Alma Mater Studiorum · Università di Bologna

DIPARTIMENTO DI INTERPRETAZIONE E TRADUZIONE
Corso di Laurea Magistrale in Specialized Translation (LM-94 R)

TESI DI LAUREA
in NATURAL LANGUAGE PROCESSING

On the Automatic Captioning of Gastronomic Procedural Pictures

CANDIDATA:
Elena Benini

RELATORE:
Alberto Barrón Cedeño
CORRELATRICE:
Maja Miličević Petrović
CORRELATORE:
Paolo Gajo

Anno Accademico 2024/2025
Terzo Appello

Acknowledgements

I wish to express my deep gratitude to all the individuals who contributed to the completion of this thesis.

This endeavor would not have been possible without my supervisor, Alberto Barrón Cedeño, who introduced me to an interdisciplinary field that naturally aligns with my wide-ranging curiosity and desire to engage with many disparate topics.

I would like to express my deepest gratitude to my co-supervisor, Paolo Gajo, for the dataset and for the many hours he spent helping me hunt down bugs. His explanations helped me understand the finer details of the topic and are the reason why my understanding of it has improved as much as it did compared to when I started working on this project.

Many thanks to my co-supervisor Maja Miličević Petrović, for her patience and availability, as well as for her teachings about statistics, which helped make my work better.

I'm extremely grateful to Eurac Research and the people I met during my internship there, as that experience helped prepare me to undertake this work. The deduplication algorithm we developed during my time with them, which they allowed me to reuse, proved very useful in the course of this project.

Finally, I could not have undertaken this journey without the support of my family and friends. They put up with my disappearances whenever I was too absorbed in the work, and soldiered through my tendency to answer questions on my thesis with over-technical ramblings in a mixture of languages.

GenAI Use

In the course of this project, Gen AI was used to assist in the creation of the Python scripts. This was mainly to assist in troubleshooting and error debugging, as well as to help ensure compatibility with the libraries that the project leverages. The models used for these purposes were OpenAI's ChatGPT-5 and Google's Gemini 2.5 Pro.

Abstract

Automated image captioning using artificial neural networks allows for applications that go beyond the creation of a description in natural language of the visual information contained in an image. This work explores the use of image captioning to generate the instructions to perform a gastronomic procedure depicted by an input picture. To do this, the model must learn to focus on the appropriate visual elements of the image, as well as to mimic the required style of the captions. A multilingual dataset of recipes is used to fine-tune an English vision encoder-decoder model, and to prefix-tune an Italian model built using CLIP as an image encoder and mGPT as a linguistic decoder with a lightweight network to bridge the modalities. Lack of context that goes beyond the individual image causes the most issues, but both of the resulting models perform well overall. This is especially evident in the case of the English fine-tuned model. However, most of the automated metrics used struggle to reliably evaluate the quality of the results. BERTScore fares the best among them, both when only the baseline BERT model is used and when the model is adapted to the domain. The presence of noisy references probably contributes to the issues encountered during the evaluation, but is certainly not the only factor. In short, while this kind of non-standard application of image captioning can be modeled successfully, the selection of appropriate evaluation metrics is non-trivial, and a time-consuming manual evaluation may be necessary for a fully informed assessment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Developments in the natural language processing field have seen the rise of multimodal applications that perform tasks that until recently posed significant challenges. One of these tasks is image captioning, which leverages advances in both computer vision and natural language processing. Typically, the process uses a neural model to generate a description in natural language of a given picture, with consequent applications ranging from accessibility to information retrieval, from image indexing to medical imaging (Osman et al., 2024). The advances made for these established use cases equally support non-standard purposes, such as generating text in a format other than a description of the input picture.

Leveraging a dataset of recipes, this work explores the possibility of creating models to generate cooking instructions when given in input a picture depicting the action. The model needs to replicate the linguistic style of the dataset, giving guidance on how to perform the selected step of the recipe. To achieve this, the output text should accurately describe the action performed, as well as the ingredients and tools involved. Consequently, the model needs to have in-depth knowledge of the gastronomy domain and its specialized terminology, not to mention the ability to recognize domain-specific equipment and items from visual information.

These requirements, in addition to the atypical style of the desired captions, mean that general-purpose models are unlikely to be able to perform the task without further instruction, with the potential exception of the most recent state-of-the art multimodal architectures. However, the latter have high memory and hardware requirements, making them challenging to deploy in practical settings. For this same reason, their use would impose

limits on the adoption of any real-world application derived from this work. Therefore, adapting a pretrained model to the domain and this specific task is probably the most effective strategy within the external limitations imposed on this project.

The experiments conducted saw the creation of two different models for this task, one for English and one for Italian. Their architectures differ slightly, owing to the different pretrained models used as base.

Despite this difference, both models showed similar behavior, occasionally struggling with lack of context, but correctly mimicking the required style. However, the evaluation of the results proved challenging, as many automated metrics can struggle when applied to a highly specialized and sometimes noisy dataset. This work examines these aspects in detail and is organized as follows.

Chapter 2 provides an overview of the background knowledge necessary to understand the problem and the way the experiments engaged with it. The overview starts with introducing the field of natural language processing, and progressively narrows the focus onto the architectures used in the experiments. It then continues with a brief overview of common approaches to the image captioning task. Lastly it includes information on the evaluation metrics used in this project.

Chapter 3 discusses the dataset used to fine-tune the models. It explains why Italian and English were preferred to other languages in the experiments, and details the various steps used to clean and preprocess the data. It also examines the biases implicit in the data.

Chapter 4 describes the experiments conducted and discusses their results, both for Italian and for English. It also discusses the adaptation of the BERT model used during the evaluation of the Italian experiment.

Chapter 5 concludes by summarizing the work and by drawing conclusions. Additionally, it introduces possible directions for future work.

Chapter 2

Background

2.1 Natural Language Processing

Natural Language Processing (NLP), is a subfield of computer science and artificial intelligence that focuses on enabling computers to process natural language data (Hincal, 2023; Stryker and Holdsworth, 2025). It is thus an interdisciplinary field that uses knowledge from linguistics, data mining, and statistics among others (Hincal, 2023).

From computer science, NLP inherits an approach aimed at problem-solving. However, linguistic phenomena often contain complexities that make the clear definition of a problem challenging, if not impossible (Nivre, 2001). Consequently, attempts to create algorithms capable of solving abstract problems can often fail. For this reasons, NLP typically uses mathematical models to approximate reality, the results of which depend on the quality of the model (Nivre, 2001).

There is no restriction on the modality of the language data a model may focus on; NLP can deal with speech or text. Regardless of whether the natural language was originally in audio or text format, it needs to be converted into numerical representations that are *understandable* to machines (Stryker and Holdsworth, 2025). Appropriate representation techniques are required for each modality, leading first to specialized systems and later, as the field developed, to integrating multiple modalities into a single model (ConfigrTechnologies, 2024).

Multimodal systems are currently the cutting-edge of the field, combining two or more of audio, text, and images (ConfigrTechnologies, 2024; Grando,

2025). The ability to analyze and integrate information from different modalities can improve the system's understanding of the nuances of natural language, by including some of the many complexities of human communication that go beyond the mere words used, such as tone and gestures (Grando, 2025).

The first approaches to NLP were rule-based, leveraging if-then-else decision trees (Masoumzadeh, 2023; Stryker and Holdsworth, 2025). This type of algorithm predicts a response to the input data based on the rules defined during its construction (The MathWorks, Inc., 2026). In the case of NLP, these rules are derived from syntax, grammar, and other language data, such as dictionaries (Pirinen, 2023).

A simple example of a decision for a sentiment analysis task could be *if the word "wonderful" is present in the review, consider it positive* (Masoumzadeh, 2023). A decision tree would include multiple conditions similar to this one, creating a multi-step process. Yet, this example already shows some of the challenges of this approach. Indeed, this condition does not consider the possibility the word "wonderful" could appear accompanied by a negation, or that it might be used ironically. This shows one of the challenges of implementing this algorithm: all possible cases and exceptions have to be considered in advance and included in the handcrafted ruleset, or they will not be taken into account by the model. Their inclusion might not even be possible within the structural limitations imposed by the algorithm.

An example of successful rule-based algorithm for sentiment analysis is VADER. It is based on a lexicon of words that express a sentiment, scoring them on a scale from -4 to $+4$, thus capturing both their polarity (positive or negative) and the intensity of the sentiment expressed. Other features of the input sentence are also considered, such as the use of exclamation marks, degree adverbs, or *all-caps*, all of which increase the intensity of the sentiment expressed without changing its polarity. Negation and the conjunction "but" also influence the final score assigned to the sentence (Hutto and Gilbert, 2014). This example shows an advantage of the rule-based approach, which is the ease of explaining why and how the model reached a result. In the case of VADER, this is easily done by looking at its rules and the values in the lexicon (Hutto and Gilbert, 2014). However, understanding the results and the process that led to them is not always easy with other types of models (Pirinen, 2023).

The introduction of machine learning marked a significant step forward in the field, as it allowed computers to model probabilistic aspects of language

from data, typically large textual corpora (Masoumzadeh, 2023; Stryker and Holdsworth, 2025). This learning process is referred to as training a model. Statistical systems tend to frame problems in terms of optimization, such as maximizing the probability of the expected output given a certain input (Nivre, 2001). Their reliance on data means they can struggle if insufficient examples of a certain linguistic phenomenon have been provided (Masoumzadeh, 2023), which greatly disadvantages languages that lack large amounts of digital data to use in the creation of these models (Pirinen, 2023).

Additionally, statistical models frequently apply deterministic algorithms (Nivre, 2001). For example, when faced with a word with multiple possible translations in the target language, a statistical machine translation system may choose the term that is most probable according to its training data, which often considers a limited context. As a result, such systems can fail to capture the full richness of natural language.

Further strides forward were made with the introduction of neural networks and deep learning, which are currently the dominant paradigm (Stryker and Holdsworth, 2025).

A neural network is a type of machine learning model that can approximate different families of mathematical functions depending on its structure. It can be conceptualized as a network of computing nodes, as shown in Figure 2.1. Each node or neuron in the hidden layer performs mathematical operations in order to go from the numerical representation of the input to that of the output (Prince, 2023). These operations generally consist of dot products between the input and the weights the model learned during training (see Appendix). Additionally, the layer can include an activation function, which allows the model to apply non-linear transformations (Prince, 2023). Each progressive step of the elaboration makes the representation of the input more abstract and complex, which makes it difficult to explain the internal *reasoning* that leads to the production of an output (Pirinen, 2023).

A shallow neural network contains only one hidden layer, whereas a deep neural network contains more than one, greatly increasing the variety and complexity of the relationships it can represent (Prince, 2023).

Neural models learn the value of the weights required to represent the relationship between the given input and the desired output through the training process, which is done by exposing them to a large amount of data (Stryker and Holdsworth, 2025).

These data can be annotated or not. A dataset is considered annotated when a human manually specified the desired output associated to an in-

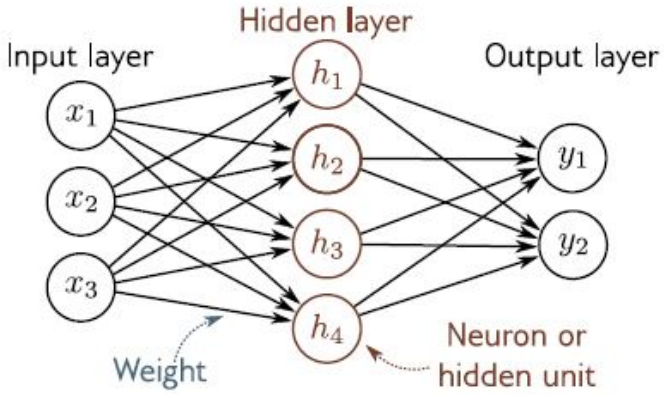


Figure 2.1: Structure of a shallow neural network (Prince, 2023).

put instance (Delua, 2025). For example, in a classification task where the model is expected to distinguish pictures of dogs and cats, the annotations would be the appropriate class labels ("dog" or "cat") that the model is expected to produce when analyzing the corresponding input picture. These input-output pairs provide *supervision* during the learning process, when the model iteratively makes a prediction, compares it against the annotation, and adjusts its internal weights so the next attempt will be closer to the desired output. Consequently this is called supervised learning (Delua, 2025; Prince, 2023).

