, we decided to test this prompting method by adding, to each input sentence, the corresponding scene elaboration produced by the *DREAM* model. We analyzed five cases for each experiment done earlier: the first four cases involve adding one dimension at a time (emotion, motivation, social norm, and consequence), and the fifth case is the one where we have all four dimensions together, called *Dream SE*. For this fifth category, there is an implementation issue as the input grows significantly even with a few examples, and thus, above K=15, it already exceeds the maximum limit of context window of LLaMA. Therefore, we have results for this case only up to K=12.
In general, contrary to what was expected, it was observed that this method does not improve the results obtained from a simple few-shot approach. In Table 6.8, we have decided to report only the Accuracy@0, and only for the cases of K = 2 and K = 6, 12, 18, 24, 30 of the balanced method, with null temperature.
We can observe that only for the case of K=2 the method with the addition of the entire *Dream SE* perform better than adding a single dimension; in all other cases, the best method, for the classification task, is the one with the addition of the 'emotion' dimension.
Table 6.9 presents the results with the addition of 'emotion' in the best cases, K = 12, 18, and 24, for both temperatures. However, it is noted that these results are much lower both compared to the baseline and compared to the experiments presented before with a simple few-shot prompting.

Few Shot with *DREAM* **Temperature: T=0**						
		Emotion	*Motivation*	*Social Norm*	*Consequence*	*SE*
Accuracy@0	**K=2**	0,6115	0,5694	0,6008	0,6162	0,6676
	K=6	0,6615	0,5547	0,6175	0,5861	0,5668
	K=12	0,7423	0,6162	0,6769	0,6168	0,6702
	K=18	0,7210	0,6535	0,7036	0,6722	-
	K=24	0,7670	0,7210	0,7029	0,7230	-
	K=30	0,6742	0,6162	0,6789	-	-

Table 6.8: Accuracy@0 for few-shot prompting with the use of *DREAM* with K = 2, 6, 12, 18, 24 and 30. In the table, all the *DREAM*'s input possible: emotion, motivation, social norm, consequence and *Dream SE*, in the case with null temperature.

Few Shot with ***DREAM***						
	K=12 *+ emotion*		***K=18*** *+ emotion*		***K=24*** *+ emotion*	
Temperature	**T=0**	**T=0.3**	**T=0**	**T=0.3**	**T=0**	**T=0.3**
Acc@0	0,7423	0,745	0,721	0,6929	0,7670	0,7470
Acc@50	0,6228	0,6248	0,6121	0,6061	0,6482	0,6101
Acc@60	0,3371	0,3518	0,3224	0,3151	0,3511	0,3298

Table 6.9: Classification metrics for few-shot prompting with K = 12, 18 and 24, where the examples are extracted with the 'balanced' method. In the table, there are the cases with the adding of *DREAM*'s dimension 'emotion' with null and positive temperature.

6.2.2 Text Generation Task

To assess the quality of the generated explanations, we primarily employ two metrics: BERTscore and ROUGE. For ROUGE, we consider two types: Rouge-N, with N=1 and N=2, and Rouge-L. For each of these metrics, recall, precision, and F1-score have been computed.

The baseline results, for both the *FLUTE* model and the *DREAM-System1* model, are presented in Table 6.10. For the subsequent tables, as before, the numbers in bold will represent those that surpass the baseline of the *FLUTE* model. Instead, for clarity, in most cases only the F1 metric will be reported, excluding precision and recall, as F1 is a weighted average of the other two.

		FLUTE	**DREAM**
BERTscore	r	0,6598	0,6744
	p	0,6633	0,6890
	f	0,6601	0,6804
Rouge-1	r	0,463	0,486
	p	0,452	0,524
	f	0,443	0,490
Rouge-2	r	0,224	0,269
	p	0,223	0,275
	f	0,213	0,261
Rouge-L	r	0,413	0,449
	p	0,403	0,482
	f	0,395	0,452

Table 6.10: Text generation baseline metrics from *FLUTE* and *DREAM-System1* models. Let 'p' for precision, 'r' for recall and 'f' for F1-score

As before let's begin by analyzing the zero-shot scenario. Table 6.11 illustrates this case with zero and positive temperatures, and the case where the input is enriched with the entire *Dream SE* (i.e. all the 4 dimensions).
As expected, the metrics are considerably lower compared to the baseline, and they do not hold much significance. As said before the challenge in this scenario lies specifically in the prompt; without any examples, the model varies its output format inconsistently and it is not feasible to automatically save the requested label and explanation.

Shifting to the results obtained in the few-shot prompting scenario, let's start again with the two simplest cases: K=1 and K=2. In Table 6.12, there is the case with zero and positive temperatures, and the case where the input is enriched with *Dream SE*.

