POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATION

Institute of Computing Science

Bachelor's thesis

VISUAL WORD SENSE DISAMBIGUATION - PREDICTING SENSES OF WORDS BASED ON IMAGES

Sofya Aksenyuk, 150284

Uladzimir Ivashka, 150281

Oleksandr Yasinskyi, 150570

Supervisor

dr inż. Dawid Wiśniewski

POZNAŃ 2024

Contents

Chapter 1

Abstract

1.1 English

This thesis explores Visual Word Sense Disambiguation (Visual-WSD) in Natural Language Processing, focusing on the integration of visual contexts to address language ambiguities like polysemy and homonymy. An evaluation framework is introduced, assessing the capabilities of existing models in interpreting complex visual-textual data. The research concludes the superior performance of the CLIP model over other multimodal models in Visual-WSD tasks. Through fine-tuning, a model performance improvement is achieved in addressing the issue. This research delves into the architectural nuances of models, investigating the reasons behind their varying levels of performance for the Visual-WSD task. The study sets a foundation for future research in the field.

1.2 Polish

W niniejszej pracy badany jest problem ujednoznaczniania sensu słów w kontekście wizualnym (Visual-WSD) w ramach przetwarzania języka naturalnego. Skupiono się na integracji kontekstów wizualnych, aby rozwiązać wyzwania językowe, takie jak polisemia i homonimia. Wprowadzony został framework, który pozwala na ocenę zdolności współczesnych modeli do interpretacji złożonych danych wizualno-tekstowych. Stwierdzono, że model CLIP wykazuje lepszą wydajność w zakresie Visual-WSD w porównaniu z innymi modelami multimodalnymi. Poprzez fine-tuning uzyskano poprawę dokładności modelu. Dodatkowo, dokonano analizy architektonicznych niuansów modeli, aby wyjaśnić przyczyny ich różnych poziomów wydajności w analizowanym problemie. Uzyskane wyniki stanowią fundament dla dalszych badań w tej dziedzinie.

Chapter 2

Introduction

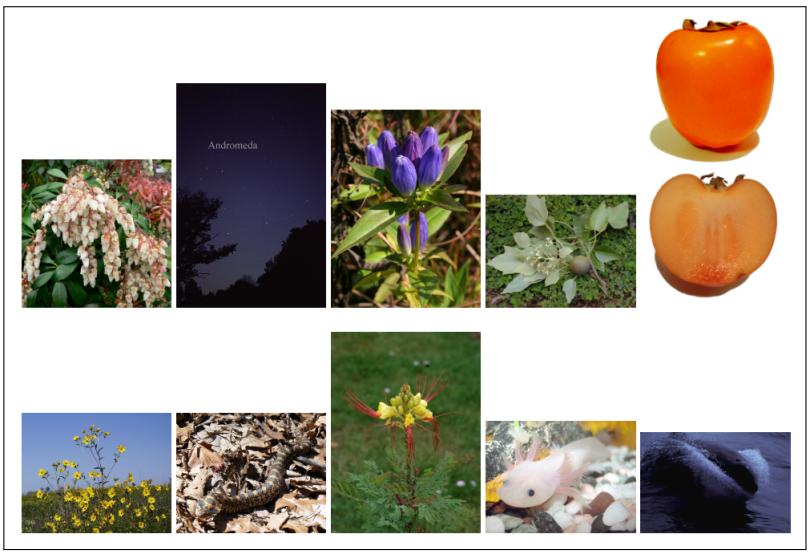
2.1 Project summary

Natural Language Processing (NLP) can be characterized as a set of techniques and methodologies within the field of Artificial Intelligence (AI) that enable computers to understand human language in spoken and written forms. Despite significant advancements, the field of NLP still faces challenges, particularly in dealing with language's inherent ambiguity.

Language is a tricky tool, full of words that might mean different things given different contexts (polysemy) as well as words that sound alike but mean different things (homonymy). Although humans usually rely on context to navigate such ambiguities, machines continue to find it complicated to grasp such language specifications.

Visual Word Sense Disambiguation (Visual-WSD) is an area that extends beyond text to include visual contexts. Visual-WSD concerns the intersection of visual and textual comprehension, offering a more comprehensive approach to interpreting a language. It's objective is to identify the correct image from a set of options that best represents the meaning of a specific word within a given textual context.

Figure 2.1
Example of Visual-WSD Challenge



To illustrate the fundamental challenge of Visual-WSD, let's examine the following example depicted on **Figure 2.1**[11]: Given the phrase *andromeda tree* where ***andromeda*** is the ambiguous term, and presented with ten possible images, the objective is to select the image that correctly represents the meaning of andromeda. In this case, the correct image is the first one on the left.

This thesis sets out to investigate the effectiveness of Visual-WSD within the realm of NLP. To approach the problem of Visual-WSD, a concept of Deep Learning (DL) model's multimodality has to be introduced. Multimodality in Visual-WSD area refers to the integration of multiple types of data: textual and visual. Based on this, this work will focus on models of that type.

2.2 Project objective

The project objective results in exploratory research work. It involves an in-depth look at current models for the Visual-WSD problem as well as their applicability in real-world scenarios. As a result, this thesis aims to establish an experimental evaluation framework that can be employed to assess the performance, ease of use, and computational efficiency of various image-to-text models, both in the context of general image captioning tasks and from a Visual-WSD perspective. Additionally, the work intents to develop a refined model that is specifically designed to adeptly address the Visual-WSD problem, while keeping its initially targeted capabilities.

2.3 Work structure

The project was divided into 4 main parts, with each group member assigned responsibility for at least one part of the task. Nonetheless, certain steps were accomplished collaboratively through teamwork.

- **Data procurement** was prepared by **Uladzimir Ivashka**. The step involved locating suitable datasets aligned with the project's objectives, exploratory data analysis, and data preprocessing, ultimately providing the data used to test the models later on.
- **Model acquisition and evaluation:**
  - **Model acquisition** was prepared by **Uladzimir Ivashka**. This part consisted of collecting a list of the most relevant pre-trained models.
  - **Experimental evaluation framework** was prepared by **Sofya Aksenyuk**. It required the implementation of the architecture of the standardized evaluation framework.
  - **Model implementation** was prepared by **Sofya Aksenyuk**. The subtask included the implementation of models as an extension to **Experimental evaluation framework** upon which the research was founded.
  - **Metrics** was prepared by **Uladzimir Ivashka**. It contained research and selection of the most appropriate metrics for evaluating the models' performance and creation of the evaluation pipeline.
  - **Model evaluation** was prepared by **Sofya Aksenyuk**. This step encompasses a thorough evaluation of models on a dataset specifically created for the Visual Word Sense Disambiguation (Visual-WSD) challenge, along with a general baseline dataset used for image-captioning tasks.

- **Model comparison** was prepared by **Oleksandr Yasinskyi**. This point included detailed analysis of the examined models' architectures in order to determine the reason of an architecture leading to better performance for Visual-WSD problem, as well as analysis of the running times of each model, their ease of use, such as installation and provided features.
- **Model fine-tuning** was prepared by **Uladzimir Ivashka** and **Oleksandr Yasinskyi**. This part involved fine-tuning the best-performing model so far for Visual-WSD problem.

Chapter 3

Theoretical foundation

3.1 Introduction to Machine Learning and Deep Learning

3.1.1 Machine Learning: An Overview

Machine learning (ML), a cornerstone of artificial intelligence (AI), is a field of study of data analysis that automates analytical model building. It is a field based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention. The concept of ML is not new, but its relevance has soared in recent years due to the exponential increase in data generation and advancements in computing capabilities.

Machine learning algorithms are often categorized based on how they learn from data. Understanding these learning types is crucial for selecting the appropriate algorithm for a given problem.

- **Supervised Learning:** In supervised learning, the algorithm is trained on a labeled dataset, where the desired output is known. The goal is to learn a mapping from inputs to outputs, making predictions based on new, unseen data.
- **Unsupervised Learning:** Unsupervised learning involves training the model on data without labeled outcomes. The aim is to discover underlying patterns or structures in the data.
- **Semi-Supervised Learning:** This approach falls between supervised and unsupervised learning. It uses both labeled and unlabeled data for training.
- **Reinforcement Learning:** Reinforcement learning is about training models to make a sequence of decisions. The algorithm learns to achieve a goal in an uncertain, potentially complex environment.
- **Transfer Learning:** Transfer learning involves taking a pre-trained model and adapting it to a new, but related problem. It is especially popular in deep learning, where substantial computational resources are required to train large models.

3.1.2 Deep Learning: Delving Deeper

Deep learning (DL) is a specialized subset of ML that uses algorithms inspired by the structure and function of the brain, called artificial neural networks. Unlike some traditional ML algorithms, which are linear, DL models are known for their superior ability to process and interpret vast amounts of complex, unstructured data.