On the other hand, unsupervised learning is when models are trained on unannotated data. In this case, a model learns to recognize patterns and anomalies within the data without the explicit guidance provided by annotations (Delua, 2025; Vankayalapati, 2024). This can be a better approach when patterns can be derived from raw sequences of text, often making the construction of a large dataset faster and cheaper than when annotations are employed. Ultimately, the type of data and methodology suitable for a task is often determined by the nature of the task itself, not to mention by the data available (Delua, 2025).

This last point, data availability, is not to be underestimated. Annotating data is a time-intensive task that often requires specialized knowledge. With neural models being extremely data-hungry, having a large fully annotated dataset may not be possible. For this reason, semi-supervised learning has emerged as a third alternative. A middle ground between the other two

methodologies, semi-supervised learning makes use of unsupervised learning enhanced by some amount of annotated data (Delua, 2025).

2.2 The Transformer Architecture

The foundation of many, if not most, modern NLP systems is the Transformer architecture, first introduced in the paper "Attention is all you Need" (Vaswani et al., 2017). Transformers are neural models capable of learning complex representations and are known for their ability to capture long-range dependencies between words, which previous neural models struggled with (Masoumzadeh, 2023).

Their strengths derive from their innovative sole reliance on the attention mechanism (Vaswani et al., 2017), which allows them to pay attention to certain relevant portions of the input while generating the output (Kulshrestha, 2020). Additionally, self-attention allows them to establish similar connections within the same sentence, creating a better understanding of its nuances when applied to the input, and ensuring better internal cohesion and semantics when used to generate the output (Kulshrestha, 2020). This is done through assigning a value to all other words in the sentence based on how "related" they are to the word currently being examined (Alammar, 2018).

Figure 2.2 shows an example of how self-attention works for the sentence *The animal didn't cross the street because it was too tired*, where the intensity of the color associated with each unit in the sentence represents how important they are to understand the meaning of the word *it* (Alammar, 2018). This is only one of the attention patterns generated by the model during its analysis of the input sentence. Transformer models contain a multi-head-attention module (see Figure 2.3) which allows them to create multiple attention patterns simultaneously. Each one focuses on different features of the input and captures different aspects of the meaning and structure of a sentence, allowing the model to consider multiple such dependencies and create a rich representation (Alammar, 2018; Masoumzadeh, 2023; Nguyen, 2023). These aspects do not need to be preprogrammed into the model, rather they emerge as part of the training process (Nguyen, 2023).

When used for sequence-to-sequence tasks, that is to say, tasks that take a sequence of text as input and generate a sequence of text as output (though modalities other than text are also possible), Transformers are built with an

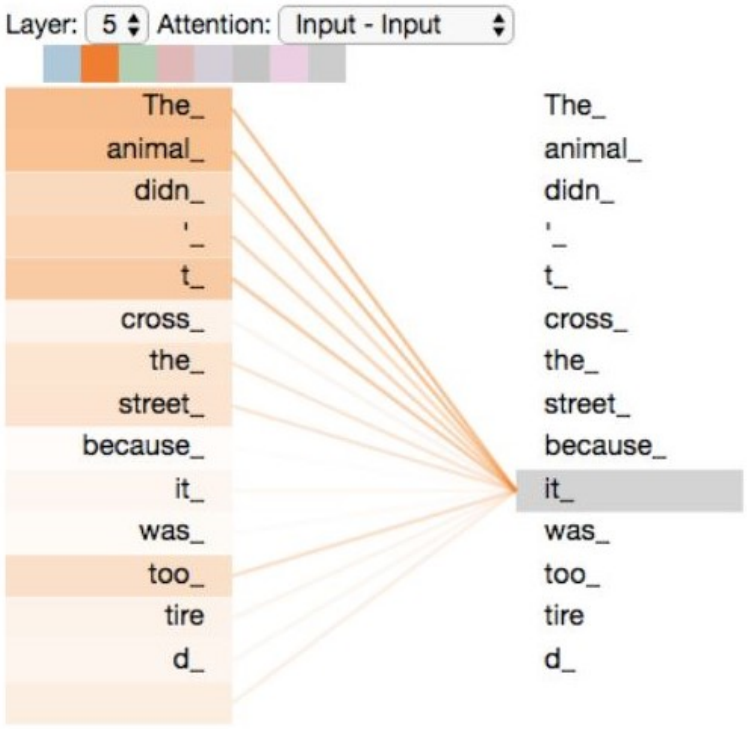


Figure 2.2: Illustration of the attention mechanism (Alammar, 2018).

encoder-decoder structure (Kulshrestha, 2020; Vaswani et al., 2017) as shown in Figure 2.3. The encoder, depicted on the left, is a stack of a certain number of layers, each of which can be broken down into a self-attention sub-layer and a feed forward sub-layer (Vaswani et al., 2017). The latter refines the contextual information provided by the attention (Nguyen, 2023). It is worth pointing out that while all the encoder layers share the same structure, each contains its own parameters (Alammar, 2018).

The decoder, shown on the right in Figure 2.3, contains a matching number of decoder layers, although their internal structure is different (Vaswani et al., 2017). In between the two sub-layers already described when discussing the encoder, the decoder contains an additional sub-layer dedicated to cross-attention, ensuring the model pays attention to relevant features of the input while producing the output (Alammar, 2018).

Depending on whether it is an encoder or decoder layer, each block produces a different output. Within the encoder this is progressively more refined

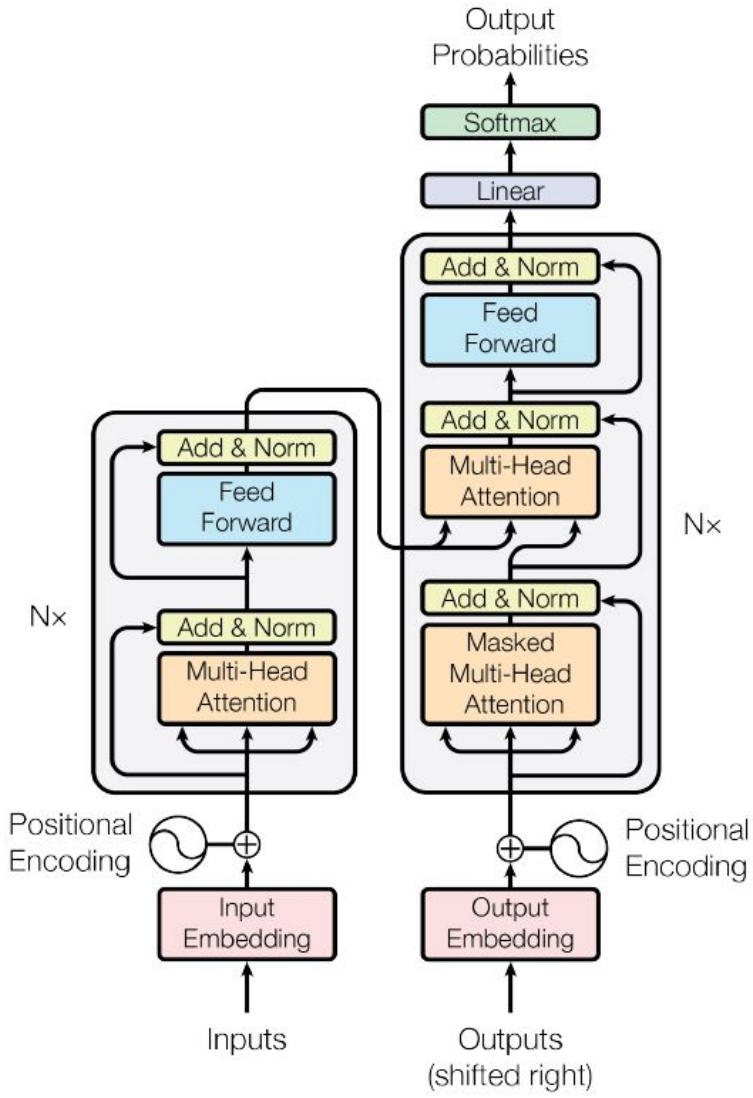


Figure 2.3: Transformer structure as introduced in (Vaswani et al., 2017).

and context-rich embeddings which then become the input to the decoder (Nguyen, 2023). Passing through the decoder layers transforms this input into logits, raw predictions which are then normalized to produce the probability distribution used to identify each next token to generate (Nguyen, 2023).

Tokens are generated iteratively, which is why Figure 2.3 shows the outputs (shifted right) being fed into the decoder. The shift makes the decoder input tokens lag behind the desired output by one position, hiding the token to be generated during the current iteration from the decoder. For example, if the desired output was *I am hungry*, in order to generate the word *hungry* the encoder would see a special beginning of sentence token followed by the words *I am*.

This shift and the masked self-attention layer shown at the bottom of the decoder stack in Figure 2.3 ensure the decoder is aware of all previously generated tokens, but cannot see future tokens in the sentence, not even during training. The output is thus at any moment conditioned on the previous elements of the sequence (Vaswani et al., 2017).

Another innovation the Transformer architecture leverages is the addition of positional embeddings to the input embeddings, marking the absolute position of each word in the sequence, thus keeping track of word order (Masoumzadeh, 2023; Vaswani et al., 2017) without the need for recurrence.

Discarding the convolutions and recurrence used by previous neural models speeds up computing and makes it possible to parallelize both training and execution (Vaswani et al., 2017). This translates to lower costs and better performances.

Scalability is another strength of this architecture. An increase in size can be achieved by adding more layers with the same inner structure as those already present, or increasing the number of parameters per layer, thus reaching the massive sizes of modern large language models, often with a corresponding improvement in performance (Prince, 2023).

All parameters, including those that compute attention, are learned during training, which requires large amounts of data (Nguyen, 2023). For this reason most Transformer models, be they encoder only, decoder only, or sequence-to-sequence, first undergo unsupervised pretraining on a large unlabeled dataset, and are later fine-tuned on a smaller task-specific dataset (Nguyen, 2023).

2.3 Vision Transformer

The Transformer architecture has proven extremely versatile and has been adapted to modalities other than text. This versatility is key to multimodal applications, since it provides a unifying underlying architecture that uses

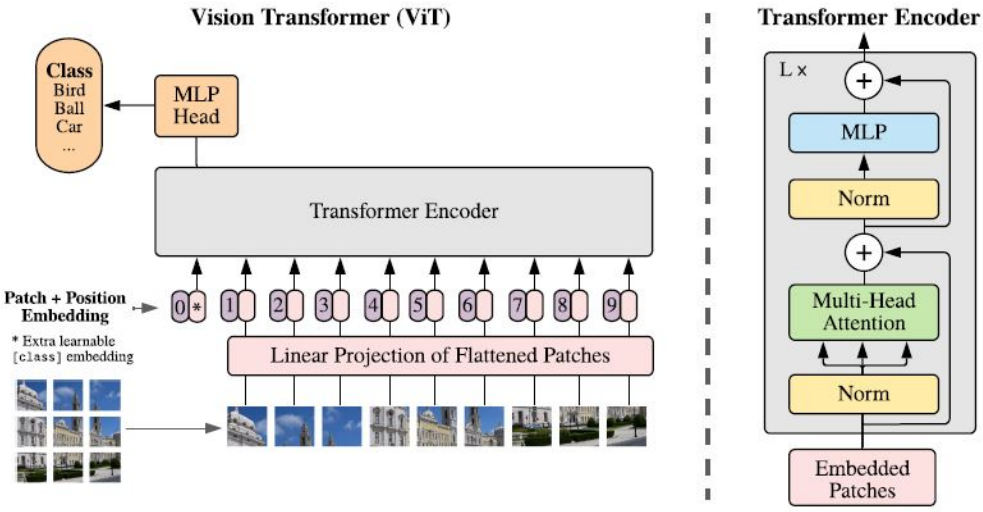


Figure 2.4: Overview of the Vision Transformer (Dosovitskiy et al., 2020)

embeddings and attention regardless of modality (Grando, 2025).

In the case of images, the adaptation of the Vision Transformer, frequently shortened to ViT, is done with the least possible amount of modifications to the architecture (Dosovitskiy et al., 2020). However, while the Transformer introduced in (Vaswani et al., 2017) was a sequence-to-sequence architecture, the Vision transformer is primarily conceived as an encoder (Callis, 2024) and was originally presented for use in classification tasks (Dosovitskiy et al., 2020).