Zero-Shot			
Temperature	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*
BERTSCORE	0.5184	0.5212	0,2581
ROUGE-1	0.221	0,223	0.001
ROUGE-2	0.056	0,055	0
ROUGE-L	0,163	0,160	0.002

Table 6.11: BERTscore and ROUGE, F1 metrics for zero-shot prompting. In the table, consider the cases with temperature null and positive, and the case with the addition of *Dream SE* in the input.

It can be observed that there is not much difference either when considering the number of input examples or when considering the addition of *Dream SE*.

Few-Shot						
	K=1			***K=2***		
Temperature	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*	**T=0**	**T=0.3**	***T=0*** *+ Dream SE*
BERTSCORE	0,5706	0,5724	0,5533	0,5962	0,5970	0,5817
ROUGE-1	0,318	0,316	0.288	0,328	0,325	0.279
ROUGE-2	0,102	0,101	0.095	0,116	0,113	0.092
ROUGE-L	0,244	0,233	0.212	0,243	0,240	0.209

Table 6.12: BERTscore and ROUGE, F1 metrics for few-shot prompting with K=1 and K=2. In the table, consider the cases with temperature null and positive, and the case with the addition of *Dream SE* in the input

For the experiments with few-shot prompting involving more than 2 examples, we will present the same cases presented in the previous section.
The results for the 'random' strategy with K=10 and K=20 are presented in Table 6.13. We can observe that in general, in these two cases, the use of the *DREAM* dimension 'emotion' lowers the ROUGE metric, but in the case of K=20, it slightly increases the BERTscore.

In Table 6.14, we present the results on the Test set for the experiments conducted using the 'balanced' method with a number of input examples K = 6, 12, 18, 24, 30, 36, 42. In the table, we report the results with null and positive temperature. We can observe that, as for the classification task, there is a standard trend for both temperatures, where as the input examples increase, so do the performances until a certain saturation point is reached. After this point, any

Few Shot - ***random method***						
	K=10			***K=20***		
Temperature	**T=0**	**T=0.3**	***T=0* + emotion**	**T=0**	**T=0.3**	***T=0* + emotion**
BERTscore	0,6268	0,6254	0,6118	0,5962	0,5970	0,6245
Rouge-1	0,417	0,412	0,385	0,439	0,435	0,408
Rouge-2	0,195	0,190	0,174	0,206	0,201	0,186
Rouge-L	0,369	0,365	0,337	0,387	0,382	0,355

Table 6.13: BERTscore and ROUGE, F1 metrics for few-shot prompting with K=10 and K=20, where the examples are extracted with the 'random' method. In the table, consider the cases with temperature null and positive, and the case with the addition of*DREAM*'s dimension 'emotion' in the input.

additional examples only add "noise," causing the performances to decrease again. However, unlike the previous task, none of the configurations presented manage to reach the baseline in terms of the quality of the explanations produced. The best cases for the classification task, K = 18, 24, and 30, are reproduced in Table 6.15 for a more complete analysis.

Few Shot - ***balanced method***							
	K=6	***K=12***	***K=18***	***K=24***	***K=30***	***K=36***	***K=42***
Temperature: T=0							
BERTscore	0,6192	0,6348	0,6368	0,6418	0,6379	0,6319	0,6311
Rouge-1	0,402	0,394	0,443	0,421	0,418	0,417	0,399
Rouge-2	0,170	0,169	0,202	0,192	0,176	0,178	0,176
Rouge-L	0,338	0,331	0,384	0,365	0,352	0,356	0,344
Temperature: T=0.3							
BERTscore	0,6193	0,6352	0,6366	0,6409	0,6358	0,6330	0,6303
Rouge-1	0,398	0,391	0,439	0,425	0,407	0,420	0,397
Rouge-2	0,167	0,166	0,203	0,195	0,168	0,178	0,164
Rouge-L	0,330	0,332	0,381	0,367	0,338	0,358	0,336

Table 6.14: BERTscore and ROUGE, F1 metrics for few-shot prompting with K = 6, 12, 18, 24, 30, 36 e 42, where the examples are extracted with the 'balanced' method. In the table, consider the cases with temperature null and positive.

Let us now revisit the experiments conducted with LLaMA-2-7b-hf instead of LLaMA-2-7b-chat-hf. Regarding the classification task, as we had seen in the previous section, the results remained significantly inferior to both the baseline

Few Shot - *balanced method*							
		K=18		***K=24***		***K=30***	
Temperature		**T=0**	**T=0.3**	**T=0**	**T=0.3**	**T=0**	**T=0.3**
BERTscore	r	0,6408	0,6412	0,6453	0,6443	0,6420	0,6399
	p	0,6354	0,6347	0,6410	0,6401	0,6362	0,6340
	f	0,6368	0,6366	0,6418	0,6409	0,6379	0,6358
Rouge-1	r	0,487	0,477	0,467	0,466	0,469	0,453
	p	0,430	0,428	0,400	0,409	0,395	0,387
	f	0,443	0,439	0,421	0,425	0,418	0,407
Rouge-2	r	0,230	0,229	0,222	0,224	0,209	0,197
	p	0,195	0,196	0,180	0,185	0,165	0,159
	f	0,202	0,203	0,192	0,195	0,176	0,168
Rouge-L	r	0,424	0,415	0,405	0,405	0,396	0,378
	p	0,371	0,370	0,346	0,352	0,333	0,321
	f	0,384	0,381	0,365	0,367	0,352	0,338