The *deep* in deep learning refers to the number of layers through which the data is transformed. More layers allow for more complex features to be extracted and learned. These layers are composed

of nodes, or *neurons*, each performing simple computations on the input data. The output of these computations is then passed on to the next layer. This hierarchical, layer-based structure enables deep learning models to recognize patterns and features at different levels of abstraction, making them incredibly effective for a wide range of applications, from image and speech recognition to natural language processing (NLP).

3.2 Fundamentals of Natural Language Processing (NLP)

3.2.1 Overview of NLP

NLP aims to facilitate interactions between computers and human (natural) language. It involves the development of algorithms and systems that can understand, interpret, generate, and respond to human language in a meaningful way. This is challenging due to the complexity, ambiguity, and varied structure of human language.

3.2.2 Key Concepts and Techniques in NLP

In its pursuit to bridge human communication and computer understanding, NLP involves several key concepts and techniques:

- **Tokenization and Text Normalization:** These are foundational steps in NLP where text is converted into tokens (smaller units like words) and normalized (standardized in terms of casing, punctuation, etc.).
- **Stemming and Lemmatization:** Both techniques reduce words to their base or root form, with lemmatization ensuring that the root word (lemma) belongs to the language.
- **Part-of-Speech Tagging and Chunking:** Identifying parts of speech and grouping different tokens into meaningful phrases or chunks.
- **Named Entity Recognition (NER) and Entity Linking:** Detecting named entities and linking them to entities in a knowledge base.
- **Syntax and Dependency Parsing:** Analyzing the grammatical structure of sentences and understanding the dependencies between words.
- **Semantic Analysis:** Understanding the meaning and interpretation of words and sentences in context.
- **Sentiment Analysis and Opinion Mining:** Extracting subjective information like opinions, emotions, and sentiments from text.
- **Coreference Resolution:** Identifying when different words refer to the same entity in a text.
- **Machine Translation:** Automatically translating text from one language to another.

3.2.3 Recent Advances in NLP

Recent advances in NLP have been driven by deep learning, particularly through the use of models like Transformers. These have led to significant improvements in tasks such as language modeling, machine translation, and question-answering. Transfer learning, where a model is pre-trained on

a large corpus and fine-tuned on specific tasks, has become a dominant approach. Despite its progress, NLP faces challenges like understanding context, handling idiomatic expressions, and managing biases in language models.

3.3 Transformers in NLP and Computer Vision

3.3.1 Introduction to Transformers

Transformers have revolutionized the field of NLP since their introduction in the seminal paper **Attention Is All You Need**[12] in 2017. Transformers use a mechanism called *attention* to weigh the significance of different parts of the input data. Prior to the transformer model, there were attempts to implement attention mechanisms in recurrent networks. First idea of attention was introduced in ***Neural Machine Translation by Jointly Learning to Align and Translate***[1]. However, a major limitation of these approaches was their continued reliance on recurrent connections. **Vaswani et al.**[12] demonstrated that these connections were unnecessary, enabling a significant increase in parallel processing capabilities during both training and evaluation phases. This allows for parallel processing of sequences and a much more efficient handling of dependencies, irrespective of their distance in the input data.

3.3.2 Working of Transformers

The core innovation of the transformer is the *self-attention* mechanism, which computes the attention scores for each word in a sentence, relative to all other words. This mechanism allows the model to focus on relevant parts of the input sequence when performing a task. Transformers consist of an encoder to process the input and a decoder to produce the output. Each of these is composed of multiple layers of self-attention and feed-forward neural networks.

3.3.3 Transformers in NLP

The advent of transformer technology in NLP has been a game-changer, giving rise to groundbreaking models such as **BERT** (Bidirectional Encoder Representations from Transformers)[3], **GPT** (Generative Pretrained Transformer)[15], and **T5** (Text-To-Text Transfer Transformer)[10]. These models have redefined performance standards across a range of NLP applications, including language comprehension, machine translation, and question-answering tasks.

3.3.4 Transformers in Computer Vision

The success of transformers in NLP has also been extended to the field of computer vision. Vision Transformer (ViT) is one such adaptation, which treats images as sequences of pixels or patches and applies the transformer architecture for tasks like image classification, object detection, and more. This has opened up new possibilities in integrating NLP and computer vision, particularly in multimodal tasks where understanding both textual and visual information is crucial.

3.4 Multimodal Models in Machine Learning

3.4.1 Definition and Importance

Multimodal models in Machine Learning are designed to process and relate information from multiple types of data sources or modalities, such as text, images, and audio. These models are

increasingly important in tasks where complex interpretations are required, involving more than one type of data input. For instance, in understanding human communication, both verbal (text or speech) and non-verbal cues (gestures, facial expressions) are essential.

3.4.2 Application Examples

In practice, multimodal models have been applied in various fields, demonstrating their versatility and effectiveness. Examples include:

- **Healthcare:** Analyzing medical images alongside clinical notes for better diagnosis.
- **Autonomous Vehicles:** Integrating visual, auditory, and sensor data for navigation and decision-making.
- **Social Media Analysis:** Combining text, image, and video data for enhanced content understanding and user interaction analysis.

3.4.3 Connection to Visual-WSD

Multimodal models are particularly relevant to Visual Word Sense Disambiguation (Visual-WSD) as they enable the integration of textual context with visual cues. This is essential for accurately interpreting the meaning of words in diverse contexts, which is central to Visual-WSD.

3.5 Model Training

3.5.1 Key concepts

- **Tensors:** Tensors are a central data structure in deep learning, acting as multidimensional arrays. In the context of neural networks, including transformers, tensors are used to store and process data, weights, and activations. They can be thought of as generalized matrices, with the capability to handle higher dimensions, which is essential for managing complex data structures like images, text sequences, or higher-dimensional features.
- **Loss Functions:** The loss function is a critical component in training, as it provides a measure of how well the model is performing. In classification tasks, common loss functions include cross-entropy loss, which compares the model's predicted probabilities (often in the form of logits) with the actual class labels. For regression tasks, mean squared error (MSE) is frequently used. For more specific tasks, there are more specific losses used, such as contrastive loss and others.
- **Logits:** In the context of deep learning, particularly in classification tasks, logits refer to the raw, unprocessed outputs of the final layer of a neural network. These logits are essentially the inputs to the final activation function like softmax, which transforms them into probabilities. Understanding logits is crucial, as they are the direct outputs from which the loss is calculated, influencing the model's learning.
- **Cosine Similarity:** Cosine similarity is a metric used to measure how similar two vectors are, irrespective of their size. Commonly used in text analysis, cosine similarity calculates the cosine of the angle between two non-zero vectors in their vector space.

3.5.2 Training Deep Learning Models

The process of training a deep learning model, including transformers and multimodal models, typically involves several key steps:

- **Forward Propagation:** Input data is passed through the network, layer by layer, using tensors. Each layer transforms its input tensor into a new output tensor using learned weights.
- **Calculating Loss:** After forward propagation, the model's output is compared with the target value, and a loss value is computed. This loss quantifies the model's error or deviation from the desired output.
- **Backward Propagation:** Using the loss value, the algorithm computes gradients for each weight in the network, a process known as backpropagation. These gradients represent the partial derivatives of the loss function with respect to each weight.
- **Weight Update:** The weights are then adjusted in the direction that reduces the loss, typically using optimization algorithms like stochastic gradient descent (SGD) or its variants.

Transformer models, renowned for their self-attention mechanisms, follow the same fundamental training principles. However, their architecture allows for more complex patterns and dependencies to be learned. In multimodal models, where different data types (e.g., text and images) are processed, tensors play an even more crucial role in handling and integrating diverse data formats. The training of such models often requires careful consideration of the interaction between modalities, and how loss functions can encapsulate the combined learning objectives.

3.5.3 Contrastive Learning

Contrastive Learning is a technique in the realm of Machine Learning, particularly within unsupervised and self-supervised learning frameworks. It revolves around the concept of learning by contrasting *positive pairs* against *negative pairs*. This approach enables models to understand and distinguish the underlying features of data points by focusing on what makes each data point unique compared to others.

In other words, Contrastive Learning involves training a model to bring closer (in terms of distance in a feature space) those data points that are similar (positive pairs) and push apart those that are dissimilar (negative pairs). The objective is to learn representations that are invariant to minor changes in the input data while being sensitive to the actual differences between distinct data points.

3.6 Evaluation Metrics

3.6.1 Importance of Metrics in Machine Learning

Evaluation metrics are essential in Machine Learning for assessing the performance of models. They provide a means to quantify accuracy and effectiveness, crucial for model comparison, parameter tuning, and guiding the research process. The choice of metrics is particularly important, as it can significantly influence how the performance of a model is perceived and interpreted.