While language models work with tokenized sentences as input, the Vision Transformer divides the input picture into evenly sized areas, called patches (Callis, 2024; Dosovitskiy et al., 2020), as shown on the left side of Figure 2.4. These patches are first flattened and linearly projected to the required dimension, before adding to them the position embeddings that allow the model to understand the spatial relationships between the various portions of the image (Callis, 2024; Dosovitskiy et al., 2020). The resulting embeddings are then passed to the encoder, whose inner architecture is essentially unchanged compared to the original Transformer (Dosovitskiy et al., 2020). Figure 2.4 shows this structure on the right. Aside from the differences in visual style, it is the same design as the encoder block on the left side of Figure 2.3. Here too the encoder features a multi-head-attention to capture multiple depen-

dencies between the various units of the input, followed by a feed forward layer, in this case indicated by the letters MLP, which stand for multi-layer perceptron, another name for this type of layer.

2.4 Image Captioning

Image captioning is a task at the intersection of computer vision and natural language processing, enabling computers to *translate* pictures into coherent text (He et al., 2020; Megne, 2023; Suresh, 2024). It is therefore a task that needs the use of a multimodal model trained on aligned pairs of visual and textual data instances, specifically a large number of input pictures and the corresponding desired output captions (Grando, 2025). The caption text should describe all important aspects of the corresponding image, avoiding omissions or off-topic associations that could lead the model to learn spurious correlations (Grando, 2025).

The initial approach to image captioning has been translational, combining an image encoder capable of extracting the salient characteristics of the input picture and a linguistic decoder capable of turning that into a coherent text (He et al., 2020). In such a setup, cross-attention is the logical way of bridging the modalities, allowing the decoder to pay attention to the salient features of the input image while generating the caption (He et al., 2020).

Further adaptation of the Transformer architecture to the specification of these tasks has been attempted, giving the Image Transformer thus created a different internal structure, with an increased number of multi-head attention modules capable of encoding or decoding the complex spatial relationships between different regions of the image (He et al., 2020). Others sought an alternative technique for bridging the gap between modalities, such as prefix-tuning (Mokady et al., 2021), which will be discussed more in depth in Section 4.2.

A milestone among the approaches to the task was the family of models named Flamingo (Alayrac et al., 2022b). This type of model is built by combining a large language model and a visual model through the addition of innovative layers to join the two architectures. As the model can take in input an undetermined number of visual elements interleaved with text, a smaller module composed of multiple layers, called a Perceiver Resampler, is used to reduce the arbitrary number of visual features elaborated by the encoder to a fixed number of elements. Additionally, a cross-attention between the two

modalities attends to the output of the Resampler, bridging the modalities (Alayrac et al., 2022b).

Most of the layers of the original models combined to form this architecture were frozen during training, allowing for a rapid adaptation of the model to a multitude of downstream tasks using carefully curated yet unannotated multimodal data from the Web (Alayrac et al., 2022a).

Given visual input and a text prompt, Flamingo is capable of performing multiple tasks, such as generating text in the shape of answers to open or closed questions related to the visual input, as well as creating descriptive captions, among other (Alayrac et al., 2022a,b). In order to do this and achieve competitive results with models extensively fine-tuned on a single task, Flamingo needs to be provided with a few examples of the task, performing what is termed few-show learning (Alayrac et al., 2022b; Grando, 2025). In short, the vision model is used to analyze the visual input, while the large language model elaborates the data and generates the required answer. This is an example of the strategy called intermediate fusion, in which the visual and textual inputs are elaborated separately at the start, then merged halfway through the process, before further elaboration takes place on the joint embeddings (Pulapakura, 2024).

The introduction of models such as Gemini or Chameleon that are natively multimodal (Google AI Blog, 2023; Grando, 2025; The Sage, 2024) has once again changed the paradigm. These models are capable from inception of handling multiple modalities and thus use early fusion, merging the representations of the different modalities at the input level (Pulapakura, 2024). This allows the seamless integration of different modalities and allows them to achieve state-of-the-art results on many tasks (Google AI Blog, 2023; The Sage, 2024).

2.5 BERT

BERT (Devlin et al., 2018) is a model built as a Transformer encoder, which makes it excellent at capturing the contextual and semantic structure of a text (Pruthvi, 2025).

This ability derives from its pretraining using a technique called masked language modeling (MLM), which consists in a random 15% of tokens being masked during training while the model is required to output the original unmasked sentence (Devlin et al., 2018). Among the masked tokens, 80% are

replaced using the special [MASK] token, whereas the remaining is equally split between being replaced with a random token and being left unchanged.

Despite their strengths, generic BERT models can struggle to deal with specialized data without being fine-tuned (Amit, 2024).

2.6 Evaluation Metrics

Regardless of the specifics of the architecture or how it was trained, models almost never perform perfectly at inference. Consequently, part of the process of developing a model consists in evaluating the quality of the result it produces (Sabetzadeh and Arora, 2024; Saravanan, 2021).

To do this, often a portion of the original data is excluded during training and kept aside for evaluation. In other words, the dataset is split into a training and a testing set (Leelapisuth, 2025). Any overlap between the two is detrimental to the reliability of the evaluation, as it can inflate the scores due to the model memorizing instances it was exposed to during training (Lee et al., 2022), instead of learning to recognize patterns and applying this knowledge at inference time. If the model can successfully recognize these patterns on never seen before data it means it has learned to generalize (Saravanan, 2021).

Another method to evaluate a model, often used when there is a limited amount of data available, is k-fold cross validation. In this case the dataset is divided into k subsets or folds. A model is trained on all these folds except for one, which is used to perform the evaluation. This process is repeated until each of the folds has been used for evaluation exactly once. The evaluations are then averaged, and the final model is trained on all the data available (Leelapisuth, 2025).

However, the use of an appropriate testing dataset is not enough to guarantee a reliable evaluation. The choice of which metric used for the assessment also plays a major role, so ensuring the metric reported is appropriate to the model and the task it performs is just as important as the values obtained (Sabetzadeh and Arora, 2024).

For example, BLEU (Papineni et al., 2002) is an extremely common metric used to compare a generated text to a reference, due to the ease and speed of its computation. However, it has well-known weaknesses, among which stands out the fact that it takes neither meaning nor sentence structure into account (Tatman, 2019). While it may still be useful when applied

to a task such as machine translation, it is less than ideal for text generation tasks that are not constrained to close parallelism to the input, such as image captioning.

Different metrics may also highlight different aspects of a model's performance, thus it may perform better according to some metrics and worse according to others. Using multiple metrics is therefore advisable (Saravanan, 2021). In this work, four metrics are used to assess the quality of the models developed. ChrF++ and BERTScore compare the generated text against the reference text, CLIPScore compares the former against the input picture, and RefCLIPScore takes all three elements — picture, generated text, and reference text — into account.

2.6.1 ChrF++

ChrF++ is a metric first developed for machine translation evaluation (Popović, 2015, 2017), and as such compares a text against a reference text (Popović, 2017). It is based on the chrF score, which calculates the overlap of character n-grams present in the candidate and reference text. To do this, it computes the F_1 measure combining precision (how many character n-grams from the candidate are present in the reference) and recall (how many character n-grams from the reference are present in the candidate text) (Popović, 2015). ChrF++ extends chrF with the addition of word uni-grams and bigrams for a better correlation to human judgment (Popović, 2017).

Like BLEU, this metric is fast and language independent (Popović, 2017), but its use of character n-grams makes it less sensitive to exact spelling and morphology. Indeed, whereas BLEU requires exact word matches and treats different inflected forms or spelling variants as distinct, chrF++ can reward even partial matches (Hassan, 2025).

2.6.2 CLIPScore

CLIP (Radford et al., 2021) is a multimodal model with the ability to map both text and images to the same embedding space. This means that the embeddings it produces for either type of input are comparable across modalities (Rustamy, 2023).

CLIPScore takes advantage of this, resulting in a multimodal metric that does not require a reference. Instead, it compares the embeddings of an

image with those of a text, and calculates the cosine similarity between the two (see Appendix), making it especially suited to evaluate tasks such as image captioning (Hessel et al., 2021).

2.6.3 RefCLIPScore

RefCLIPScore extends CLIPScore by considering a reference text in addition to the input image and output text previously mentioned. To achieve this, it computes the cosine similarity of the candidate caption against the reference (see Appendix) and calculates the harmonic mean between this value and the result of CLIPScore (Hessel et al., 2021).

The use of the harmonic mean means that situations where one similarity is much better than the other are penalized, and more weight is assigned to the lower value (Kamat, 2023), making the overall measure more balanced and conservative.

2.6.4 BERTScore

BERT (Devlin et al., 2018), as introduced in Section 2.5, is a model that employs context embeddings, vectorial representations of a word that do not just reflect the word itself, but also the context in which it appears (Devlin et al., 2018; Zhang et al., 2019).

BERTScore is a metric obtained by calculating the cosine similarity (see Appendix) between the context embeddings of two sentences (Zhang et al., 2019), which means it takes semantics into account. In addition to this strength, BERTScore is also relatively fast, robust and correlates well with human judgements (Zhang et al., 2019).

In this work, two BERT models were used to calculate this metric: the first, "dbmdz/bert-base-italian-uncased" [1], was used for Italian, while the second, "google-bert/bert-base-uncased" [2], was used for English. The uncased model was preferred for both languages due to the input texts featuring irregular capitalization at the beginning of the instances (see Section 3.2), thus the choice of these models should prevent this characteristic from skewing the scores.

[1] `https://huggingface.co/dbmdz/bert-base-italian-uncased`
[2] `https://huggingface.co/google-bert/bert-base-uncased`

Chapter 3

Data

For the image captioning task a multimodal dataset that pairs an image with a corresponding text or caption is required. This makes the Giallo Zafferano website (Giallo Zafferano, 2026) an ideal source, as each step of every recipe is aligned with a picture of the action described in the text, as shown in Figure 3.1.

Figure 3.1: Recipe instruction as displayed on (Giallo Zafferano, 2026).

3.1 Languages

The website is available in multiple languages, although not all recipes have been translated into all languages. This opens the question about which language (or languages) is best suited to use in this project. The choice is non-trivial as some options bring different advantages and disadvantages to the table, especially in light of the fact that the texts in most of the languages available are translations.

Italian, as the source language, has the largest amount of material available (∼7k recipes). Gastronomy is also an area of widespread interest in Italian culture; therefore, it can be assumed the domain-specific language is well-known and understood among most speakers, which in turn is likely reflected in the contents of this website. This makes it a good dataset to use to train neural models to perform tasks related to this domain.

However, the language itself is not as widespread as others, which means fewer pretrained models will be available for adaptation. The dataset being of moderate size means that training a model from scratch is not possible even when disregarding other issues like the available timeframe or computing power.

On the other hand, the English recipes are fewer in number (∼2k) and are the result of a translation process, with the unavoidable ensuing characteristics. A translation has to balance fidelity to the source material with the sometimes arduous task of fluently expressing the same concept in the target language, which results in effects like simplification and interference that effectively mean the translated language can be considered a dialect of the original (Volansky et al., 2015). This is especially relevant when fine-tuning a pretrained model, as there is a chance the non-standard language variety might make it harder for the model to learn to recognize linguistic patterns. (Srivastava and Chiang, 2024). Increasing the amount of data can help overcome these difficulties, and this strategy is especially effective for lexical and semantic variation (Srivastava and Chiang, 2024), which are also concerns when adapting to a highly specialized domain and its specialized terminology.

The quality of the translation is important as well, because models may also struggle when dealing with text that contains misspellings, typos, and grammatical errors (Srivastava and Chiang, 2024). The specific translation technique used for this website is unconfirmed, though comparison of source

and target recipes suggests post-edited machine translation. The reason for this belief comes from the fact that translation features a sometimes slightly awkward adherence to the flow of the source. For instance this can happen when the source text uses long intricate sentences, which are not as common in everyday Enslish as in Italian. At the same time, the translation tends to appropriately localize expressions and geographical references beyond the mere language change, inserting contextual knowledge that is not obvious to non-Italian readers. The first characteristic suggests the work of an automated system, while the second makes some kind of human intervention likely. With the latter being assumed, it was also assumed the human translator or post-editor would have caught and corrected any major language issues or mistakes, making the resulting dataset acceptable with respect to quality.

Given the prominence of the English language in the NLP field (Ore, 2022), these issues may be balanced out by a much wider selection of pretrained models available, with the consequent option of fine-tuning the one most suited to the task within the external limitations of this project.