Table 6.15: BERTscore and ROUGE metrics for few-shot prompting with K = 18, 24 e 30, where the examples are extracted with the 'balanced' method. In the table, consider the cases with temperature null and positive. Let 'p' for precision, 'r' for recall and 'f' for F1-score

and the corresponding case with LLaMA-Chat. However, concerning the text generation task, we can observe from Table 6.16 that the model change improves performance, even surpassing the baselines on some occasions, especially with null temperature.

Even concerning the text generation task, the addition of *DREAM* did not lead to any improvement. In Table 6.17, we report the BERTscore F1 metric in the cases of Few-shot prompting with the 'balanced' method. As can be seen, besides all metrics being below the baseline and below the corresponding case without the addition of *DREAM*, there is practically no difference among the different types of information additions used, whether adding one dimension at a time or the entire *Dream SE*. In Table 6.18, we provide more detailed results for the best cases, i.e. K=12, 18, 24, in the case of adding the *DREAM* dimension 'motivation'.

In order to improve the performance related to explanation generation, we attempted to prompt LLaMA assuming knowledge of the rhetorical figure corresponding to the input sentence of the test set to be analyzed. In this way, depending on the type of premise-hypothesis pair we need to classify, input examples will not be balanced on the general distribution of the four types of rhetorical figures, but examples related exclusively to the type of rhetorical figure to be analyzed

LLaMA-2			7B-CHAT-HF		7B-HF	
Temperature			T=0	T=0.3	T=0	T=0.3
Few-Shot	***K=18***	BERTSCORE	0,6368	0,6366	0,6192	0,6190
		ROUGE-1	**0,443**	0,439	**0,474**	**0,468**
		ROUGE-2	0,202	0,203	**0,219**	0,208
		ROUGE-L	0,384	0,381	**0,415**	**0,405**
	K=24	BERTSCORE	0,6418	0,6409	0,6325	0,6284
		ROUGE-1	0,421	0,425	**0,459**	**0,450**
		ROUGE-2	0,192	0,195	**0,221**	0,199
		ROUGE-L	0,365	0,367	**0,406**	0,387
	K=30	BERTSCORE	0,6379	0,6358	0,6204	0,6195
		ROUGE-1	0,418	0,407	**0,474**	**0,458**
		ROUGE-2	0,176	0,168	0,209	0,204
		ROUGE-L	0,352	0,338	**0,410**	0,393

Table 6.16: BERTscore and ROUGE, F1 metrics for few-shot prompting with K = 18, 24 e 30, where the examples are extracted with the 'balanced' method. In the table, there is a comparison between the usage of the model LLaMA-2-7b-chat-hf and LLaMA-2-7b-hf in cases with null and positive temperature.

Few Shot with *DREAM* **Temperature: T=0**							
			Emotion	*Motivation*	*Social Norm*	*Consequence*	*SE*
BERTscore	F1SCORE	**K=6**	0,6126	0,6115	0,6079	0,5996	0,6074
		K=12	0,6204	0,6212	0,6120	0,6099	0,6020
		K=18	0,6163	0,6172	0,6167	0,6043	-
		K=24	0,6193	0,6261	0,6135	0,6079	-
		K=30	0,6157	0,6219	0,6160	-	-

Table 6.17: BERTscore F1 metric for few-shot prompting with the use of *DREAM* with K = 6, 12, 18, 24 and 30. In the table, all the *DREAM*'s input possible: emotion, motivation, social norm, consequence and *Dream SE*, in the case with null temperature

will be passed. In this way, it is as if we "train" the model in a specific manner, thus improving its ability to provide adequate explanations. The theoretical idea is confirmed by the results, Table 6.19, where the baseline is practically always surpassed.
In Table 6.20, we expand the results for the top 3 K values in general, K = 18, 24, and 30, providing the F1 metrics specific for each class. We note that the

Few Shot with *DREAM* **Temperature: T=0**			
	K=12 *+ motivation*	***K=18*** *+ motivation*	***K=24*** *+ motivation*
BERTscore	0,6212	0,6172	0,6261
Rouge-1	0,376	0,390	0,414
Rouge-2	0,150	0,166	0,183
Rouge-L	0,316	0,339	0,347

Table 6.18: BERTscore and ROUGE, F1 metrics for few-shot prompting with K = 12, 18 and 24, where the examples are extracted with the 'balanced' method. In the table, there are the cases with the adding of *DREAM*'s dimension 'motivation' with null and positive temperature.

rhetorical figure that performs best with this specific method, is simile.