3.6.2 Types of Metrics

- **Confusion Matrix Metrics:** Several key metrics used in Machine Learning are derived from the confusion matrix, which describes the performance of a classification model on test data with known true values. The primary metrics include: **Accuracy**, **Precision**, **Recall**, **F1 Score**. The **ROC AUC (Receiver Operating Characteristic - Area Under Curve)** is also a valuable metric, as it represents the model's ability to discriminate between classes.
- **Ranking Metrics:** For tasks where the ranking of predictions is critical, used ranking metrics include: **Mean Reciprocal Rank (MRR)**, **Precision at K (P@K)**, among others.
- **Distance Metrics:** In regression tasks or for measuring error in predictions, distance metrics like **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)** are commonly used. These metrics quantify the difference between predicted values and actual values, providing a measure of the model's prediction accuracy.
- **Other Metrics:** Depending on the specific requirements of the task, additional metrics such as **Sensitivity**, **Specificity**, **Normalized Discounted Cumulative Gain (NDCG)**, and **Average Precision (AP)** can be relevant to evaluate different aspects of model performance.

Metrics used in this work are described in details in **section 6.3**.

3.7 Losses in Model Training/Evaluation

Loss functions are fundamental in machine learning, providing a measure of how well the model is performing. In our Visual-WSD project, we employed various loss functions, each suited to different aspects of the learning process.

3.7.1 Cross-Entropy Loss

- **Description:** Cross-entropy loss, also known as log loss, measures the performance of a classification model. It calculates the difference between two probability distributions - the true distribution (actual labels) and the estimated distribution (model predictions).
- **Application:** Particularly useful in models where the output is a probability value and the goal is to minimize the difference between the predicted and actual distributions.
- **Formula:** The loss increases as the predicted probability diverges from the actual label:

$$H(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{3.1}$$

where

- $H(y, \hat{y})$ is the cross-entropy loss.
- y is the true label, a vector of ground truth values.
- $\hat{y}$ is the predicted probability distribution, a vector of predicted probabilities.

3.7.2 Contrastive Loss

- **Description:** Used mainly in tasks involving learning similarities and dissimilarities. It helps the model learn which data points are similar and which are different.
- **Application:** Effective in scenarios where the relationship between data points is more important than their individual classification.
- **Formula:** The loss is designed to ensure that similar items are closer in the embedding space compared to dissimilar ones:

$$L(Y, P_1, P_2) = (1 - Y)\frac{1}{2}(D_w)^2 + Y\frac{1}{2}\{\max(0, m - D_w)\}^2 \tag{3.2}$$

where

- $L(Y, P_1, P_2)$ is the contrastive loss.
- Y is a binary label indicating if the pair of samples (P_1, P_2) are similar (1) or dissimilar (0).
- D_w is the distance between the pair of samples in the learned embedding space.
- m is a margin that defines how far the embeddings of dissimilar pairs should be.

3.7.3 Triplet Loss

- **Description:** Triplet loss is used for learning fine-grained image similarity. It involves comparing a base image (anchor) with a positive image (similar to the anchor) and a negative image (different from the anchor).
- **Application:** This loss function is particularly beneficial for tasks requiring an understanding of nuanced differences between images, such as in Visual-WSD.
- **Formula:** The goal is to ensure that the anchor is closer to the positive image and farther from the negative image in the model's feature space:

$$L = \max\left(D(A, P) - D(A, N) + \text{margin}, 0\right) \tag{3.3}$$

where

- L is the triplet loss.
- $D(A, P)$ is the distance between the anchor and the positive sample.
- $D(A, N)$ is the distance between the anchor and the negative sample.
- margin is a predefined margin that is enforced between the positive and negative pairs.

Each of these loss functions contributed to refining our models' ability to effectively process and analyze the multimodal data in the Visual-WSD context, enhancing their performance and accuracy.

3.8 Relevance to the Thesis

This chapter has established the essential theoretical background for our Visual Word Sense Disambiguation (Visual-WSD) research, integrating key concepts from various domains.

- **Machine Learning and Deep Learning:** These foundational technologies enable the sophisticated data processing and pattern recognition capabilities central to our models.
- **Natural Language Processing:** NLP techniques are crucial for the textual analysis component of Visual-WSD, facilitating accurate language understanding and contextual interpretation.
- **Transformers:** Their advanced processing of sequential data and contextual nuances significantly enhance our model's multimodal capabilities, essential for interpreting text-image relationships.
- **Multimodal Models:** The core of our study, these models adeptly integrate and analyze both textual and visual inputs, critical for effective word sense disambiguation in varied contexts.
- **Evaluation Metrics:** The chosen metrics, allow for a comprehensive assessment of our models' performance in Visual-WSD.

In summary, the theoretical insights discussed here provide the framework for our approach to Visual-WSD, combining NLP, Computer Vision, and Machine Learning to develop and evaluate models capable of nuanced language and visual analysis. Every term mentioned is cited from **Pattern Recognition and Machine Learning**[2].

Chapter 4

Related works

The objective of Visual Word Sense Disambiguation (Visual-WSD) is to correctly interpret the meaning of a word within a given context, typically limited to a small amount of text, and match it with the appropriate image from a selection of possibilities. This task expands on traditional Word Sense Disambiguation, which focuses only on text, by incorporating visual elements. This integration reflects real-life situations where words are often understood in conjunction with visual cues.

Recent advancements in language models have propelled progress in this field, yet the challenge remains to effectively bridge the gap between text-only settings and those involving multiple modalities like images[5].

To tackle this, it was determined to employ models that excel in the image captioning domain, like OpenAI **CLIP** (Learning Transferable Visual Models From Natural Language Supervision)[9] and Salesforce **BLIP** (Bootstrapping Language-Image Pre-training)[7]. These models incorporate the forefront of this field, demonstrating exceptional ability to leverage extensive datasets for enhanced learning. Additionally, this thesis delves into various other models with distinct architectural approaches to image captioning, such as Google **ALIGN** (Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision)[4], Anthropic **BridgeTower** (BridgeTower: Building Bridges Between Encoders in Vision-Language Representation Learning)[14], and NVIDIA **GroupViT** (GroupViT: Semantic Segmentation Emerges from Text Supervision)[13]. The key differences are highlighted in **chapter 8**. The inspiration for this work stems from **SemEval-2023 Task 1: Visual Word Sense Disambiguation**[11], with an aim to develop novel solutions for this evolving challenge.

Our work utilized the benchmark provided in the **SemEval-2023 Task**[11], where the data is sourced from Wikidata, OmegaWiki, and BabelPic. BabelNet serves as a connecting medium there, linking these three resources. Additionally, **Flickr30k dataset**[8], typically a standard benchmark for sentence-based image description tasks, was adapted for our specific requirements. This was done to assess the performance of the chosen models on data that did not include any elements of disambiguation.

In our study, we adapt models originally developed for different image-text challenges, modifying and enhancing them to effectively address the Visual Word Sense Disambiguation problem. This approach aims to extend the applicability of these models to a new domain, demonstrating their versatility and potential for broader utility.

Chapter 5

Data procurement

5.1 Introduction

The foundation of our Visual Word Sense Disambiguation (Visual-WSD) project is built upon a comprehensive dataset. This dataset was sourced from one of the tasks of **SemEval 2023**, a series of evaluations designed to explore computational semantic analysis techniques. SemEval (Semantic Evaluation) tasks have historically been instrumental in advancing the field of natural language processing. For our project, we selected the largest available dataset, named "*train+trial*", totaling **17 GB** in size and comprising **13,159 images**.

5.2 Dataset Source and Structure

The Visual-WSD dataset is inherently complex, encapsulating a broad spectrum of English language contexts, including nuanced uses and even Latin names for flora and fauna. Each data sample includes a phrase with an ambiguous word, known to us, and ten candidate images. The challenge lies in correctly identifying or ranking these images, with the target image ideally positioned first. Notably, some images serve multiple samples, underscoring the dataset's diverse application.

5.3 Rationale for Restructuring

Originally, the dataset was partitioned into three subsets: training, trial and test. The trial set was a small representational part of the train set and our task required a unified dataset, necessitating the merger of these subsets. Restructuring was also essential to simplify the data structure for model evaluation and fine-tuning purposes, making it more logical and compatible with our specific tasks. Furthermore, the initial dataset was fragmented across four text files, prompting the need to consolidate this information into a single, cohesive *CSV* file.

5.4 Data Download and Preparation

Given the dataset's substantial size and our reliance on cloud sessions for model training, which necessitated repeated dataset downloads, we employed asynchronous programming. This approach significantly enhanced performance, accommodating the dataset's volume and the variable internet speeds encountered during downloads.