The website is also available in other languages, such as French or Spanish. These share with English the issues that arise from being translated texts, and with Italian the availability of a smaller number of pretrained models. Given the lack of upsides, they were excluded from consideration.

The final decision was to run experiments with both English and Italian, using different architectures depending on the pretrained models available.

3.2 Cleaning and Preprocessing

The raw dataset contains 7,116 recipes, each of them having an Italian version and some an English version. A total of 67 recipes are missing some portion of the data, either the pictures or in one case the text, making them unusable for this task.

Excluding these, the textual data was extracted from the other recipes using regex, leveraging sentence boundaries and the numbering of the steps to identify instruction boundaries.

Since an instance may contain multiple steps, this sometimes resulted in incomplete sentence snippets. The possibility of discarding the shortest and presumably least meaningful among them was considered, but closer examination revealed this to be unadvisable. Instructions such as “aggiungete il

lievito" and "e il sale" share the same number of tokens, yet one is a complete and clear instruction that does not need further context to be understood. Even shorter valid instructions exist, such as "e servitelo," in which the first token could easily be omitted without making the instruction any less meaningful. In other words, the length of a text snippet is not a good indicator of its usefulness for the purpose of this task.

The exploration of the data further revealed that the use of formatted and unformatted apostrophes was inconsistent in the dataset, so all instances of the former were replaced with the latter as part of the cleaning process. This was done because the use of one type of apostrophe over the other is aleatory and not a relevant pattern for the model to learn and imitate.

Each step was then matched with the corresponding image thanks to the numbering included within the text itself and in the image filenames.

Some of the recipes turned out to contain a mismatched number of instructions and images. Manual examination of some of these instances showed that the numbering of some instructions had been omitted in the text, but the instructions where the numbering was present were correctly labeled and matched to the corresponding picture. This could have potentially made it possible to include only the correctly numbered pairs in the final dataset. However, ascertaining this was always the reason for the mismatch would have required manually checking all the recipes involved, which was deemed too large a task to manage in a reasonable timeframe. As such, all the recipes where the mismatch occurred were excluded from the dataset.

From the English raw data 606 recipes had to be discarded due to containing such a mismatch, resulting in 1,554 recipes being used. From these, an initial count of 31,365 image-instruction pairs were obtained.

From the Italian raw data, 2,188 recipes presented the same issue, which resulted in the use of 4,861 recipes, for an initial count of 96,360 image-instruction pairs.

At this point these raw pairs were screened for duplicates, as deduplication of a dataset has been shown to improve a model's ability to generalize (Aghabagherloo et al., 2025; Lee et al., 2022). This assessment was done for both images and instructions.

The texts were lowercased and stripped of numbers, punctuation, and extra spaces to make the search slightly fuzzy. The images were hashed with the MD5 algorithm, which turned each of them into an alphanumerical string, with identical images resulting in the same string (Godse, 2023). Even a single dot of a different color is enough to obtain a different hash string, as

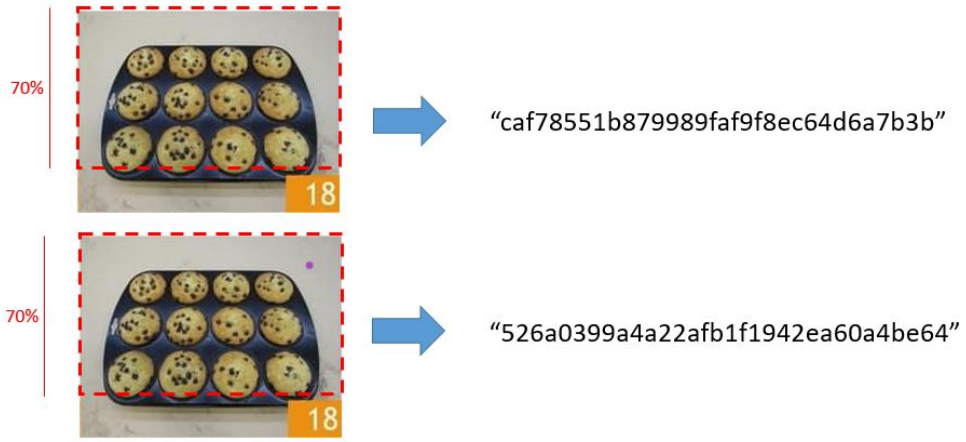


Figure 3.2: Example of the use of the MD5 algorithm.

shown in Figure 3.2. Due to this sensitivity to the minimum variation, the algorithm was applied only to the upper 70% of the image. This excluded the numbering in the lower right corner, which is not always present and is likely to vary even if the same picture is reused in different recipes.

Regardless of whether the string thus obtained was a cleaned text or a hash string, the deduplication itself was performed by leveraging the fact that the lookup of a dictionary key in python has a constant cost. In other words, starting from an empty dictionary, the cleaned text was compared to the keys already present in it. If a match was found, the instance was marked as duplicate, and if not, it was added to the dictionary. Then the process was repeated for the next instance until all instances had been checked. Only image-text pairs in which both elements were duplicates were removed from the dataset.

In English, 3,741 text duplicates were found, as well as 44 image duplicates. However, only 10 instances contained duplicate elements for both image and text, resulting in a final dataset of 31,355 image-text pairs.

In Italian, the number of text duplicates turned out to be 13,535 and that of image duplicates 258. The number of pairs in which both elements were duplicate was instead 143, resulting in a final number of 96,217 image-text pairs. The final dataset statistics are summarized in Table 3.1.

The dataset was then deterministically shuffled and divided between training, development, and testing, each containing 85%, 5%, and 10% of instances

	English	**Italian**
Initial Number of Recipes	7,116[1]	7,116
Recipes Missing Data	4,956[2]	67
Recipes Containing all Data	2,160	7,049
Recipes with Image-Picture Mismatch	606	2,188
Recipes Used to Build the Dataset	1,554	4,861
Initial Number of Image-Caption Pairs	31,365	96,360
Caption Duplicates	3,741	13,535
Image Duplicates	44	258
Both Image and Caption Are Duplicate	10	143
Final Number of Image-Caption Pairs	31,355	96,217

Table 3.1: Dataset Statistics

respectively.

As a last preprocessing step these subsets were converted into tensors using the appropriate tokenizer and feature extractor and saved in that format. This avoided having to repeat this portion of the process with every test during the experimental phase, speeding up the work.

3.3 BERT-Specific Preprocessing

Fine-tuning BERT required a slightly different preprocessing of the dataset. First of all, only one language was used for this task, Italian, as only one BERT model was adapted. Secondly, only textual data was necessary, which made it possible to include text portions from the original data that do not have the same close alignment with pictures. This means that the recipes presenting a mismatch between the number of pictures and instructions could be included. Additionally, the paragraph presenting the recipe was also extracted from every recipe and added to the tally, as it too contains fluent in-domain text.

[1]Not all recipes are translated, but an estimate of the original number of English recipes is difficult to obtain due to some recipes missing some data. See next row for more details.

[2]The count includes untranslated recipes, as well as recipes that are missing a portion of the relevant data.

In order for the model to learn context and long-range dependencies, splitting the dataset in short snippets would have been counterproductive. Instead it was split into paragraphs, and each paragraph became a data instance. This preserves as much context as possible without exceeding the model's maximum allowed length in token.

This made the deduplication less effective: while no duplicate paragraphs were found using the technique described in Section 3.2 it is entirely possible the paragraphs may contain duplicate sentences or portions of sentences. The potential presence of any such repeated subsection was deemed to be less of a problem than damaging the context to remove them would have been.

The data was then lowercased to match the specifics of the model chosen, deterministically shuffled and split into training, development, and testing in the same percentages used before, 85%, 5%, and 10%. As a last step before training, 15% of tokens, selected randomly, were masked, matching BERT's original training strategy.

3.4 Biases

Learning representations from data means that models also learn the inherent biases contained within such data (Klein and D'Ignazio, 2024), which makes knowledge of the dataset used and of its limitations a necessary element of the process of building a model.

The data used for this project comes from an Italian website, which means the recipes it contains are mainly from Italian gastronomy. Any foreign recipe present has likely been adapted to Italian taste and product availability. Consequently, any model trained on this dataset will yield better results when used to perform inference on data with these same qualifications, and may not achieve the same performance when used on recipes from other gastronomic traditions.

Chapter 4

Experiments and Results

A translational approach was chosen for this project, thus requiring a model with an encoder capable of transforming the input image into a representation the decoder can work with, while the decoder itsel is aimed at producing text starting from the encoding of the image (He et al., 2020). Two different ways to achieve this were implemented, one for each of the languages used, leveraging different pretrained models.

Each of the sections in this chapter details the fine-tuning process and evaluation of a different model. The first two are image-captioning models developed for the main task, one for each of English and Italian. For each of these a qualitative discussion of the results is included in order to better understand the scores assigned by the chosen metrics. The third section discusses the adaptation of the BERT model used for evaluation in Section 4.2.

4.1 Baseline and Fine-Tuned Model

For English, the pretrained model "ydshieh/vit-gpt2-coco-en" [1] was used as a starting point. This is a Vision Encoder Decoder Model, built using a Vision Transformer as the encoder and GPT2 as the decoder, thus fulfilling the qualifications for the chosen translational approach.

It is worth noting that this model is trained on a more traditional style of captions, so it tends to output nominal sentences to describe the image. Consequently, fine-tuning serves the double purpose of domain and style

[1] `https://huggingface.co/ydshieh/vit-gpt2-coco-en`

Model	ChrF++	CLIPScore	RefCLIPScore	BERTScore
Baseline	27.786	**0.282**	0.407	0.430
Fine-Tuned	**33.807**	0.275	**0.419**	**0.649**

Table 4.1: Comparison between the results of the baseline and adapted English models in terms of four metrics.

adaptation.

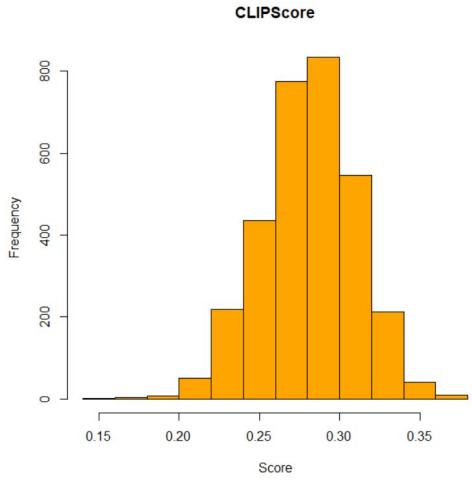
The model was fine-tuned on the English dataset, matching its original training language. A first implementation was attempted using the Hugging Face trainer class [2]. However, ensuing issues caused the strategy to be abandoned in favor of a PyTorch loop [3]. The training was set to run for 5 epochs with a learning rate of 5×10^{-5}. Early stopping with patience 2 was used. Fine-tuning ran for 3 epochs, with the best validation results being achieved after the first epoch.

The median value of the evaluation is reported for CLIPScore, RefCLIPScore and BERTScore. The median is used because applying the Shapiro-Wilk test on a sample of size 1000 of the scores showed that nearly all their distributions are close to but still significantly differ from a normal distribution, as shown in Figures 4.1 and 4.2. This makes the mean an unsuitable measure of central tendency for these results (Mishra et al., 2019). The lone exception that does not significantly differ a normal distribution is the BERTScore of the baseline model. In this case the median was reported for consistency with the other measures.

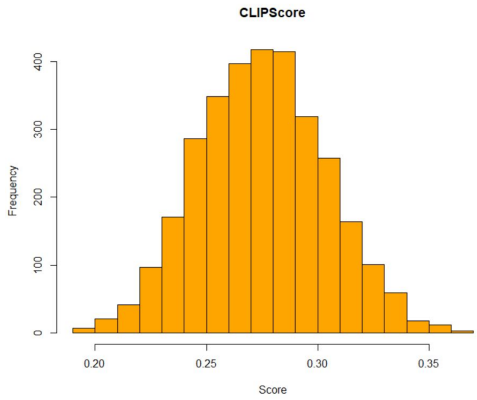
Unlike the other metrics, ChrF++ was computed for every individual instance and at the corpus level. The latter score is the one reported in Table 4.1, so there was no need to use a measure of central tendency for it, nor was checking the shape of the distribution necessary. The results of the Shapiro-Wilk Test for ChrF++ are nevertheless included in picture Figure 4.2 for completeness.

All results of the Shapiro-Wilk tests are listed below the histograms of the respective score distributions in Figures 4.1 and 4.2.