Few Shot - *class specific method*							
	K=6	***K=12***	***K=18***	***K=24***	***K=30***	***K=36***	***K=42***
Temperature: T=0							
BERTscore	0.6527	**0.6638**	**0,6619**	**0,6625**	**0,6604**	**0,6657**	0,4582
Rouge-1	**0,465**	**0,464**	**0,469**	**0,477**	**0,469**	**0,471**	**0,470**
Rouge-2	**0,230**	**0,246**	**0,242**	**0,250**	**0,238**	**0,246**	**0,241**
Rouge-L	**0,416**	**0,422**	**0,427**	**0,430**	**0,425**	**0,428**	**0,427**
Temperature: T=0.3							
BERTscore	0,6512	**0,6624**	**0,6605**	**0,6625**	0,6601	**0,6614**	0,457
Rouge-1	**0,469**	**0,466**	**0,464**	**0,469**	**0,473**	**0,468**	**0,465**
Rouge-2	**0,231**	**0,242**	**0,234**	**0,244**	**0,244**	**0,245**	**0,236**
Rouge-L	**0,420**	**0,421**	**0,424**	**0,427**	**0,433**	**0,427**	**0,457**

Table 6.19: BERTscore and ROUGE, F1 metrics for few-shot prompting with K = 6, 12, 18, 24, 30, 36 e 42, where the examples are extracted with the 'balanced' method. The prompt is personalized according to the type of rhetorical figure in input. In the table, consider the cases with temperature null and positive.

The latest analysis conducted to better evaluate the produced explanations involved a comparison between the utilized models. Specifically, we compared each of our models to both the base model of *FLUTE* and the *System1* of *DREAM*. What we did was select examples where both models being compared produced the same label, and then we evaluated ROUGE and BERTscore for the explanations produced in these cases. The second step involved evaluating the explanations

Few Shot - *class specific method*							
		K=18		***K=24***		***K=30***	
Temperature		**T=0**	**T=0.3**	**T=0**	**T=0.3**	**T=0**	**T=0.3**
IDIOM	BERTscore	0,7000	0,6992	0,7111	0,7118	0,7229	0,7212
	Rouge-1	0,529	0,527	0,545	0,551	**0,577**	**0,573**
	Rouge-2	0,338	0,339	0,344	0,346	**0,370**	0,366
	Rouge-L	0,489	0,491	0,504	0,507	**0,537**	**0,532**
METAPHOR	BERTscore	**0,6286**	**0,6277**	**0,6183**	**0,6192**	**0,6253**	**0,6265**
	Rouge-1	0,323	0,322	0,326	0,329	**0,339**	**0,345**
	Rouge-2	0,133	0,134	0,125	0,132	**0,146**	**0,149**
	Rouge-L	0,281	0,279	0,270	0,272	0,282	0,288
SARCASM	BERTscore	**0,6625**	**0,6602**	**0,6626**	**0,6623**	0,6543	0,6524
	Rouge-1	0,376	0,370	0,366	0,363	0,363	0,359
	Rouge-2	0,123	0,118	0,116	0,114	0,114	0,115
	Rouge-L	0,280	0,276	0,272	0,270	0,272	0,270
SIMILE	BERTscore	**0,6551**	**0,6550**	**0,6574**	**0,6568**	**0,6514**	**0,6554**
	Rouge-1	**0,469**	**0,464**	**0,477**	**0,469**	**0,469**	**0,473**
	Rouge-2	**0,242**	**0,234**	**0,250**	**0,244**	**0,238**	**0,244**
	Rouge-L	**0,427**	**0,424**	**0,431**	**0,428**	**0,425**	**0,433**

Table 6.20: BERTscore and ROUGE, F1 metrics for few-shot prompting with K = 18, 24 e 30, where the examples are extracted with the 'balanced' method. The prompt is personalized according to the type of rhetorical figure in input. In the table the results are divided for each figure of speech, considering the cases with temperature null and positive.

produced when the label, in addition to being corresponding, was also correctly predicted. In none of the normal cases did the explanations produced by our models surpass those of the baseline models. The only exception lies in the case where we used LLaMA, specifically class by class. In these cases, the explanations produced by the models with K=12, 18, 24, 30, and 36 are superior compared to the baseline model of *DREAM* but not compared to that of *FLUTE*.
To better understand these comparisons, we report in the Tables 6.21 and 6.22 a subset of results concerning the comparison between the two baseline models and our model obtained through few-shot prompting, using the 'balanced' method, zero temperature, and K=24.