5.5 Baseline Dataset

In addition to the Visual-WSD dataset, our research leveraged the **Flickr30k** dataset as a baseline for the image captioning task. This dataset, typically used for sentence-based image description tasks, required transformation to align with our Visual-WSD objectives. We adapted it by selecting one caption and its corresponding image as the target, and then randomly choosing nine other images as candidate images. This transformation was executed at the code level, maintaining the original file structure of the dataset.

5.6 Technical Challenges

Our data procurement process encountered several technical hurdles, each requiring a tailored solution to ensure the efficiency and accuracy of our dataset preparation.

- **Repeated Restructuring in Cloud Sessions**: Due to the reliance on cloud sessions for model training, we faced the challenge of repeatedly restructuring the dataset in each new session. This necessitated the development of a robust pipeline to manage the restructuring process efficiently.
- **Handling Google Drive's Large File Downloads**: The process of downloading the sizable Visual-WSD dataset from Google Drive was complicated by a confirmation step for files too large for virus scanning. We overcame this by implementing a method to handle the confirmation token, ensuring smooth download of the dataset.
- **Managing Timeout Errors**: Given the substantial size of the dataset and variable internet speeds, we had to address frequent timeout errors. Our solution involved optimizing the download strategy to accommodate these fluctuations in internet connectivity.
- **Dataset Image Name Overlap**: The training and trial subsets of the Visual-WSD dataset had overlapping image names. To resolve this, we uniquely renamed images from the trial subset and updated corresponding references within the CSV file, ensuring consistency and accuracy in the unified dataset.
- **Caption Length Issue in Flickr30k Dataset**: Some captions in the Flickr30k dataset were too long for our model, leading to errors. To address this, we chose to filter out captions that exceeded a certain length, instead of truncating them, to maintain the integrity of the data.
- **Preventing Duplicate Candidate Images**: In adapting the Flickr30k dataset for our task, we encountered an issue where the same images were used across different caption pairs. This posed a risk of including the target image among the candidate images. We implemented a check to ensure the target image was not duplicated in the candidate set.

Chapter 6

Model acquisition and evaluation

6.1 Introduction

In the realm of Natural Language Processing (NLP), especially within its multimodal dimensions, the confluence of image captioning and Visual Word Sense Disambiguation (Visual-WSD) represents a frontier of exploration that is both challenging and immensely promising. This paper examines this intersection by deploying and analyzing a series of advanced models originally devised for image captioning tasks. These models have been instrumental in advancing our understanding of how visual and linguistic elements can be combined for enhanced machine understanding.

As it is stated in **chapter 4**, the following models were chosen for the experiment:

- OpenAI **CLIP** (Learning Transferable Visual Models From Natural Language Supervision)[9]
- Salesforce **BLIP** (Bootstrapping Language-Image Pre-training)[7]
- Google **ALIGN** (Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision)[4]
- Anthropic **BridgeTower** (BridgeTower: Building Bridges Between Encoders in Vision-Language Representation Learning)[14]
- NVIDIA **GroupViT** (GroupViT: Semantic Segmentation Emerges from Text Supervision)[13]

Initially designed for image captioning - a task that involves generating coherent and contextually relevant textual descriptions of visual data - these models excel in correlating images with appropriate text snippets, capitalizing on their training over extensive image-text pair datasets. However, our research extends their application into the domain of Visual-WSD, a nuanced task that not only pairs images with text but also demands the precise interpretation of words within text snippets, influenced and contextualized by accompanying visual cues.

Visual-WSD, therefore, is not just about the correlation of text and images but about embedding a deeper layer of understanding - discerning the correct meaning of a word within a textual fragment, in harmony with the visual context. This necessitates an advanced level of disambiguation, elevating the complexity of the task beyond standard image captioning.

Our research proposes the hypothesis that these models, with their inherent architectural prowess and learning capacities, can be adapted and augmented for the Visual-WSD task. The objective is to explore and delineate the effectiveness of each model in this new context, unraveling the specific architectural and algorithmic elements that contribute to their performance in Visual-WSD. This exploration aims to establish a foundational framework for future developments in this field, setting a benchmark for subsequent innovations and enhancements in multimodal NLP.

6.2 Model implementation

In our research, the robust evaluation and implementation of diverse models for Visual Word Sense Disambiguation (Visual-WSD) constrains a uniform and comprehensive approach. To achieve this, we introduced an ***Experimental Evaluation Framework***, embodied in an abstract base class named **BaseModel**. This framework was applied consistently across all selected models that are described in **section 6.1**.

6.2.1 Architecture of Experimental Evaluation Framework

Each of the models was implemented with the **Base Model** as its skeleton. The Base Model is designed as an abstract class to serve as the structural backbone for model evaluations. It encompasses critical attributes and methods, which are foundational to handling and evaluating each model:

- **Attributes:**
  - **model:** The pretrained model being evaluated
  - **processor:** Handles model's image processing and text tokenization
- **Methods:**
  - **process_image:** Dedicated to processing input images in line with a model's specifications
  - **process_text:** Handles the conversion of textual input into a format suitable for a model
  - **evaluate:** Integration and assessment of both image and text data, yielding a tensor of logits

6.2.2 Uniform Application Across Models

In applying this BaseModel, each of the aforementioned models was adapted into this framework while preserving their unique architectural and functional characteristics. For each model, the ***process_image*** and ***process_text*** methods were tailored to accommodate the specific preprocessing requirements intrinsic to each model's architecture. The ***evaluate*** method, consistently implemented across all models, was pivotal in ensuring that the assessment was not only about processing efficiency but also about the effectiveness of integrating and interpreting multimodal data (text and image) in the context of Visual-WSD, as well as fair approach to evaluation of each model. It allowed us to evaluate each model's performance under the ***same evaluative criteria***.

6.2.3 Significance of the Standardized Evaluation Framework

The adoption of a standardized experimental evaluation framework in our study is more than a methodological choice, it's rather a strategic decision that ensures the *integrity* and *comparability* of our evaluations. By employing a consistent evaluative structure, it makes it possible to draw more accurate comparisons across different models, providing a comprehensive understanding of their performance for Visual Word Sense Disambiguation problem. This approach not only facilitates a ***fair assessment*** but also paves the way for identifying potential areas for improvement and further development.

6.3 Metrics

6.3.1 Introduction

In the realm of Machine Learning and Natural Language Processing, the performance of a model is not just a measure of its accuracy, but a multi-dimensional evaluation of its ability to make precise, reliable, and relevant predictions. The choice of metrics for model evaluation is a critical step, as it directly influences how the model's performance is interpreted and understood.

The selected metrics offer unique insights into different aspects of the model's performance. They collectively provide a holistic view of the model's capabilities, including its effectiveness in classification tasks (**Accuracy, Precision, Recall, F1 Score**) and its proficiency in ranking and retrieval tasks (**MRR**).

6.3.2 Theoretical Background

At the core of most evaluation metrics are the concepts of ***True Positives, True Negatives, False Positives, and False Negatives***. These terms arise from what is known as a *confusion matrix*, a table used to describe the performance of a classification model on a set of test data for which the true values are known.

- **True Positives (TP):** These are cases where the model correctly predicts the positive class. **Example:** in a spam detection model, this would be the spam emails that the model correctly identifies as spam.
- **True Negatives (TN):** These are cases where the model correctly predicts the negative class. **Example:** in the spam detection, these would be the regular emails that the model correctly identifies as not spam.
- **False Positives (FP):** Also known as a Type I error, these occur when the model incorrectly predicts the positive class. **Example:** a regular email that is incorrectly flagged as spam.
- **False Negatives (FN):** Also known as a Type II error, these occur when the model incorrectly predicts the negative class. **Example:** in the spam detection context, this would be a spam email that is incorrectly identified as not spam.

For better understanding of **Mean Reciprocal Rank (MRR)** described further we need to elaborate on the following terms in the context of this particular metric:

- **Rank:** It is the position of the first relevant or correct answer in the list of predictions provided by the model for a given query.
- **Reciprocal Rank:** This is the multiplicative inverse of the rank of the first relevant answer. **Example:** if the correct answer is at the 3rd position, the reciprocal rank is 1/3.
- **Query:** It refers to an individual instance of a request for information that the model is expected to provide ranked responses. **Example:** this could be a search term in a search engine.

Understanding these components is crucial for interpreting the subsequent metrics, as they form the basis of most evaluation techniques in classification models as well as ranking task employed in our solution. All terms described are referenced from **Pattern Recognition and Machine Learning**[2].

6.3.3 Metrics Description

Accuracy measures the proportion of true results (both *True Positives* and *True Negatives*) among the total number of cases examined. It is a useful metric for general performance but can be misleading in cases where the class distribution is imbalanced.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Precision assesses the model's accuracy in predicting positive labels. It represents the proportion of positive identifications that were actually correct.