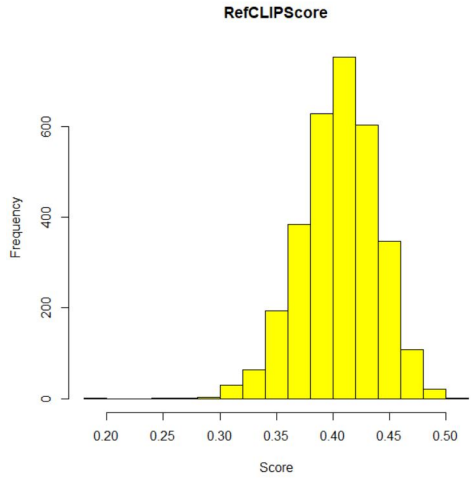
[2] `https://huggingface.co/docs/transformers/en/main_classes/trainer`
[3] `https://docs.pytorch.org/docs/stable/index.html`



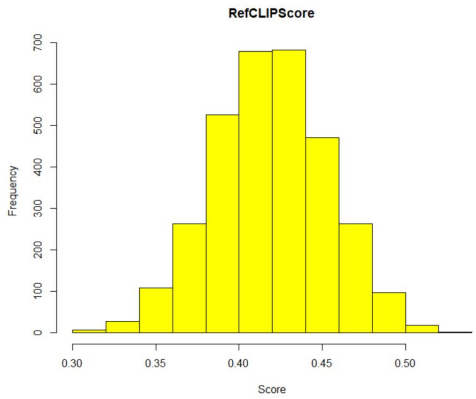
(a) CLIPScore (Baseline)
W=0.9964, p-value=0.0215



(b) CLIPScore (Fine-Tuned)
W=0.9986, p-value=0.0106

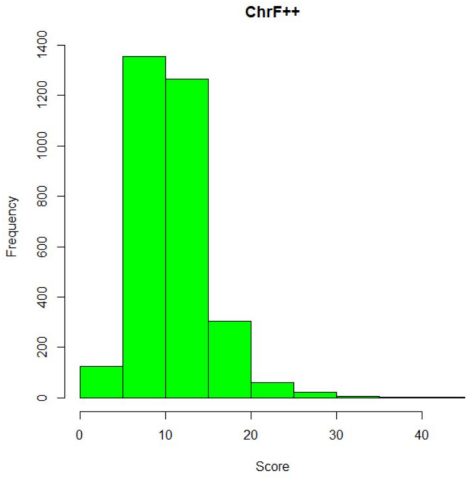


(c) RefCLIPScore (Baseline)
W=0.9843, p-value=6.879e-9

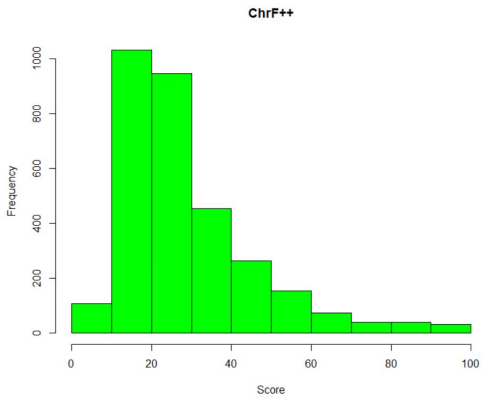


(d) RefCLIPScore (Fine-Tuned)
W=0.9961, p-value=0.0124

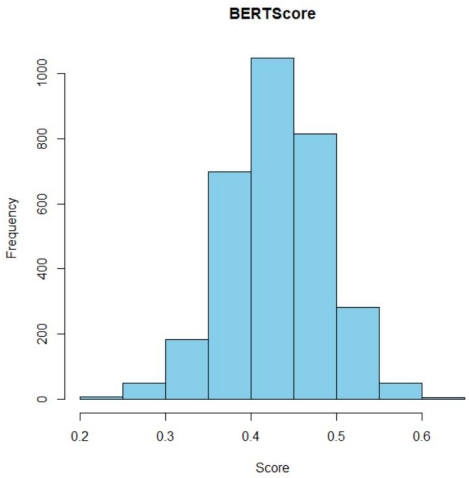
Figure 4.1: Histograms of the distributions of CLIPScore and RefCLIPScore (English). The baseline model is on the left and the adapted model on the right. W refers to the Shapiro-Wilk Test.



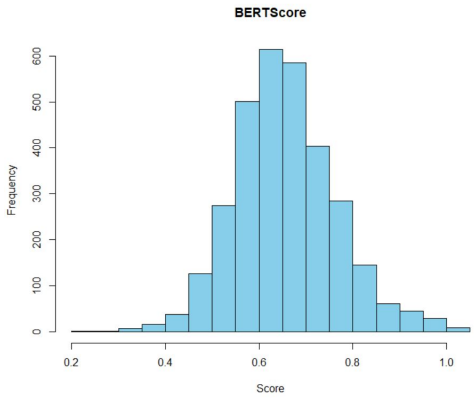
(a) ChrF++ (Baseline)
W=0.9414, p-value<2.2e-16



(b) ChrF++ (Fine-Tuned)
W=0.8573, p-value<2.2e-16



(c) BERTScore (Baseline)
W=0.9978, p-value=0.2159



(d) BERTScore (Fine-Tuned)
W=0.9937, p-value=0.0003

Figure 4.2: Histograms of the distributions of ChrF++ and BERTScore (English). The baseline model is on the left and the adapted model on the right. W refers to the Shapiro-Wilk Test.

The scores reported in Table 4.1 show that the training improved the English model's performance according to all metrics except CLIPScore, whose

value instead worsened with adaptation.

Comparing the distributions of the scores of both models for the same metric, it is easy to see the improvement reported by both ChrF++ and BERTScore. The distributions associated with the two models have a somewhat similar overall shape, but the ones associated with the adapted model cluster around higher scores and extend to the highest possible ranges, which they did not reach in the baseline model. Low scores are still present, but they have clearly diminished in number.

On the other hand, CLIPScore and RefCLIPScore show minimal improvement in the range of their scores, with fewer instances being evaluated in the lower end of the range for the adapted model, but no correspondingly noticeable increase in the upper range. CLIPScore also shows a major change in the frequency of the most common values. While the range of the results remains somewhat similar, the peak density of the mode is only about half as high in the adapted model compared to the baseline. The lower medians suggest the density increased in the lower ranges instead. This leads to questions about why these two metrics, CLIPScore especially, report results so markedly different from the other two.

A reason for the small improvement or lack of improvement reported by some of the metrics may be the fact that the original model was trained on a different style of captions. Consequently, it tends to output text as nominal descriptive sentences, such as "a bowl of food with a spoon in it", which differs significantly from the style of the reference captions. Examining the captions generated by the adapted model confirms that it successfully learned to mimic the style of the fine-tuning dataset. However, having to learn to output text in a different style certainly made the adaptation more difficult, which is probably reflected in the results.

The results of the evaluation performed using CLIPScore and to a lesser degree RefCLIPScore probably suffered due to this difference in format. CLIP's tendency to perform better when the text associated with the picture follows the template "a photo of..." was noticed since its creation (Radford et al., 2021), and this format does not match that of the captions in this dataset, nor that of the captions generated by the adapted model. However, the baseline model tends to output results in a format that is closer to the one CLIP favors, so this may be the reason why CLIPScore evaluated the baseline model better than the adaptation.

In order to obtain a more reliable evaluation of the fine-tuned model using CLIPScore, adding the string "a picture of..." to the dataset for further

experiments was considered, but in the majority of cases it would have made the resulting sentence nonsensical. Consequently, it would have likely worsened most if not all the scores, so the idea had to be discarded. This is before even considering how such an addition would have negatively impacted the final purpose of the captions, making the resulting instructions less fluent and harder to follow.

Among the metrics used, BERTScore does not take into account the image part of the dataset, but it is known for considering semantics (Zhang et al., 2019). The fact that this metric showed the greatest improvement shows that the format of the caption is not the only aspect influencing these results.

A qualitative analysis of the results of the adapted model confirms that there is more to consider. Table 4.2 shows some examples of generated captions alongside their input image, their reference, and their individual scores. This is not a representative sample of the model's results; these specific examples were chosen because they offer insights relevant to the discussion.

In Example 1, the hypothesis and reference caption match perfectly, yet only ChrF++ and BERTScore reflect this. CLIPScore assigns a low similarity score, suggesting that, according to CLIP, the image and caption are poorly matched. RefCLIPScore too assigns a score below the model's median, further supporting this theory.

It is possible CLIP does not consider the images and references captions paired in the dataset to be good matches, which means it may consistently underscore the performance of a model trained on this data. This discrepancy also calls into question the reliability of any score assigned by CLIPScore or RefCLIPScore when used on this dataset. A systematic and time-consuming manual evaluation of the results is likely required for definitive confirmation, but even without it caution may be advised when deriving insights from these scores.

Example 2 in Table 4.2 shows a different situation in which the generated caption is excellent, especially considering it was generated from the picture alone without any further context. It is also completely different from the reference, which is reflected in all scores. However, in this case it is probably the reference which is lower quality, as it contains unnecessary noise.

Example 3 in Table 4.2 shows a case in which the only correct part of the hypothesis caption is the numbering. The seemingly ungrammatical style of the caption follows some of the patterns seen in the dataset, so the performance is better than it may appear at first glance. Additionally, the picture

<table>
<tr><th></th><th>Input</th><th>Results</th><th></th></tr>
<tr><td>1</td><td>  </td><td>Hypothesis: Deglaze with white wine <12>
Reference: Deglaze with white wine <12>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BS</th></tr><tr><td>100.000</td><td>0.247</td><td>0.397</td><td>1.000</td></tr></table></td><td>✓</td></tr>
<tr><td>2</td><td>  </td><td>Hypothesis: and then whip it with an electric whisk <26>
Reference: After the cooling time, you can focus on the decoration: whip the cream with the powdered sugar <26>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BS</th></tr><tr><td>14.397</td><td>0.216</td><td>0.346</td><td>0.592</td></tr></table></td><td>✓</td></tr>
<tr><td>3</td><td>  </td><td>Hypothesis: and pepper <11>
Reference: and use a blowtorch to obtain a crunchy, caramelized crust <11>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BS</th></tr><tr><td>9.973</td><td>0.235</td><td>0.366</td><td>0.583</td></tr></table></td><td>×</td></tr>
<tr><td>4</td><td>  </td><td>Hypothesis: Remove the broccoli from the middle stalk <1>
Reference: To prepare the coconut fried rice, clean the broccoli, remove the outer leaves, separate the florets <1>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BS</th></tr><tr><td>23.336</td><td>0.346</td><td>0.493</td><td>0.681</td></tr></table></td><td>∼</td></tr>
</table>

Table 4.2: Qualitative results of the fine-tuned English model. For readability the names of the metrics were shortened to CF for ChrF++, CS for CLIPScore, RCS for RefCLIPScore, and BS for BERTScore. The mark on the right represents whether the caption should be considered as correct.

could be considered slightly ambiguous. While a human would be capable of inferring from contextual clues that there is no way the picture could represent the hypothesized instruction, it is also evident where the model's incorrect assumptions originated.

Considering the model's limited ability to infer context, together with the

fact that the original pictures are small and not always of the highest quality, this picture could potentially be assumed to represent the use of a tool to scatter an ingredient on the preparation, an ingredient that is darker than the underlying material. After these considerations, the model's guess no longer appears as far-fetched an interpretation. Nevertheless, it is incorrect, yet both CLIPScore and RefCLIPScore rate it higher than the previous example, casting further doubt on their reliability on this dataset.

Another pattern the model appears to have captured successfully is the fact that instructions numbered "<1>" should start with the beginning of a sentence, as shown in Example 4, as well as several other instances. The instruction generated for this example is more direct and less noisy than the reference, although the omission of the word "floret" may cause momentary confusion since "broccoli" typically refers to the whole vegetable. Overall it can be considered a suitable caption for the input picture, yet, once again, it would be difficult to reach this conclusion from the scores alone, even though they report better results than for some of the previous examples.

From these examples, BERTScore emerges as probably the most reliable metric, so BERT was fine-tuned on this dataset in the hope it would improve the model's accuracy for the evaluation of the next image captioning experiment.

4.2 Prefix-Tuned Model

A pretrained model fulfilling the required characteristics could not be found for the Italian language, thus requiring a different approach. This led to using two independently trained models, one for images and one for text, with the addition of a lightweight mapping network in between to bridge the modalities as in ClipCap (Mokady et al., 2021).