Comparison of generated explanations				
Number of elements	Correspondence 1234/1498		Correspondence & Correctness 1200/1498	
Model	LLaMA 2	FLUTE	LLaMA 2	FLUTE
BERTscore	0,649	0,684	0,650	0,686
Rouge-1	0,369	0,455	0,371	0,456
Rouge-2	0,141	0,231	0,143	0,232
Rouge-L	0,295	0,394	0,296	0,395

Table 6.21: Comparison of the generated explanations between the *FLUTE* model and our model, prompted through a few-shot approach with K=24 and null temperature. BERTscore and ROUGE F1 metrics reported.

Comparison of generated explanations				
Number of elements	Correspondence 1138/1498		Correspondence & Correctness 1056/1498	
Model	LLaMA 2	Dream-S1	LLaMA 2	Dream-S1
BERTscore	0,645	0,665	0,649	0,668
Rouge-1	0,364	0,411	0,366	0,410
Rouge-2	0,136	0,178	0,138	0,177
Rouge-L	0,288	0,339	0,290	0,337

Table 6.22: Comparison of the generated explanations between the *DREAM-System1* model and our model, prompted through a few-shot approach with K=24 and null temperature. BERTscore and ROUGE F1 metrics reported.

6.2.3 Qualitative Analysis

Let's complete the automatic evaluation with a qualitative analysis. Specifically, we will show some examples from the best performing configuration, including two successful ones and two failures. The outputs of our model will be compared with the ground truth and with the outputs of the two baseline models, *FLUTE* and *DREAM-System1*.
The explanation score required by the FigLang 2022 challenge is reported only to provide a general indication of how much that explanation has been evaluated.

First example, a correct prediction regarding the 'Entailment' class.

Premise = I can't believe my ex won't just move on.
Hypothesis = I don't understand why my ex-boyfriend can't just move on and find someone new.
Figurative Type = sarcasm

Ground Truth:

Label = Entailment
Explanation = It's often hard to move on after a break-up and it can be even harder to see an ex-partner move on before oneself.

LLaMA model:

Predicted Label = Entailment
Model Explanation = When someone is still hung up on an ex, it can be hard for them to understand why the ex won't just move on and find someone else.
Explanation score = 56

FLUTE model:

Predicted Label = Entailment
Model Explanation = It is natural to want to move on from an ex-boyfriend who has hurt you in the past and so it is understandable to wonder why he won't do that.
Explanation score = 53

DREAM-System1 model:

Predicted Label = Entailment
Model Explanation = It is natural to want to move on from an ex-boyfriend who has hurt you and is still a person, so it is understandable to want to understand why he won't just move on.
Explanation score = 51

Given this pair of input sentences, it is easy to understand that it is an entailment, and indeed all three models correctly predict the label. We note that the explanations produced by the reference models are very similar and differ from the one produced by our model, which, however, obtains the highest score. A particular detail that we can highlight is that, while the concept of the difficulty of moving on after the end of a relationship is present in all 4 explanations, the concept that it can be even harder to see an ex-partner move on before oneself is explicitly stated only in the gold explanation.

Second example, a correct prediction regarding the 'Contradiciton' class.

Premise = My friend is always at least 30 minutes late to everything.
Hypothesis = I'm always waiting 30 minutes for my super disciplined friend.
Figurative Type = sarcasm

Ground Truth:
Label = Contradiction
Explanation = Most people would not call someone disciplined if they are consistently late and do not have good time management skills.

LLaMA model:
Predicted Label = Contradiction
Model Explanation = Someone who is consistently late is not super disciplined, they are just consistently late.
Explanation score = 60

FLUTE model:
Predicted Label = Entailment
Model Explanation = The friend is always late to everything and so the person who is waiting for them is not disciplined.
Explanation score = 54

DREAM-System1 model:
Predicted Label = Contradiction
Model Explanation = A friend who is always late to things is not disciplined and hence the person waiting for them is not disciplined.
Explanation score = 55

In this case as well, it is very easy to understand that the two input sentences contradict each other. However, unlike the first example, not all three models correctly predict the label. The baseline model *FLUTE* erroneously predicts the label and creates a rather confusing explanation; however, we note that the obtained score is not even so low.
The other reference model, *DREAM*, although correctly predicting the label, does not produce a clear explanation. In fact, it states that the person waiting for the latecomers is not disciplined, which cannot be inferred from the input sentence pair and is not present in the ground truth. The best output is from our model, which not only assigns the correct label but also generates a correct and comprehensive

explanation.

Third example, a wrong prediction regarding the 'Entailment' class.