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{6.2}$$

Recall measures the proportion of actual positives that were correctly identified. It is crucial in situations where missing a positive instance (*False Negative*) carries a significant cost.

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{6.3}$$

F1 Score is the harmonic mean of *Precision* and *Recall*, providing a balance between the two. It is particularly useful when you need to take both False Positives and False Negatives into account.

$$\textbf{F1} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{6.4}$$

MRR (Mean Reciprocal Rank) is a measure used to evaluate the effectiveness of a model in scenarios where the response is a ranked list of items. It is defined as the average of the reciprocal ranks of the first correct answer among a set of queries. The rank of an answer is the position it appears in the model's list of predictions.

$$\textbf{MRR} = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank_i} \tag{6.5}$$

where,

- Q (Query) - total number of queries
- $rank_i$ - rank position of the first relevant answer for the i-th query

Each of these metrics contributes to a comprehensive evaluation framework, allowing us to assess the model's performance with nuance and precision. They provide insights into the model's accuracy and effectiveness in various scenarios, including error management and ranking effectiveness in retrieval tasks.

6.3.4 Mean Reciprocal Rank in Our Study

It is important to clarify the specific application of **Mean Reciprocal Rank (MRR)** in our study. As it is mentioned in **subsection 6.3.3** above, MRR is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. It is calculated as the average of the reciprocal ranks of the first correct answer among a set of predictions for a given set of queries.

In Visual-WSD, the queries represent the visual contexts surrounding an ambiguous word, while the responses consist of a hierarchically arranged set of images. The image at the top of this ranking is the one that best aligns with the most appropriate meaning of the query. A correct prediction by the model is indicated when the target image achieves the highest rank.

To illustrate the application of MRR in our evaluations, the following table provides a exemplary snapshot of how the model might rank images in response to certain queries.

Table 6.1
Example of MRR computation on Visual-WSD Dataset

Query	Ambiguous Word in Visual Context	Model's Ranked Images	Target Image Position	Reciprocal Rank (1/Position)
Q1	[Ambiguous Word with Context 1]	[Img3, Img1, Img5, ...]	2	0.50
Q2	[Ambiguous Word with Context 2]	[Img2, Img4, Img1, ...]	1	1.00
Q3	[Ambiguous Word with Context 3]	[Img5, Img1, Img3, ...]	4	0.25
		...		

Table 6.1 aligns with the specific application of MRR in your study. Each row in the table represents a distinct query, comprising an ambiguous word within a visual context. ***Model's Ranked Images*** column shows the order in which the model has ranked the candidate images based on their relevance to the query. ***Target Image Position*** reflects the position or rank of the correct image (the one that actually corresponds to the ambiguous word in the given visual context) within the model's ranked list, while ***Reciprocal Rank*** column demonstrates the calculation of MRR for each individual query. The goal in a well-performing model is for this position to be as close to 1 as possible, indicating that the model consistently ranks the correct image at or near the top of its list. This way, the ***Target Image Position*** equal to 1 would indicate the absolutely correct prediction of the model. **Table 6.1** demonstrates the way MRR offers insights into the model's capability to discern and prioritize the most relevant visual response to the given textual context.

Chapter 7

Model evaluation in practice

7.1 Introduction

The evaluation of models in the field of Natural Language Processing, and more specifically in Visual Word Sense Disambiguation, plays a pivotal role in advancing our understanding and capabilities within this domain. This chapter delves into the practical aspects of evaluating models, bridging the gap between theoretical knowledge and its application in real-world scenarios.

Our approach to model evaluation is comprehensive, encompassing various stages from dataset preparation to the intricate process of model fine-tuning and analysis. Through this process, we hope to shed light on the multifaceted nature of model evaluation in Visual-WSD, offering a detailed guide that encapsulates the essence of practical model evaluation.

7.2 Dataset Description and Preparation

The foundation of our model evaluation process in Visual Word Sense Disambiguation is a meticulously structured dataset, crucial for ensuring the accuracy and relevance of our results. This dataset comprises a blend of visual and textual elements, structured as follows:

Dataset Structure: The dataset consists of a folder containing images and a corresponding *CSV* file. The *CSV* file is structured with 12 columns, encompassing both the textual and visual aspects required for Visual-WSD.

- **Ambiguous Word:** The first column lists the ambiguous word that is the focal point of the Visual-WSD task.
- **Ambiguous Phrase:** The second column contains the phrase in which the ambiguous word is used, providing contextual information.
- **Target Image:** The third column specifies the filename of the target image that correctly represents the ambiguous word in its given context.
- **Candidate Images:** The subsequent nine columns list the filenames of wrong candidate images that can be potentially related to the phrase.

7.3 Train-Test Split

A critical step in preparing our dataset for Visual Word Sense Disambiguation was the division into training and testing sets. This division is essential for assessing the generalizability and performance of our models on unseen data.

- **Split Ratio:** The dataset was split into two subsets: the training set and the testing set. The training set is used to train the model, while the testing set is reserved for evaluating the model's performance. The split ratio, typically a percentage of the total dataset, is a crucial parameter. We chose a split ratio that ensures a sufficient amount of data for both training and testing, while maintaining a balance to avoid overfitting or underfitting.
- **Random Sampling:** To ensure the representativeness of both training and testing sets, the splitting process involved random sampling of the data. This randomness helps in avoiding any bias in the model training and ensures that the evaluation on the test set is reliable.
- **Consistency Across Models:** It is important that each model evaluated in our study is trained and tested on the same split of the dataset. This consistency is crucial for a fair comparison of model performances.

The train-test split is a fundamental practice in machine learning, providing us with a reliable method to train our models effectively and evaluate their performance objectively on unseen data.

7.4 Data Loading and Preprocessing

In the context of our Visual-WSD project, efficient and effective data handling is paramount. The data loading and preprocessing steps are facilitated through a custom Python class, ***VisualWSDDataset***, and data loader functions, designed specifically for our dataset.

7.4.1 VisualWSDDataset Class

The *VisualWSDDataset* class, inheriting from PyTorch's Dataset class, is tailored for handling the Visual-WSD dataset. It supports both training and evaluation modes, allowing for flexible dataset management. Key functionalities include:

- **Initialization:** The class initializes with the dataset's path, CSV file, and images folder. It also accepts optional parameters for image transformations and specifies the mode (training or evaluation).
- **Train-Test Split:** Utilizing the ***train_test_split*** function, the dataset is divided into training and testing subsets based on the specified ratio.
- **Image Processing:** Images are loaded and converted to RGB format. Custom transformations (such as resizing and cropping) are applied if specified.
- **Data Retrieval:** The ***__getitem__*** method retrieves a single data sample, including the word, context, target image, and candidate images.

7.4.2 Data Loaders

Data loaders are crucial for batch processing of the dataset during model training and evaluation. They are implemented using PyTorch's DataLoader class. The data loader functions are designed to handle:

- **Batch Processing:** Configuring batch size for efficient model feeding.
- **Shuffling:** Optionally shuffling the data to ensure randomization, which is important for training.

- **Multithreading:** Utilizing multiple workers for parallel data loading, enhancing performance.

These components collectively ensure that our dataset is optimally prepared and presented to the models, facilitating accurate and efficient Visual-WSD task execution.

7.5 Tokenization Process

Tokenization is a foundational step in preparing textual data for Natural Language Processing tasks, including Visual Word Sense Disambiguation. In our project, the tokenization process plays a vital role in converting raw text from our dataset into a structured format that our models can interpret and analyze.

Tokenization involves breaking down text into smaller units, typically words or subwords. This process is essential for:

- **Standardizing Text:** Converting text into a uniform format that the models can process.
- **Facilitating Analysis:** Allowing models to understand and analyze the text at a granular level.

For the Visual-WSD task, our tokenization process includes:

- **Processing Ambiguous Phrases:** Tokenizing the phrases containing the ambiguous words to create a sequence of tokens representing each phrase.
- **Handling Contextual Information:** Ensuring that the context around the ambiguous words is preserved during tokenization, which is crucial for accurate sense disambiguation.

We utilize advanced NLP tools and techniques for tokenization, ensuring that the textual data is optimally prepared for our models. These tools are capable of handling various linguistic nuances, making them suitable for the diverse and complex nature of the Visual-WSD dataset.

Through effective tokenization, we lay the groundwork for our models to accurately process and interpret the textual component of our dataset, which is a critical step in achieving reliable Visual-WSD.