The model chosen to encode the image is CLIP, which is trained to map images and their associated captions to the same embedding space through the use of contrastive loss. In other words, during the training process, the model learns to maximize the cosine similarity (see Appendix) of the embeddings of matching image-caption pairs and minimize that of mismatched pairs (Radford et al., 2021; Rustamy, 2023). To this purpose, CLIP is made up of two encoders, one for images and one for text (Mokady et al., 2021; Radford et al., 2021). The image encoder is language-agnostic, whereas the text encoder is not.

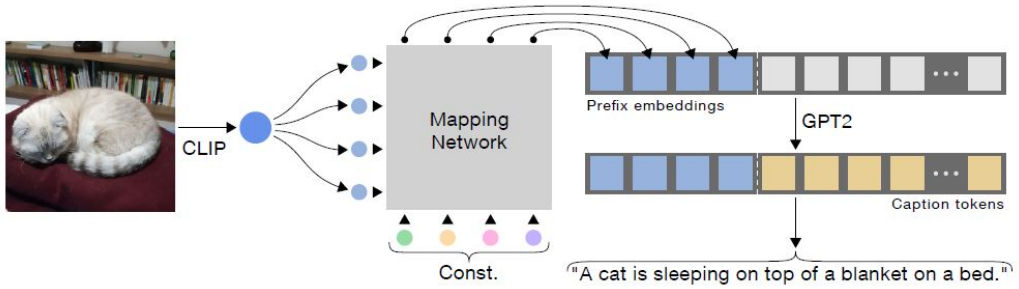


Figure 4.3: Overview of ClipCap prefix-tuning (Mokady et al., 2021).

For the purpose of this prefix-tuning task only the image encoder was necessary. Nevertheless, a multilingual version of CLIP was preferred, because the ensuing evaluation stage required the use of a version of CLIP compatible with the Italian language, and using the same model for both stages of the pipeline saved on memory resources. The model chosen was "sentence-transformers/clip-ViT-B-32-multilingual-v1" [4], which encodes over 50 languages, including Italian (Sentence-Transformers, 2021).

While the multilingual version of CLIP was chosen for convenience, a decoder language model capable of producing output in Italian was mandatory. A suitable monolingual model proved elusive, so another multilingual model was selected, having care to pick one that included Italian among its languages. The choice landed on "ai-forever/mGPT" [5]. This model was trained on 61 languages, including Italian, with data from Wikipedia and the C4 Corpus (Shliazhko et al., 2022).

Figure 4.3 shows an overview of the combined model architecture and data flow. The embedding generated by CLIP is passed through this mapping network, which turns it into a fixed length prefix. This is then passed to mGPT, which begins the generation conditioned on this prefix (Mokady et al., 2021).

The use of two separate models combined into one architecture, including a decoder which contains 1.3 billion parameters (Shliazhko et al., 2022), would normally require lengthy fine-tuning. However, the chosen strategy offers a faster solution. Instead of fine-tuning the whole models, their pa-

[4] `https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1`

[5] `https://huggingface.co/ai-forever/mGPT`

Model	ChrF++	CLIPScore	RefCLIPScore	BERTScore (Baseline)	BERTScore (Adapted)
Prefix-Tuned	59.116	0.245	0.388	0.715	0.711

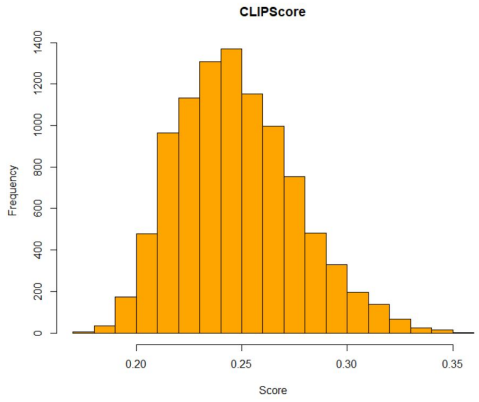
Table 4.3: Results of the Italian model in terms of five metrics.

rameters are frozen, and only the parameters of the small mapping network added in between are adjusted during the fine-tuning process. This has the additional effect of lowering data requirements, making adaptation possible even with a dataset of moderate size (Mokady et al., 2021), which makes this strategy ideal for this project.

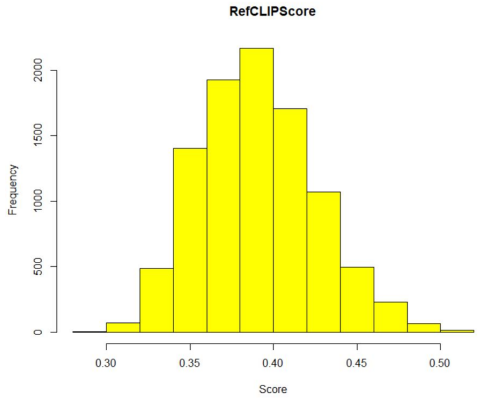
Even with the majority of the parameters frozen and thus not updated during the fine-tuning process, the forward passes through the model still require extensive computation, making mixed precision [6] a necessity to reduce the time required to a more reasonable amount (Micikevicius et al., 2017). Thus, Mixed Precision was added to the PyTorch loop implementation. The learning rate selected was 5×10^{-5}, and the training was set to run for 5 epochs. Early stopping and patience 2 were implemented to avoid using unnecessary resources on a computationally expensive process.

The fine-tuning for this model ran for the full 5 epochs, reporting improvement at every epoch. It could have potentially been trained further, but time constraints prevented it. Table 4.3 reports the results obtained after these 5 epochs, including BERTScore calculated with an adapted model whose fine-tuning is described in Section 4.3.

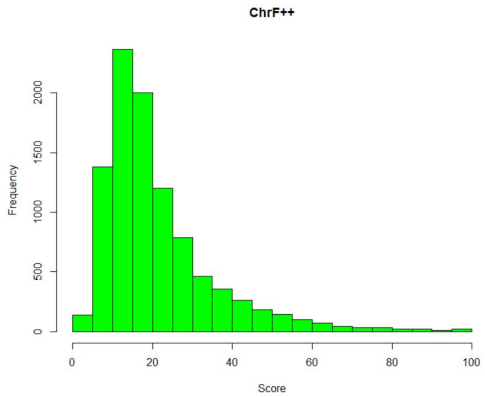
[6] While training a model, mixed precision is used to store floating point numerical information using more or less memory. Using higher precision occupies more memory and generally leads to slower computation, but more precise and stable calculations. Conversely, using lower precision has the opposite effect. However, depending on the specific operations performed, the loss of quality caused by using lower precision can sometimes be minimal or irrelevant. Consequently, there are times, such as during the forward pass through the model, when an acceptably small loss in numerical accuracy can bring about a major reduction in computing time. On the other hand, precision is fundamental during certain operations, such as when updating the model weights. In this case, using lower precision could lead to undesirable results. Thus, mixed precision allows the reduction of training time by using lower precision where possible, while still ensuring the model's quality by using higher precision where numerical accuracy is critical. (Micikevicius et al., 2017).



(a) CLIPScore Shapiro-Wilk Test: W=0.9868 p-value=7.97e-8



(b) RefCLIPScore Shapiro-Wilk Test: W=0.9915 p-value=1.631e-5

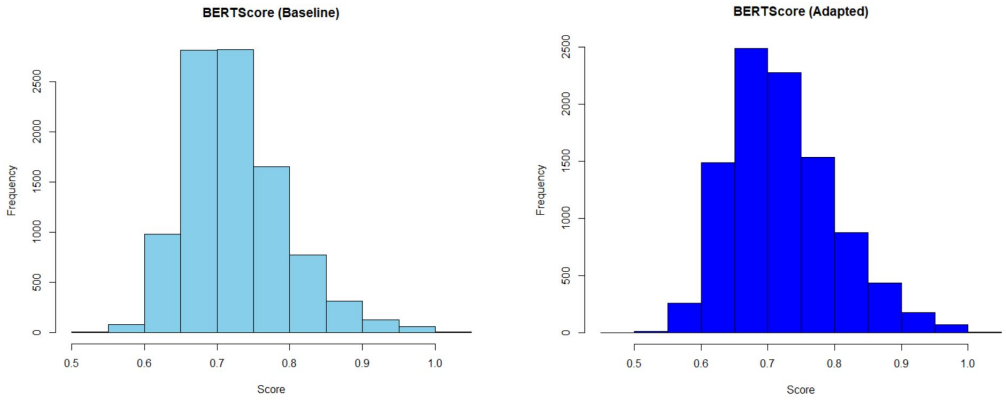


(c) ChrF++ Shapiro-Wilk Test: W=0.7842 p-value<2.2e-16

Figure 4.4: Histograms of the distributions of the scores (CLIPScore, RefCLIPScore, ChrF++) of the prefix-tuned model (Italian).

As in Section 4.1, the aggregated results for all metrics except ChrF++ are reported using the median value, as Shapiro-Wilk tests performed on a sample of 1000 instances revealed the distributions significantly differ from a normal distribution. The results of these tests are reported in Figures 4.4 and 4.5, below the histogram of the respective score distribution. Again, the test on the ChrF++ scores was included for completeness.

Having no baseline against which to compare these results makes them



(a) BERTScore (Baseline)
Shapiro-Wilk Test:
W=0.9658, p-value=1.397e-14

(b) BERTScore (Adapted)
Shapiro-Wilk Test:
W=0.9754, p-value=5.558e-12

Figure 4.5: Histograms of the distributions of the scores (Baseline and Adapted BERTScore) of the prefix-tuned model (Italian).

difficult to interpret, although they appear in line with the results obtained by the fine-tuned English model. However, it should be noted that the test sets used in the two experiments are different, which means the results of the two models are not comparable. Consequently, the only thing that can be inferred by looking at their results side by side is that the values reported in Table 4.3 are probably reasonable and no obvious methodological error was committed during the training procedure.

CLIPScore and RefCLIPScore report a low similarity between the input image and the generated caption in this case as well, as they did for the previously analyzed image-captioning models. On the other hand, BERTScore's higher similarity suggests the previous two metrics may once again not be very reliable. The BERTScore calculated with the adapted BERT model is just shy of that obtained with the baseline model, suggesting an increased sensitivity to the specifics of the domain.

The distributions of the results obtained by the various metrics, shown in Figures 4.4 and 4.5 are similar to those in Section 4.1. ChrF++ shows a similar skew towards the lower end of the range, while at the same time occasionally reaching the highest possible scores. Both versions of BERTScore also reach the higher parts of the possible range, whereas CLIPScore and

RefCLIPScore do not.

The difficulty of interpreting these scores makes performing a qualitative analysis even more important for this model. Table 4.4 reports some examples for this purpose. As before, it is not a representative sample.

Example 1 in the table shows the model can find it hard to tell some ingredients apart. The picture may not be the clearest in this case, or maybe the absence of the numbering in the lower right corner could also contribute to the model's confusion, even though the dataset contains multiple unnumbered examples. Regardless of the reason, the output is poor both grammatically and semantically. The adapted BERT model seems to be more adept at recognizing these issues than the baseline model, whereas the other metrics do not adequately capture and reflect the problem.

Example 2 further supports the theory that the absence or differing style of the numbering in the lower right corner is causing the model trouble. In this case, a not very visible number is present, yet the model failed to detect it and replaced it with an incorrect guess. The rest of the image is cryptic to interpret for a human as well, and so is the reference, making it difficult to adequately assess the suitability of the hypothesis caption. Looking at the picture, it could be supposed the preparation is some kind of dessert, which makes butter a plausible ingredient. Checking the original recipe revealed the addition to be a previously prepared mixture of various ingredients, including cocoa, salt and baking powder (Giallo Zafferano, 2026). In other words, the model could have never made a perfect guess with the information it had available, and it could have certainly generated worse captions.

Example 3 in table Table 4.4 is very good, despite once again lacking the numbering. The hypothesis caption and the reference are semantically well matched, and both ChrF++ and BERTScore reflect this in their scoring. On the other hand CLIPScore and RefCLIPScore confirm their unreliability on this dataset by rating this caption the lowest among these examples despite it being unquestionably the best so far.

Example 4 contains an attempt by the model to produce a more articulate caption. In the original recipe the ingredient was onion rather than shallot, yet the difference between the two vegetables can be considered minor if not irrelevant, as the latter is a type of onion. Visually, it is virtually indistinguishable from the ingredient shown in the image, which makes the guess surprisingly accurate, and possibly better than some humans may do when given the same context-scarce information.