> *Premise = The animal looked like a small German shepherd, but with a reddish-brown coat and a black mask over its eyes.*
> *Hypothesis = With its black mask and those ears, the animal was like a small german shepherd, but the rest of its fur was like a chestnut mare's.*
> *Figurative Type = simile*

Ground Truth:

> *Label = Entailment*
> *Explanation = A chestnut mare is a horse with a reddish brown coat, so saying the animal's fur was like a chestnut mare's would mean it would have a reddish brown coat*

LLaMA model:

> *Predicted Label = Contradiction*
> *Model Explanation = A chestnut mare is a type of horse with a reddish-brown coat, so it would not be correct to say that the animal's fur was like a chestnut mare's.*
> *Explanation score = 75*

FLUTE model:

> *Predicted Label = Contradiction*
> *Model Explanation = A chestnut mare is a reddish-brown horse, so saying its fur is like s chestnut mare's would imply that its coat is red and not brown.*
> *Explanation score = 74*

DREAM-System1 model:

> *Predicted Label = Contradiction*
> *Model Explanation = A chestnut mare's coat is reddish-brown, not red*
> *Explanation score = 54*

This third example presents a more challenging premise-hypothesis pair to classify as it requires a deeper understanding of the language and of the vocabularies. All models under consideration have indeed failed in the label assignment phase. The reference model *DREAM* performs the worst, while the *FLUTE* and LLaMA models, on the other hand, generate two explanations that are very similar to each other and both with very high scores (74 and 75). This high score is due to the fact that, although incorrect explanations are provided, they are very similar to the gold one, with only a few words changing. For example, between the ground truth and the explanation generated by our model, practically only a 'not' differs. So, the sentences are similar to the gold one in terms of syntax but differ in terms of semantics.

Fourth example, a wrong prediction regarding the 'Contradiciton' class.

Premise = He's very smart.
Hypothesis = He's smart as a brick.
Figurative Type = simile

Ground Truth:
Label = Contradiction
Explanation = A brick is an inanimate object, therefore it would not be considered smart.

LLaMA model:
Predicted Label = Entailment
Model Explanation = Smart as a brick means that someone is very intelligent and so this statement entails that the person being referred to is indeed smart.
Explanation score = 47

FLUTE model:
Predicted Label = Entailment
Model Explanation = To be smart as a brick means to be very smart, which is what the speaker is trying to convey.
Explanation score = 42

DREAM-System1 model:
Predicted Label = Contradiction
Model Explanation = A brick is not a very smart object, so saying someone is as smart as a brick would imply that they are not very smart.
Explanation score = 56

The last example provides us with another pair that is very easy to classify, but surprisingly only the *DREAM* model manages to correctly predict the label. The explanation provided by this model is correct and complete, but it is presented syntactically differently from the gold standard, so the explanation score is not actually that high. On the other hand, the *FLUTE* and LLaMA models generate two explanations that are completely wrong from a semantic point of view. Also the syntax is different from the gold standard, so the final scores are low.

We can conclude this qualitative analysis by saying that although the LLM used in our experiments performs quite well, the so-called bottlenecks of Figurative Language Understanding are still present.
The main issues in our case are twofold. When the model predicts correctly, it's still possible that evaluating the generated explanation results in a low score. This occurs because the two sentences, while having the same semantic meaning, differ too much in terms of syntax.
The second problem is practically the opposite, it occurs when the model fails to

predict the correct label, but evaluating the explanation produced we get an high score. This happens because the generated text, compared to the ground truth, differs semantically but remains very similar syntactically.

7 Conclusions

In general, we can say that this work once again demonstrates the enormous potential of LLMs. Observing the results obtained, we notice that our model almost always reaches the baseline proposed by the authors of the FigLang2022 challenge and, in some cases, even surpasses it. For example, in the case of few-shot prompting with examples extracted through the 'balanced' method, the model is able to achieve an Accuracy@0 between 83% and 85%, surpassing the 81.68% of the reference model *FLUTE*.
This is a remarkable outcome because unlike the baseline and all other proposed solutions, our model is not trained or fine-tuned specifically for this task.

Taking a comprehensive look at all the experiments conducted, we can also list some of the weaknesses of this method.
The first observation is that the LLM is greatly influenced by how the prompt is written. In fact, during the initial phase, various types of prompts were tried, all with the same purpose, but for some, the model did not provide the sought-after response at all.
And the second one is that the quality of the response is highly influenced by the type of examples provided as input. Taking examples randomly is not a good choice; they should be selected from the Train set with a criterion if higher performance is desired.

In overall experiments, we have noticed that on average the model seems to struggle with the figure of speech known as "idiom." This figure, being the most complex, requires the model to have a more advanced reasoning ability and a higher level of abstraction. Conversely, the rhetorical figure that appears to be better identified is the metaphor.