7.6 Logits and Softmax

In our Visual-WSD research, we implement several state-of-the-art models, described in **section 6.1**, each chosen for its unique ability to process and interpret complex multimodal data. These models, originally designed for image captioning tasks, are adapted for the nuanced demands of Visual-WSD. The core functionality of these models lies in their ability to generate logits, which are the raw outputs before applying any activation function. These logits represent the model's predictions, which then can be transformed using different functions to get probabilities. The softmax function plays a crucial role in transforming the logits into probabilities, providing a basis for calculating the evaluation metrics defined in **subsection 6.3.3**. This probabilistic output is crucial for tasks like ranking candidate images in Visual-WSD.

$$\mathbf{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \tag{7.1}$$

7.7 Hyperparameters in Model Training/Evaluation

Hyperparameters play a vital role in the configuration and optimization of machine learning models. In our Visual-WSD project, careful selection and tuning of hyperparameters were essential to achieve optimal model performance.

Hyperparameter tuning involves finding the right combination of parameters that governs the learning process of the model. This process is crucial as it directly affects the efficiency and accuracy of the model in training and evaluation phases.

7.7.1 Key Hyperparameters in Our Models

- **Learning Rate:** One of the most critical hyperparameters, determining the step size at each iteration while moving toward a minimum of a loss function.
- **Learning Rate Schedulers:** Learning rate schedulers are used to change the learning rate during training. They adjust the learning rate according to certain rules or criteria, such as the number of epochs or the reduction in loss over time.
- **Batch Size:** Influences the model's convergence rate and training stability. A larger batch size ensures more stable convergence, but requires more memory.
- **Number of Epochs:** Dictates the number of times the entire dataset is passed forward and backward through the neural network.
- **Regularization Parameters:** Includes techniques like dropout and L2 regularization to prevent overfitting of the model on training data.

7.7.2 Impact on Model Performance

The fine-tuning of hyperparameters had a significant impact on the overall performance of our models. It enhanced the models' ability to learn from the training data and improved their generalization capabilities on unseen data.

We employed methods like grid search and random search for hyperparameter optimization, allowing us to systematically explore multiple combinations of hyperparameters to find the most effective model configuration.

The careful selection and tuning of hyperparameters were pivotal in our Visual-WSD project, significantly contributing to the models' efficacy in disambiguating and understanding visual-textual relationships.

7.8 Challenges in Model Evaluation

Throughout the model evaluation phase of our Visual-WSD project, we encountered several challenges that required careful solutions to ensure accurate and efficient model performance.

7.8.1 Challenge 1: Managing Large Datasets

- **Description:** Handling the large size of the Visual-WSD dataset posed significant challenges, particularly in terms of memory management and processing speed.
- **Solution:** We employed efficient data batching and loading techniques to manage memory usage effectively. Additionally, parallel processing and cloud computing resources were utilized to enhance processing speed.

7.8.2 Challenge 2: Integrating Multimodal Data

- **Description:** Effectively integrating and processing multimodal data (textual and visual) was complex, requiring careful coordination between different model components.
- **Solution:** We focused on refining data preprocessing steps and fine-tuning model architectures to better handle the intricacies of multimodal data integration.

7.8.3 Challenge 3: Computational Resource Limitations

- **Description:** Limited access to high-performance computing resources occasionally hindered the model training and evaluation process.
- **Solution:** To mitigate this, we optimized our models for computational efficiency and leveraged cloud-based platforms to access additional resources when needed.

7.8.4 Conclusion

These challenges, while demanding, provided valuable learning opportunities. The solutions we developed not only enhanced the performance and reliability of our models but also enriched our understanding of the complexities involved in Visual-WSD model evaluation.

Chapter 8

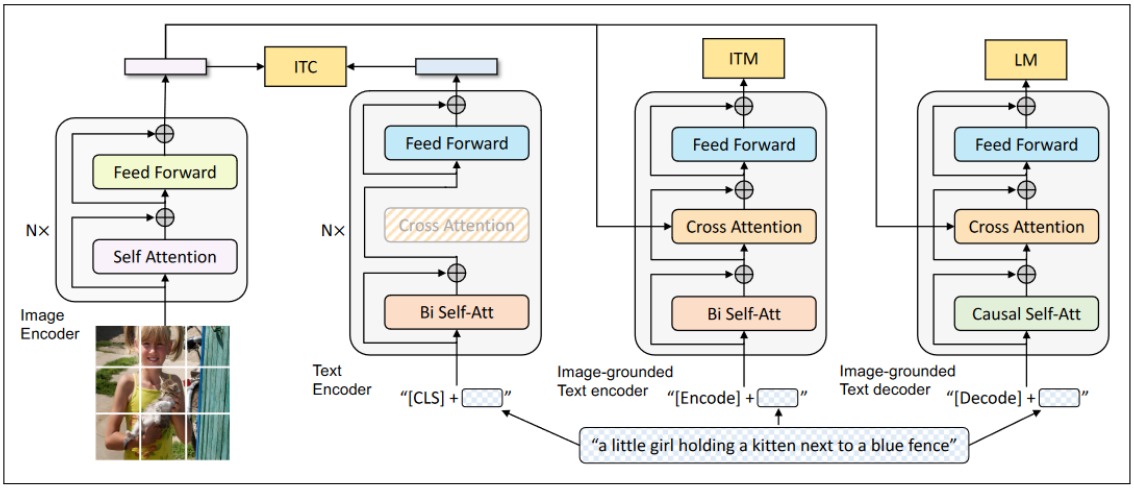
Model comparison

8.1 Architecture Comparison

This section discusses the core architectural elements of the analyzed models, highlighting their differences, advantages, and disadvantages.

8.1.1 BLIP

Figure 8.1
Visualised BLIP architecture [7]



BLIP (Bootstrapped Language Image Pretraining) model introduces a novel approach to vision-language tasks by leveraging a pre-trained vision transformer and language model.

8.1.1.1 Querying Transformer (Q-Former)

BLIP incorporates a **Querying Transformer** (Q-Former) depicted on the left of **Figure 8.1** as a lightweight module to bridge the modality gap between a frozen image encoder and a frozen language model shown an the center and on the right of **Figure 8.1**. This innovative design helps facilitate the flow of information between the vision and language modalities, enhancing the overall model performance.

8.1.1.2 Two-Stage Architecture

The **BLIP** model comprises two distinct stages:

- **Vision-Language Representation Learning:** In the first stage, **BLIP** focuses on learning representations from a frozen image encoder. This stage aims to capture the intermodal relationships between vision and language, laying the foundation for subsequent generative tasks.
- **Vision-to-Language Generative Learning:** The second stage involves generative learning from a frozen language model. **BLIP** employs the pre-trained language model to generate textual descriptions from the learned visual representations, providing a comprehensive understanding of the vision-to-language mapping.

8.1.1.3 Image Encoder

BLIP utilizes a **Vision Transformer** (ViT) as its *image encoder*. The ViT architecture:

- divides an image into fixed-size patches
- linearly embeds each patch
- processes them through **Transformer** blocks

This methodology enables the model to learn spatial hierarchies and features from images, contributing to enhanced representation learning.

8.1.1.4 Text Decoder

For the language component, **BLIP** employs a **Transformer-based decoder** similar to those found in models like *GPT*. The decoder takes the encoded image representations and generates captions through *autoregressive prediction*, producing coherent and contextually relevant textual output.

8.1.1.5 Training Objectives

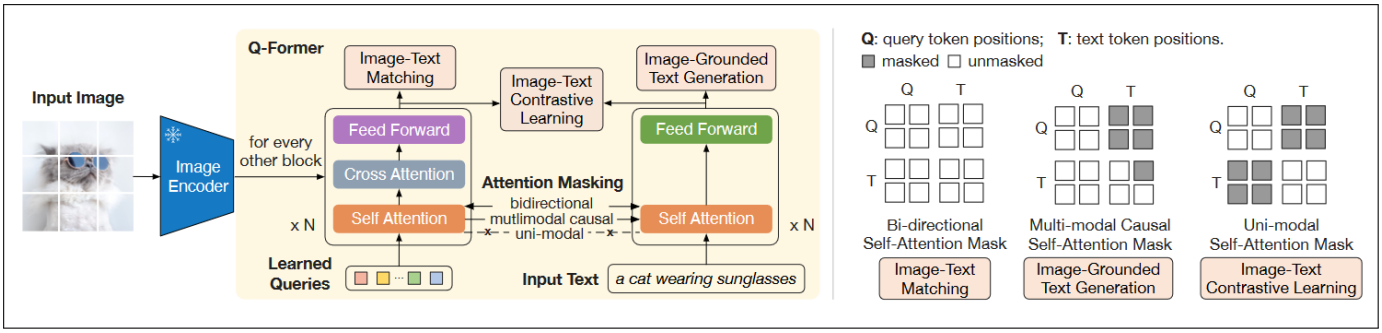
BLIP adopts a dual training approach, incorporating a combination of objectives:

- **Image-Text Matching: BLIP** is trained to match images with their corresponding textual descriptions, fostering a strong alignment between vision and language representations.
- **Autoregressive Language Modeling:** The model is trained to generate captions autoregressively, predicting one word at a time. This objective enhances the model's language generation capabilities, allowing it to produce contextually coherent and semantically meaningful descriptions.