<table>
<tr><th></th><th>Input</th><th>Results</th><th></th></tr>
<tr><td>1</td><td>  </td><td>Hypothesis: Per preparare il risotto con le zucchine e le zucchine, iniziate a pulire le zucchine <1>
Reference: Mondate i porri, tagliate la base, eliminate le foglie verdi e il primo strato di buccia <3>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BSb</th><th>BSa</th></tr><tr><td>17.712</td><td>0.290</td><td>0.432</td><td>0.701</td><td>0.666</td></tr></table></td><td>×</td></tr>
<tr><td>2</td><td>  </td><td>Hypothesis: e il burro fuso <7>
Reference: e aggiungetene metà al composto precedentemente preparato <5>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BSb</th><th>BSa</th></tr><tr><td>7.236</td><td>0.225</td><td>0.363</td><td>0.726</td><td>0.750</td></tr></table></td><td>×</td></tr>
<tr><td>3</td><td>  </td><td>Hypothesis: Unite i pinoli <8>
Reference: Aggiungete anche i pinoli <19>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BSb</th><th>BSa</th></tr><tr><td>36.325</td><td>0.198</td><td>0.328</td><td>0.875</td><td>0.874</td></tr></table></td><td>✓</td></tr>
<tr><td>4</td><td>  </td><td>Hypothesis: In una padella capiente fate rosolare lo scalogno <5>
Reference: e fatela rosolare in una padella antiaderente con un filo di olio <5>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BSb</th><th>BSa</th></tr><tr><td>40.540</td><td>0.256</td><td>0.405</td><td>0.813</td><td>0.804</td></tr></table></td><td>✓</td></tr>
<tr><td>5</td><td>  </td><td>Hypothesis: e la scorza grattugiata di un limone <2>
Reference: e aromatizzate con mezza scorza di limone, avendo cura di grattugiare solo la parte colorata <2>
<table><tr><th>CF</th><th>CS</th><th>RCS</th><th>BSb</th><th>BSa</th></tr><tr><td>25.596</td><td>0.256</td><td>0.406</td><td>0.757</td><td>0.750</td></tr></table></td><td>✓</td></tr>
</table>

Table 4.4: Qualitative results of the prefix-tuned Italian model. For readability the names of the metrics were shortened to CF for ChrF++, CS for CLIPScore, RCS for RefCLIPScore, BSb for BERTScore (Baseline), and BSa for BERTScore (Adapted). The mark on the right represents whether the caption should be considered as correct.

Example 5 in Table 4.4 contains an excellent hypothesis caption that is more direct and less noisy than the reference. Again, the model does not consider any further context, not even details a human with in-domain knowledge could have inferred from the image alone, as in this case. Unexpectedly, the evaluation with the adapted BERT assigns the same score to this instance and to the second example, whereas baseline BERT allows to discriminate their quality slightly better. Without an accurate manual evaluation it is, however, impossible to assess whether the difference between the two versions of BERT is in any way significant or merely a coincidence.

These examples, as well as those discussed in Section 4.1, show that results provided by both CLIPScore and RefCLIPScore have to be discarded as too unreliable to be of much significance. Consequently, Chrf++ turns out to be the second best metric among those employed, despite its reliance on the surface form of the sentence over its meaning limiting its ability to properly evaluate any caption that fully paraphrases the reference. At the same time, its use of character n-grams allows it to capture in-domain terminology that may not have a possible synonym, such as *limone, padella, rosolare*. This applies even when the term is used in a slightly different form, such as in the case of *grattugiare* and *grattugiate*. In other words, ChrF++ appears capable of recognizing some high quality captions, but may penalize many others that are just as good. Unfortunately, the distributions of its scores tend to skew towards the lower end of the range, as shown in Figures 4.2 and 4.4, making its usefulness limited.

4.3 BERT Fine-Tuning

Given the nature of the dataset used for this project, the consideration made in Section 2.5 about BERT sometimes struggling to deal with specialized domains likely applies, so the model "dbmdz/bert-base-italian-uncased" [7] was selected for finetuning. The model was pretrained on Wikipedia and texts from the OPUS corpora collection, which means it has already seen some in-domain data thanks to Wikipedia gastronomy pages. As a consequence even baseline performance could be acceptable, although fine-tuning is still likely to improve it.

The model was fine-tuned using the same masked language modeling tech-

[7] `https://huggingface.co/dbmdz/bert-base-italian-uncased`

Model	K=1	K=5
Baseline BERT	60.95%	76.82%
Adapted BERT	**73.18%**	**82.74%**

Table 4.5: Comparison between the performance of the baseline and adapted versions of BERT.

nique that was employed in its pretraining, which is described in Section 2.5. This training was implemented through the use of Hugging Face's Trainer class, which allowed to control more hyperparameters than during the other experiments without the job exceeding the timeframe available. The model was set to train for 15 epochs with a learning rate of 5×10^{-5} and a linear warmup rate of 0.15, providing a dynamic learning rate schedule to increase the stability of the process (Kalra and Barkeshli, 2024). A 0.01 weight decay was further used to help the model learn to generalize better, while mixed precision (Micikevicius et al., 2017) and early stopping with patience 2 were implemented to make the training more efficient. Fine-tuning ran for 11 epochs, with the best validation result reported at the end of epoch 9.

The baseline and adapted models were then tested using top-k sampling, that is to say, it was verified whether the original token was among the model's top-k choices for the masked token replacement (Gogo-Job, 2025). Table 4.5 reports the success percentages of the models, that is to say the percentage of times the original token was present among the top-k selections. This allows to compare the results between before and after the fine-tuning.

Despite the model having already seen data from the gastronomy domain during pretraining, it is immediately obvious the fine-tuning was effective and considerably improved its performance for both values of k.

This means the evaluations in Section 4.2 that were performed using this adapted model may be more reliable than those performed using the baseline model.

Chapter 5

Final Remarks

This project used a multilingual and multimodal dataset of recipes to fine-tune two different image-captioning models to generate the cooking instructions associated with an input picture. The selected languages for the experiments were English and Italian. The English data was used to adapt a pretrained encoder-decoder image-captioning model to the desired style of captions, whereas the Italian data was used to perform prefix-tuning as a way to bridge the embedding spaces of CLIP and mGPT, thus leveraging their pretraining.

The linguistic style and non-standard purpose of this task make it difficult to rigorously assess the models' performance, though results appear promising. Both models show the ability to correctly recognize domain-specific tools and ingredients, sometimes from difficult to interpret visual information, though errors can occasionally still occur. Their use of domain-specific terminology is also appropriate. Where the context of the input picture is insufficient to determine the correct instruction, the model has been shown to output a reasonable guess based on the information it has available.

The quality of the results is particularly evident in the case of the English adapted model, thanks to the possibility of comparing them against those generated by the baseline model on the same testing set. The adapted model showed an overall improvement in performance, and a qualitative analysis of the data confirmed the style of the captions had changed to match the desired style. Indeed, the generated captions of the adapted model are cooking instructions, whereas the baseline model tended to ouput a description of the picture. A similar comparison was not possible for the Italian model, although once again a qualitative analysis of the results showed the desired

style of the captions was successfully achieved by this model as well.

Overall, BERTScore is clearly the most reliable automatic metric among the ones used, with the adapted version of the model being slightly stricter in its judgments than the baseline, although the difference between the two is minimal. BERTScore's main limitation is that it does not consider the input picture at all, and instead relies solely on the comparison between the hypothesis and the reference text.

A manual evaluation would have provided a more accurate picture of the results. Unfortunately, the time-intensive nature of such an endeavor meant that performing it was not possible in the time available for this project.

Given the difficulties encountered when measuring the quality of the models with other metrics, even BERTScore should probably be subjected to some tests to fully ascertain its reliability. Checking how it correlates to human judgment when used to evaluate this kind of non-standard application of image captioning could help ensure the reliability of any results obtained in further experiments using this metric.

Reducing the noise of the dataset could also help improve the model's performance, and could also make the evaluation easier. Longer captions in particular may be noisy. A more in-depth examination of these may be warranted if further experiments are conducted; if the tendency towards noise is confirmed, removing long references from the dataset may improve the quality of the data, and thus likely that of the final model.

However, this does not help when the issue is the lack of enough context to correctly interpret the input image. A possible way to solve this issue and turn some of the noise into useful content could be creating a model that takes the flow of the whole recipe into account, and considers multiple steps and their order before generating appropriate instructions. This would likely require a model that is either recurrent or capable of considering multiple input pictures at once. Given that recipes can frequently contain 20 or 30 steps with as many pictures, a trade-off between enough context and computational expense may need to be found.

The topic could also be expanded in a different direction. For example, testing whether it is possible to generate pictures to illustrate the various steps of a recipe given its text. However, using automatic evaluation becomes even more difficult if pictures are the expected output, because the best automatic metrics found so far were those that compared the generated text against the reference text and ignored the picture entirely.

While this work has shown the viability of this approach to perform this

non-standard image-captioning task, alternative evaluation strategies may need to be explored if further experiments are conducted, and a manual evaluation should probably be performed despite the time and effort it requires.

Bibliography

Alireza Aghabagherloo, Aydin Abadi, Sumanta Sarkar, Vishnu Asutosh Dasu, and Bart Preneel. Impact of Data Duplication on Deep Neural Network-Based Image Classifiers: Robust vs. Standard Models. *arXiv preprint*, arXiv:2504.00638v2, 2025. doi: 10.48550/arXiv.2504.00638. Computer Science — Machine Learning.

Jay Alammar. The Illustrated Transformer. Blog post, 2018. URL `https://jalammar.github.io/illustrated-transformer/`. Accessed: 2026-01-29.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, and Antoine Miech. Tackling multiple tasks with a single visual language model. DeepMind Blog, 2022a. URL `https://deepmind.google/discover/blog/tackling-multiple-tasks-with-a-single-visual-language-model/`. Accessed: 2026-01-29.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a Visual Language Model for Few-Shot Learning. *arXiv preprint*, arXiv:2204.14198, 2022b. doi: 10.48550/arXiv.2204.14198.

Why Amit. Fine-Tuning BERT: A Practical Guide. Medium, Dec 2024. URL `https://medium.com/@whyamit101/fine-tuning-bert-a-practical-guide-b5c94efb3d4d`. Accessed: 2026-02-01.

Skylar Jean Callis. Vision Transformers, Explained: A Full Walk-Through of Vision Transformers in PyTorch. Medium (TDS Archive), Feb 2024. URL https://medium.com/data-science/vision-transformers-explained-a9d07147e4c8. Accessed: 2026-01-29.

ConfigrTechnologies. Multimodal AI: The Next Frontier in Artificial Intelligence. Medium (DeepDive AI), Mar 2024. URL https://medium.com/deepdive-ai/multimodal-ai-b34393d62a1c. Accessed: 2026-01-28.

Julianna Delua. Supervised vs. Unsupervised Learning: What's the Difference? IBM Think, 2025. URL https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning. Accessed: 2026-01-28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint*, arXiv:1810.04805, 2018. doi: 10.48550/arXiv.1810.04805. Computer Science — Computation and Language.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale. *arXiv preprint*, arXiv:2010.11929, 2020. doi: 10.48550/arXiv.2010.11929.

Daniel Fleisch. *Guida allo studio dei vettori e dei tensori*. Editori Riuniti University press, 2018.

Giallo Zafferano. Giallo Zafferano — Italian Recipes. Website, 2026. URL https://www.giallozafferano.it/. Accessed: 2026-01-30.

Divija Godse. Message Digest Algorithm (MD5) and Secure Hash Algorithm (SHA). Medium, May 2023. URL https://medium.com/@divijagodse/message-digest-algorithm-md5-and-secure-hash-algorithm-sha-70cde74fd883. Accessed: 2026-02-01.

Egop Gogo-Job. Prompt Engineering Explained: Understanding Top-K, Top-P, Temperature, and Advanced Techniques. Medium, Jan 2025. URL https://medium.com/@egopgogojob/prompt-engineering-explained-understanding-top-k-top-p-temperature-and-advanced-techniques-b7ae7fa49fda. Accessed: 2026-02-02.

Google AI Blog. Introducing Gemini: our largest and most capable AI model. Google AI Blog, 2023. URL `https://blog.google/innovation-and-ai/technology/ai/google-gemini-ai/`. Accessed: 2026-01-29.