Analyzing the methodologies of zero-shot and few-shot prompting, one of the initial conclusions we can draw is that the zero-shot method is not suitable for this task. Indeed, without any input examples, the model varies its output format inconsistently and given the lack of coherence with the required format, so, it is not feasible to automatically save the requested label and explanation.
On the other hand, moving to the few-shot case and analyzing the number of examples to give in input, we can reach a couple of conclusions. When K is less than 10, there are too few input examples, particularly since we deal with 4 different classes of figures of speech. Conversely, when K exceeds 30, there are too

many input examples; this abundance of data might introduce too much noise into the process, potentially hindering the model's ability to accurately learn patterns and make predictions. To achieve optimal model performance in this few-shot case is hence crucial to strike a balance in the number of examples used.
For this case, we have observed that if the figurative class membership is known in advance, the model predicts better. This happens because, assuming knowledge of the class, only examples consistent with that specific class are passed as input. Therefore, one could consider a different model that first predicts the class and then provides this prediction to the LLM to better guide it in generating the label and its explanation.

Regarding the use of *DREAM*, it can be concluded that it hasn't yielded the improvements initially envisioned. The idea was that adding context to the pair of sentences to be analyzed would facilitate classification by our LLM. However, this hasn't been the case. The reason behind this could be that this approach requires a sufficiently large number of examples (18, 24, 30) as input to work effectively. This is because the rhetorical figures to be analyzed are divided into 4 distinct classes, and there isn't an initial training phase specific to the task. So we think that the problem arises here because *DREAM* adds at least one contextual sentence for each input sentence. Therefore, even with few input examples an excessive noise is transfer to the model and moreover the maximum token threshold accepted by LLaMA is quickly reached.
With a full fine-tuning, as seen in *DREAM-FLUTE*, it would probably be possible to better utilize the additional context provided by *DREAM*. However, with an LLM, the model appears to be more sensitive to noise, leading to no improvement in performance.

Future Works

Here we present the possible modifications or future work that can be carried out based on this work.
The first idea is to try other variations of the pipeline used, such as solving the two tasks separately, with two separate prompts, or even using two different models. Alternatively, as mentioned earlier, predicting the type of rhetorical figure before and then passing it as input to the LLM could also help.
Additionally, one could consider using a LLaMA Chat model version 13B or 70B, which will perform better compared to the version we used, the 7B.
Alternatively, like the one used in the FigLang 2022 challenge, one could train or fine-tune LLaMA before prompting it. This method is certainly more computationally expensive, but it should better refine the model for the task of figurative

language understanding, and with a "trained" model, performance is likely to improve.
Of course, all these experiments can be reproduced using other LLMs, even non-open-source ones such as GPT-3.5 or GPT-4. In this last case is therefore possible to further compare the performance of open-source models with proprietary ones.

Bibliography

[1] Demystifying the temperature parameter: A visual guide to understanding its role in large language models., 2023.

[2] Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. ferret: a framework for benchmarking explainers on transformers. In Danilo Croce and Luca Soldaini, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 256–266, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[3] Irina Bigoulaeva, Rachneet Singh Sachdeva, Harish Tayyar Madabushi, Aline Villavicencio, and Iryna Gurevych. Effective cross-task transfer learning for explainable natural language inference with t5. In Debanjan Ghosh, Beata Beigman Klebanov, Smaranda Muresan, Anna Feldman, Soujanya Poria, and Tuhin Chakrabarty, editors, *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP)*, pages 54–60, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics.

[4] Jose Camacho-Collados and Mohammad Taher Pilehvar. Embeddings in natural language processing. In Lucia Specia and Daniel Beck, editors, *Proceedings of the 28th International Conference on Computational Linguistics: Tutorial Abstracts*, pages 10–15, Barcelona, Spain (Online), December 2020. International Committee for Computational Linguistics.

[5] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. 2018.

[6] Tuhin Chakrabarty, Debanjan Ghosh, Adam Poliak, and Smaranda Muresan. Figurative language in recognizing textual entailment. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3354–3361, Online, August 2021. Association for Computational Linguistics.

[7] Tuhin Chakrabarty, Arkadiy Saakyan, Debanjan Ghosh, and Smaranda Muresan. FLUTE: Figurative language understanding through textual explanations. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language*

Processing, pages 7139–7159, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[8] Tuhin Chakrabarty, Arkadiy Saakyan, Debanjan Ghosh, and Smaranda Muresan. A shared task on understanding figurative language, 2022.

[9] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable AI for natural language processing. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China, December 2020. Association for Computational Linguistics.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[11] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. Eraser: A benchmark to evaluate rationalized nlp models. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online, July 2020. Association for Computational Linguistics.

[12] Denis Emelin, Ronan Le Bras, Jena D. Hwang, Maxwell Forbes, and Yejin Choi. Moral stories: Situated reasoning about norms, intents, actions, and their consequences. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 698–718, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[13] Maxwell Forbes, Jena D. Hwang, Vered Shwartz, Maarten Sap, and Yejin Choi. Social chemistry 101: Learning to reason about social and moral norms. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 653–670, Online, November 2020. Association for Computational Linguistics.