The combination of these training objectives enables **BLIP** to develop a robust understanding of the intricate relationship between images and their textual counterparts, making it a promising model for vision-language tasks.

8.1.2 BLIP-2

Figure 8.2
Visualised BLIP-2 architecture [6]



BLIP-2, an evolution of the **BLIP** model, aims to further enhance the efficiency and effectiveness of vision-language pre-training by building upon the foundation laid by its predecessor. It is reasonable to infer that it incorporates improvements over the original **BLIP** model.

The model, like its predecessor, adheres to a dual-encoder architecture, featuring separate encoders for processing visual and textual information. This architectural choice enables the model to independently encode and learn intricate representations from both modalities, fostering a comprehensive understanding of the interplay between images and text.

8.1.2.1 Image Encoder

In the architecture of **BLIP-2**, a **Vision Transformer** (ViT) indeed serves as the image encoder which is shown on left of **Figure 8.2**. The ViT architecture has undergone improvements and refinements to enhance its ability to capture spatial hierarchies and features from images. These modifications contribute to a more effective process of representation learning within the image encoding stage of **BLIP-2**.

8.1.2.2 Text Decoder

In **BLIP-2**, the text decoder shown at the center and on the right of **Figure 8.2** is indeed a more advanced iteration of the **Transformer-based decoder** utilized in *BLIP*. This advancement encompasses improvements in autoregressive language modeling, leading to the generation of more accurate and contextually rich captions. The enhancements in the text decoder contribute to the model's capability to produce linguistically sophisticated and contextually meaningful textual outputs during the caption generation process.

8.1.2.3 Enhancements

In its architectural evolution, **BLIP-2** has indeed introduced several enhancements aimed at improving its overall performance.

Improved Attention Mechanisms: **BLIP-2** incorporates advanced attention mechanisms, enabling the model to focus more effectively on relevant regions of input data during both representation learning and generative tasks.

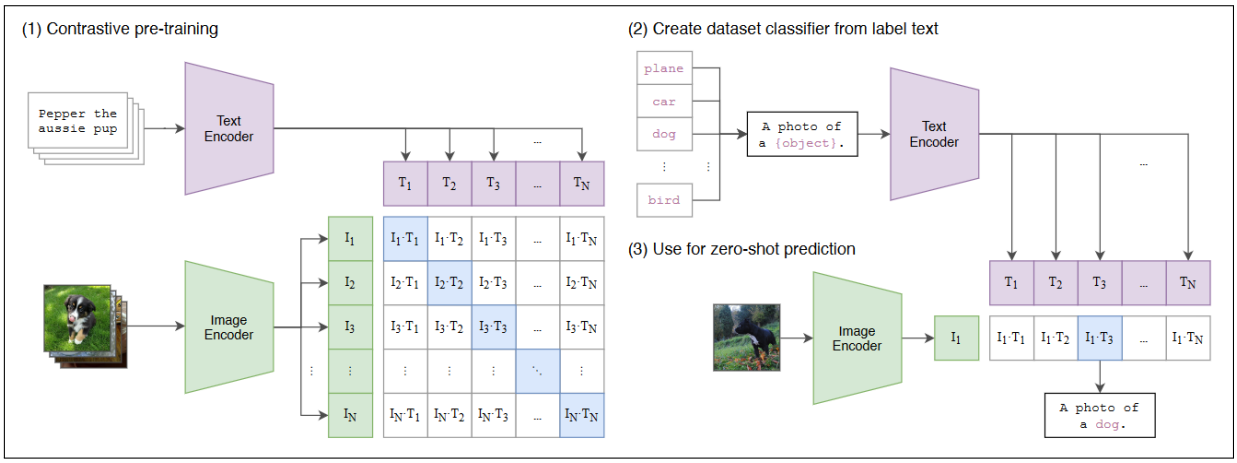
Efficient Training Strategies: The model implements more efficient training strategies, optimizing the learning process and reducing the computational resources required for pre-training. These strategies contribute to enhanced training efficiency without compromising the quality of learned representations.

Larger and More Diverse Datasets: **BLIP-2** benefits from the use of larger and more diverse datasets for training, providing the model with a broader understanding of complex image-text relationships and significantly enhancing its generalization capabilities.

These actualized enhancements collectively position **BLIP-2** as a more powerful and efficient vision-language pre-training model. The implementation of advanced attention mechanisms, efficient training strategies, and the utilization of diverse datasets align with the overarching objective of advancing the model's capacity to comprehend and generate meaningful captions for diverse visual inputs.

8.1.3 CLIP

Figure 8.3
Visualised CLIP architecture [9]



CLIP (Contrastive Language-Image Pre-training) is a state-of-the-art vision-language model that employs a unique dual-encoder architecture, **contrastive learning**, and demonstrates impressive zero-shot capabilities.

8.1.3.1 Dual Encoders

CLIP utilizes a dual-encoder architecture, consisting of two separate networks for processing images and text:

Image Encoder: Implemented using a **Vision Transformer** (ViT) like the previously described models.

Text Encoder: **Transformer-based language model**. This encoder is responsible for converting textual input into a high-dimensional vector representation. The use of two separate encoders allows **CLIP** to independently capture the features of images and text.

8.1.3.2 Contrastive Learning

CLIP's training methodology visualised on **Figure 8.3** (1) is centered around **contrastive learning**, a technique that aligns image and text representations in a shared embedding space. The model is trained to maximize the **cosine similarity** between correct pairs of images and their corresponding text descriptions while minimizing this similarity for incorrect pairs.

This approach encourages the model to learn a meaningful alignment between visual and textual modalities.

8.1.3.3 Zero-shot Capabilities

One of **CLIP's** notable strengths is its ability to perform well in **zero-shot** scenarios visualized on **Figure 8.3** (2) and (3). This means that **CLIP** can generalize to tasks without explicit training on them.

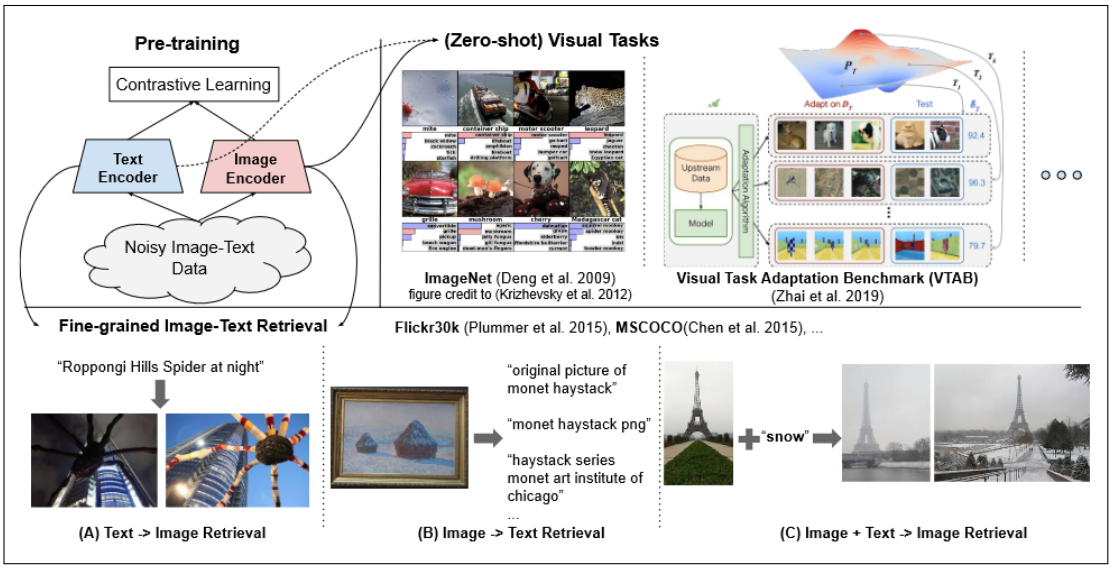
The architecture and training of **CLIP** enable it to understand a broad range of concepts, allowing it to make accurate predictions and generate relevant textual descriptions even for tasks it has not been explicitly trained on. This **zero-shot** capability extends to tasks such as image captioning, showcasing the model's versatility and generalization power.

8.1.3.4 CLIP Architecture Summary

In summary, **CLIP's** dual-encoder architecture, **contrastive learning** approach, and **zero-shot** capabilities make it a formidable model for vision-language tasks, showcasing its ability to learn rich representations and generalize effectively across diverse tasks.