Nicolò Grando. Multimodal AI and Contextual Intelligence: Revolutionizing Human-Machine Interaction. Medium, 2025. URL `https://medium.com/@nicolo.g88/multimodal-ai-and-contextual-intelligence-revolutionizing-human-machine-interaction-ae80e6a89635`. Accessed: 2026-01-28.

Mehedee Hassan. How the Deep Past Challenge Scores Your Translations. Medium, Dec 2025. URL `https://mhr007.medium.com/how-the-deep-past-challenge-scores-your-translations-0050c2c55d59`. Accessed: 2026-01-28.

Sen He, Wentong Liao, Hamed R. Tavakoli, Michael Yang, Bodo Rosenhahn, and Nicolas Pugeault. Image Captioning through Image Transformer. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 153–169, Kyoto, Japan, November 2020. Springer. doi: 10.1007/978-3-030-69538-5_10.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A Reference-free Evaluation Metric for Image Captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, Online and Punta Cana, Dominican Republic, Nov 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.595.

Gunal Hincal. Natural Language Processing (What is NLP?). Medium, Jul 2023. URL `https://medium.com/@hincalgunal/natural-language-processing-what-is-nlp-c9b86cacc032`. Accessed: 2026-01-28.

C. Hutto and Eric Gilbert. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225, 2014. doi: 10.1609/icwsm.v8i1.14550.

Dayal Singh Kalra and Maissam Barkeshli. Why Warmup the Learning Rate? Underlying Mechanisms and Improvements. *arXiv preprint*,

arXiv:2406.09405, 2024. doi: 10.48550/arXiv.2406.09405. Computer Science — Machine Learning.

Karan Kamat. Understanding Mean, Harmonic Mean, and Geometric Mean: A Comprehensive Guide. Medium, Jul 2023. URL https://medium.com/@karan.kamat1406/understanding-mean-harmonic-mean-and-geometric-mean-a-comprehensive-guide-f61e1310feaa. Accessed: 2026-01-28.

Lauren Klein and Catherine D'Ignazio. Data Feminism for AI. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, Rio de Janeiro, Brazil, 2024. Association for Computing Machinery. doi: 10.1145/3630106.3658543.

Ria Kulshrestha. Transformers. Medium (TDS Archive), Jun 2020. URL https://medium.com/data-science/transformers-89034557de14. Accessed: 2026-01-29.

Hobson Lane and Maria Dyshel. *Natural Language Processing in Action, Second Edition*. Manning Publications, 2025.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577.

Preeyanuch Leelapisuth. A Beginner's Guide to Model Evaluation in Machine Learning. Medium, May 2025. URL https://mookpreeyanuch.medium.com/a-beginners-guide-to-model-evaluation-in-machine-learning-6e4ad4a844b2. Accessed: 2026-02-02.

Omid (Saeid) Masoumzadeh. From Rule-Based Systems to Transformers: A Journey through the Evolution of Natural Language Processing. Medium, Jun 2023. URL https://medium.com/@masoumzadeh/from-rule-based-systems-to-transformers-a-journey-through-the-evolution-of-natural-language-9131915e06e1. Accessed: 2026-01-28.

Ornela Megne. Image Captioning. Medium, Jun 2023. URL `https://medium.com/@pmegne/image-captioning-5162e22ef2ac`. Accessed: 2026-01-28.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed Precision Training. *arXiv preprint*, arXiv:1710.03740, 2017. doi: 10.48550/arXiv.1710.03740. Published as conference paper at ICLR 2018.

Prabhaker Mishra, Chandra M. Pandey, Uttam Singh, Anshul Gupta, Chinmoy Sahu, and Amit Keshri. Descriptive Statistics and Normality Tests for Statistical Data. *Annals of Cardiac Anaesthesia*, 22(1):67–72, Jan-Mar 2019. doi: 10.4103/aca.ACA_157_18.

Ron Mokady, Amir Hertz, and Amit H. Bermano. ClipCap: CLIP Prefix for Image Captioning. *arXiv preprint*, arXiv:2111.09734, 2021. doi: 10.48550/arXiv.2111.09734.

Long Nguyen. Large Language Models & Transformer Architecture: The Basics. Medium, Jul 2023. URL `https://medium.com/@thisislong/large-language-models-transformer-architecture-the-basics-2bdd84a6db17`. Accessed: 2026-01-29.

Joakim Nivre. On Statistical Methods in Natural Language Processing. In *Proceedings of the 13th Nordic Conference of Computational Linguistics (NoDaLiDa 2001)*, Uppsala, Sweden, 2001. Department of Linguistics, Uppsala University. URL `https://aclanthology.org/W01-1720/`.

D.Q. Nykamp. The dot product. From Math Insight, 2026. URL `https://mathinsight.org/dot_product`. Accessed: 2026-02-05.

Mariah Ore. English dominance in natural language processing. Medium, Dec 2022. URL `https://medium.com/@mariah.ore/english-dominance-in-natural-language-processing-412091a7fdb5`. Accessed: 2026-02-12.

Asmaa A. E. Osman, Mohamed A. Wahby Shalaby, Mona M. Soliman, and Khaled M. Elsayed. Novel concept-based image captioning models using LSTM and multi-encoder transformer architecture. *Scientific Reports*, 14

(20762), 2024. doi: 10.1038/s41598-024-69664-1. Article number: 20762; Published: 05 September 2024.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135.

Flammie A. Pirinen. Introduction. In Arvi Hurskainen, Kimmo Koskenniemi, and Tommi Pirinen, editors, *Rule-Based Language Technology*, number 2 in NEALT Monograph Series, pages 1–8. Northern European Association for Language Technology, Tartu, 2023. Accessed via ResearchGate PDF.

Maja Popović. ChrF: Character n-gram F-score for Automatic MT Evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049.

Maja Popović. ChrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4770.

Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023. URL `http://udlbook.com`.

Pruthvi. Transformers Explained — The Model That Changed Everything. Medium, Jun 2025. URL `https://medium.com/@jvpnath/transformers-explained-the-model-that-changed-everything-5ce938c2b6c1`. Accessed: 2026-02-01.

Raj Pulapakura. Multimodal Models and Fusion – A Complete Guide. Medium, Feb 2024. URL `https://medium.com/@raj.pulapakura/multimodal-models-and-fusion-a-complete-guide-225ca91f6861`. Accessed: 2026-01-29.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv preprint*,

arXiv:2103.00020, 2021. doi: 10.48550/arXiv.2103.00020. Computer Science — Computer Vision and Pattern Recognition; Machine Learning.

Fahim Rustamy. CLIP Model and The Importance of Multimodal Embeddings. Medium (TDS Archive), Dec 2023. URL `https://medium.com/data-science/clip-model-and-the-importance-of-multimodal-embeddings-1c8f6b13bf72`. Accessed: 2026-01-28.

Mehrdad Sabetzadeh and Chetan Arora. Practical Guidelines for the Selection and Evaluation of NLP Techniques in RE. *arXiv preprint*, arXiv:2401.01508v1, 2024. URL `https://arxiv.org/abs/2401.01508v1`. Computer Science — Software Engineering.

Jeeva Saravanan. How to Evaluate Your Machine Learning Model. Medium (GenAI.io), May 2021. URL `https://medium.com/genai-io/how-to-evaluate-your-machine-learning-model-76a7671e9f2e`. Accessed: 2026-02-04.

Sentence-Transformers. clip-ViT-B-32-multilingual-v1. Hugging Face Model Card, 2021. URL `https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1`. Accessed: 2026-02-05.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Vladislav Mikhailov, Anastasia Kozlova, and Tatiana Shavrina. mGPT: Few-Shot Learners Go Multilingual. *arXiv preprint*, arXiv:2204.07580, 2022. URL `https://arxiv.org/abs/2204.07580`. Computer Science – Computation and Language, accessed: 2026-02-05.

Aarohi Srivastava and David Chiang. We're Calling an Intervention: Taking a Closer Look at Language Model Adaptation to Different Types of Linguistic Variation. *arXiv preprint*, arXiv:2404.07304v1, 2024. doi: 10.48550/arXiv.2404.07304. Computer Science – Computation and Language.

Cole Stryker and Jim Holdsworth. What Is Natural Language Processing? IBM Think, 2025. URL `https://www.ibm.com/think/topics/natural-language-processing`. Accessed: 2026-01-28.

Sathya Krishnan Suresh. Image Caption Model from scratch — ViT+GPT. Medium, Jun 2024. URL `https://medium.com/@mr.sk12112002/`

image-caption-model-from-scratch-vit-gpt-94afaae30fb7. Accessed: 2026-01-28.

Rachael Tatman. Evaluating Text Output in NLP: BLEU at Your Own Risk. Medium (TDS Archive), Jan 2019. URL https://medium.com/data-science/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213. Accessed: 2026-01-28.

The MathWorks, Inc. Decision Trees — Statistics and Machine Learning Toolbox Documentation. Online documentation, 2026. URL https://uk.mathworks.com/help/stats/decision-trees.html. Accessed: 2026-02-02.

The Sage. A Powerful Native Multimodal LLM. Say Hello to Meta's Chameleon. Medium, Jun 2024. URL https://medium.com/@TheSage/a-powerful-native-multimodal-llm-say-hello-to-metas-chameleon-7255e949162f. Accessed: 2026-01-29.

Manogna Vankayalapati. A Comprehensive Guide to Supervised and Unsupervised Learning Algorithms and Their Applications. Medium (Tech Writings), Feb 2024. URL https://medium.com/tech-writings/a-comprehensive-guide-to-supervised-and-unsupervised-learning-algorithms-and-their-applications-ea0f619d6f0d. Accessed: 2026-01-28.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint*, arXiv:1706.03762, 2017. doi: 10.48550/arXiv.1706.03762. Computer Science — Computation and Language; Machine Learning.

Vered Volansky, Noam Ordan, and Shuly Wintner. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118, 2015. doi: 10.1093/llc/fqt031.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. *arXiv preprint*, arXiv:1904.09675, 2019. doi: 10.48550/arXiv.1904.09675. Computer Science — Computation and Language.

Appendix

Dot Product and Cosine Similarity

The dot product compares two vectors of the same cardinality (Lane and Dyshel, 2025), and reveals at which extent one is pointing in the same direction as the other (Nykamp, 2026).

The result of a dot product is always a scalar (Fleisch, 2018; Lane and Dyshel, 2025; Nykamp, 2026), and its computation is commutative (Fleisch, 2018).

If the Cartesian coordinates of the vectors are known, as is typically the case in the NLP field, the dot product can be calculated as follows:

$$\mathbf{A} \cdot \mathbf{B} = A_1 B_1 + A_2 B_2 + ... + A_n B_n$$

The above formula is generalized to any dimension n, and can also be written as:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i$$

The dot product represents the projection of $\mathbf{A}$ in the direction of $\mathbf{B}$ multiplied by the magnitude of $\mathbf{B}$ (Fleisch, 2018). As the projection of $\mathbf{A}$ in the direction of $\mathbf{B}$ can be written as $|\mathbf{A}| \cos\theta$ (Fleisch, 2018), where θ is the angle between the vectors, the dot product can also be written as:

$$\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}|\,|\mathbf{B}| \cos\theta$$

where $|\mathbf{A}|$ and $|\mathbf{B}|$ represent the magnitudes (lengths) of the vectors.

The dot product can be used as a proxy for similarity, although it yields higher values when the vectors have a larger magnitude. A better similarity

result can be obtained by normalizing both sides of the equation for length, that is, by dividing both sides by the product of the magnitudes of $\mathbf{A}$ and $\mathbf{B}$.

$$\frac{|\mathbf{A}|\,|\mathbf{B}|\cos\theta}{|\mathbf{A}|\,|\mathbf{B}|} = \frac{\mathbf{A}\cdot\mathbf{B}}{|\mathbf{A}|\,|\mathbf{B}|}$$

This becomes the operation called cosine similarity, which calculates the cosine of the angle between the two vectors (Lane and Dyshel, 2025):

$$\cos\theta = \frac{\mathbf{A}\cdot\mathbf{B}}{|\mathbf{A}|\,|\mathbf{B}|}$$

This limits the range of the resulting scalar to the interval $[-1, 1]$, with a value of -1 representing opposite vectors, a value of 0 representing perpendicular vectors, and a value of 1 indicating that the normalized vectors are identical (Lane and Dyshel, 2025).