[14] Debanjan Ghosh, Elena Musi, and Smaranda Muresan. Interpreting verbal irony: Linguistic strategies and the connection to theType of semantic incongruity. In Allyson Ettinger, Gaja Jarosz, and Joe Pater, editors, *Proceedings of the Society for Computation in Linguistics 2020*, pages 82–93, New York, New York, January 2020. Association for Computational Linguistics.

[15] Yuling Gu, Bhavana Dalvi, and Peter Clark. DREAM: Improving situational QA by first elaborating the situation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1115–1127, Seattle, United States, July 2022. Association for Computational Linguistics.

[16] Yuling Gu, Yao Fu, Valentina Pyatkin, Ian Magnusson, Bhavana Dalvi Mishra, and Peter Clark. Just-DREAM-about-it: Figurative language understanding with DREAM-FLUTE. In Debanjan Ghosh, Beata Beigman Klebanov, Smaranda Muresan, Anna Feldman, Soujanya Poria, and Tuhin Chakrabarty, editors, *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP)*, pages 84–93, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics.

[17] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online, July 2020. Association for Computational Linguistics.

[18] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[19] Yash Kumar Lal and Mohaddeseh Bastan. SBU figures it out: Models explain figurative language. In Debanjan Ghosh, Beata Beigman Klebanov, Smaranda Muresan, Anna Feldman, Soujanya Poria, and Tuhin Chakrabarty, editors, *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP)*, pages 143–149, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics.

[20] Perci Liang. Stanford-cs324, Winter 2022.

[21] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[22] Chi-kiu Lo. YiSi - a unified semantic MT quality evaluation and estimation metric for languages with different levels of available resources. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 507–513, Florence, Italy, August 2019. Association for Computational Linguistics.

[23] Scott Lundberg and Su-In Lee. Axiomatic attribution for deep networks. 2017.

[24] Andreas Madsen, Siva Reddy, and Sarath Chandar. Post-hoc interpretability for neural nlp: A survey, 2021.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[26] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[27] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[28] Khoa Thi-Kim Phan, Duc-Vu Nguyen, and Ngan Luu-Thuy Nguyen. NLP@UIT at FigLang-EMNLP 2022: A divide-and-conquer system for shared task on understanding figurative language. In Debanjan Ghosh, Beata

Beigman Klebanov, Smaranda Muresan, Anna Feldman, Soujanya Poria, and Tuhin Chakrabarty, editors, *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP)*, pages 150–153, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics.

[29] Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. Modeling naive psychology of characters in simple commonsense stories. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2289–2299, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[30] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. “why should I trust you?”: Explaining the predictions of any classifier. In John DeNero, Mark Finlayson, and Sravana Reddy, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June 2016. Association for Computational Linguistics.

[31] Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. Inseq: An interpretability toolkit for sequence generation models. In Danushka Bollegala, Ruihong Huang, and Alan Ritter, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, Toronto, Canada, July 2023. Association for Computational Linguistics.

[32] Thibault "Sellam, Dipanjan Das, and Ankur" Parikh. "bleurt: Learning robust metrics for text generation". In Dan "Jurafsky, Joyce Chai, Natalie Schluter, and Joel" Tetreault, editors, *"Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics"*, pages 7881–7892, Online, 2020. Association for Computational Linguistics.

[33] Hiroki Shimanaka, Tomoyuki Kajiwara, and Mamoru Komachi. RUSE: Regressor using sentence embeddings for automatic machine translation evaluation. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 751–758, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[34] Sia. Adaptation to downstream tasks, 2023.

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014.

[36] Miloš Stanojević and Khalil Sima'an. BEER: BEtter evaluation as ranking. In Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, and Lucia Specia, editors, *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[37] Kevin Stowe, Prasetya Utama, and Iryna Gurevych. IMPLI: Investigating NLI models' performance on figurative language. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5375–5388, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[38] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. A unified approach to interpreting model predictions. 2017.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. 2023.

[40] Hugo Touvron, Louis Martin, Kevin Stone, and Yasmine Babaei Nikolay Bashlykov Soumya Batra Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic Sergey Edunov Thomas Scialom Peter Albert, Amjad Almahairi. Llama 2: Open foundation and fine-tuned chat models. 2023.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[42] Lena Voita. Nlp course | for you, 2018.

[43] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, and Hyung Won Chung. What language model architecture and pretraining objective work best for zero-shot generalization? 2022.

[44] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2022.

[45] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. 2020.

[46] Wayne Xin Zhao, Zou Khun, and Yunji li. A survey of large language models, 2023.

[47] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. 2023.

[48] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized BERT pre-training approach with post-training. In Sheng Li, Maosong Sun, Yang Liu, Hua Wu, Kang Liu, Wanxiang Che, Shizhu He, and Gaoqi Rao, editors, *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China, August 2021. Chinese Information Processing Society of China.