8.1.4 ALIGN

Figure 8.4
Visualised ALIGN architecture [4]



ALIGN (A Large-scale Image and No-text model), a robust vision-language model, shares similarities with **CLIP** in its dual-encoder architecture, but it distinguishes itself through a scalable approach to **contrastive learning** and large-scale training on diverse datasets.

8.1.4.1 Dual Transformer Architecture

ALIGN, like **CLIP**, adopts a dual-encoder architecture, employing separate **Transformer networks** for processing images and text as shown on **Figure 8.4**:

Image Encoder: Implemented using a **Transformer network**, processes visual input to generate high-dimensional embeddings. This Transformer-based architecture allows **ALIGN** to capture intricate visual features and relationships within images.

Text Encoder: Based on a **Transformer network**, converts textual input into embeddings within the shared space. The use of **Dual Transformers** enables **ALIGN** to independently capture the features of both modalities, fostering a comprehensive understanding of images and text.

8.1.4.2 Large-scale Training

One distinguishing feature of **ALIGN** is its extensive training on a **large-scale dataset**. This dataset encompasses a wide variety of images and textual content, collected from the web. This approach enables **ALIGN** to learn robust representations from a vast and diverse set of visual and textual data, enhancing its ability to generalize effectively across different tasks.

8.1.4.3 Scalable Contrastive Learning

ALIGN employs a scalable approach to **contrastive learning**, allowing it to train on a large-scale and noisy dataset. The model is trained to match image and text pairs, emphasizing robust representation learning even in the presence of noisy or diverse data. This **scalable contrastive learning** strategy contributes to **ALIGN's** adaptability to various real-world scenarios.

8.1.4.4 Image-Text Alignment

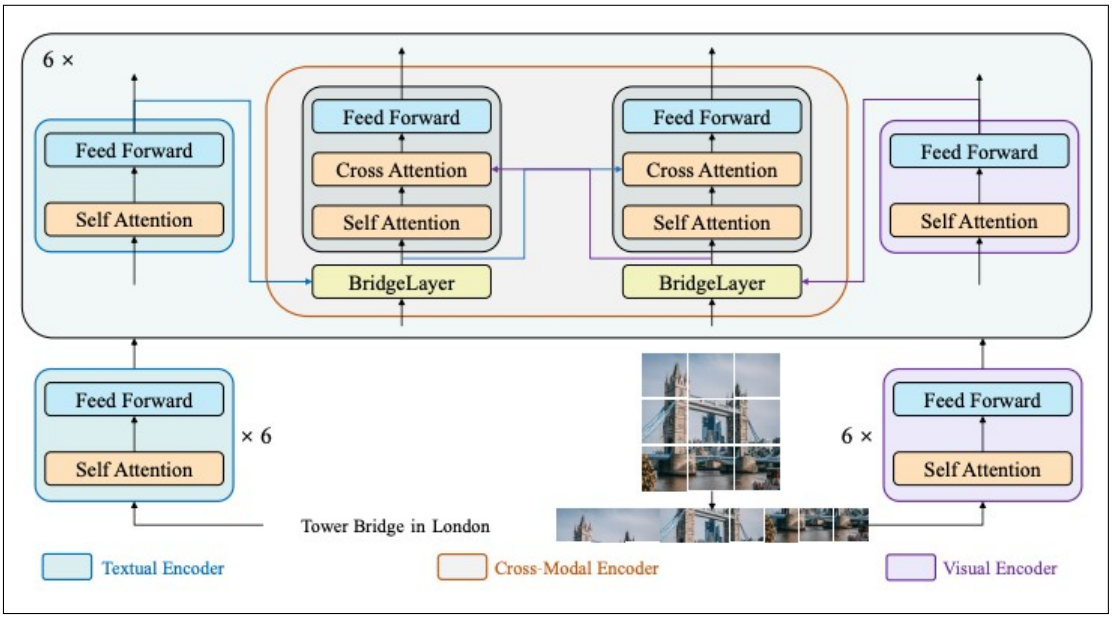
A key focus of **ALIGN** is on aligning the representations of images and text in the shared embedding space. The model strives to ensure that corresponding image and text pairs are close in this space, facilitating a deep understanding of the relationship between visual and textual information. This capability proves valuable in tasks requiring a nuanced comprehension of image-text relationships, such as image captioning.

8.1.4.5 ALIGN Architecture Summary

In summary, **ALIGN's** dual Transformer architecture, **large-scale training**, and **scalable contrastive learning** make it a powerful vision-language model. Its ability to align image and text representations effectively positions **ALIGN** as a versatile and robust solution for tasks demanding a nuanced understanding of visual and textual content.

8.1.5 BridgeTower

Figure 8.5
Visualised BridgeTower architecture [14]



The **BridgeTower** (Building Bridges Between Encoders in Vision-Language Representative Learning) model represents an innovative approach in the field of multi-modal learning. Its primary goal is to establish a robust connection between *uni-modal* and *cross-modal* encoders. This is achieved through the integration of multiple layers that facilitate comprehensive and detailed interactions at each stage of the **cross-modal encoding** process.

The **BridgeTower** model comprises three key components: a **visual encoder**, a **textual encoder**, and a **cross-modal encoder**. Each of these components plays a vital role in the overall functionality of the model.

8.1.5.1 Visual Encoder

The visual encoder shown in purple color on **Figure 8.5** is responsible for processing visual inputs. It extracts features and patterns from raw images, translating them into a format that can be integrated with textual data. This component utilizes advanced **Convolutional Neural Networks** (CNNs) to capture the intricacies of visual information.

8.1.5.2 Textual Encoder

Similarly, the textual encoder depicted in cyan color on **Figure 8.5** deals with textual inputs. It analyzes and encodes text data, preparing it for interaction with the visual data. This encoder typically employs **Recurrent Neural Networks** (RNNs) or **Transformers** to understand and process linguistic structures and semantics.

8.1.5.3 Cross-Modal Encoder

The cross-modal encoder highlighted in orange on **Figure 8.5** is the centerpiece of the **BridgeTower** model. It merges the outputs of the visual and textual encoders to create a unified representation

of both modalities. This encoder is designed to facilitate deep and intricate interactions between the visual and textual features.

8.1.5.4 Bridge Layers

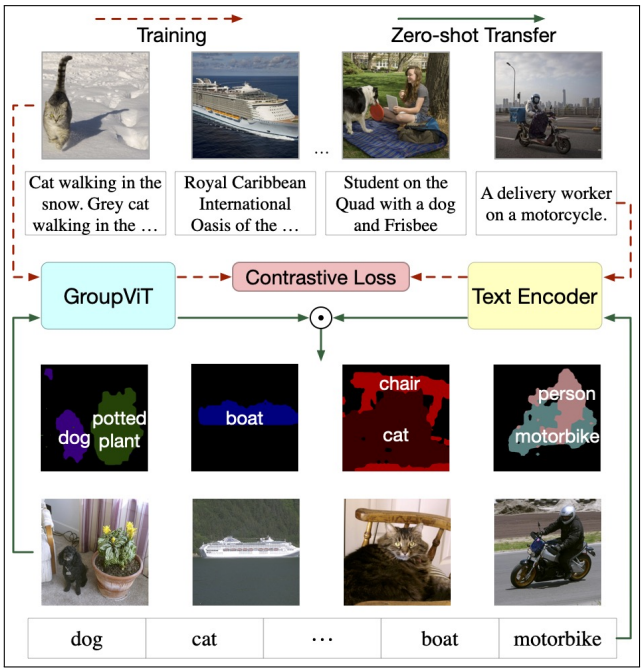
A unique feature of the cross-modal encoder is its multiple **lightweight bridge layers**. These layers are strategically positioned to build a ***bridge*** between each **uni-modal encoder** and the **cross-modal encoder**. They enable the cross-modal encoder to interact comprehensively and in detail with each layer of the uni-modal encoders. This architecture allows for a more thorough integration of visual and textual data, leading to improved performance in multi-modal tasks.

8.1.5.5 BridgeTower Architecture Summary

In conclusion, the **BridgeTower** model represents a significant advancement in **multi-modal learning**. Its architecture, especially the incorporation of **bridge layers** in the **cross-modal encoder**, offers a promising approach to understanding and integrating complex **multi-modal data**. This model has the potential to contribute substantially to various applications in the field of **Artificial Intelligence**.

8.1.6 GroupViT

Figure 8.6
Visualised GroupViT architecture [13]



GroupViT is a notable vision-language model that stands out through its unique architectural choices and training strategies.

8.1.6.1 Dual Encoder Architecture

Similar to **CLIP** and **ALIGN**, **GroupViT** employs a dual-encoder architecture, featuring separate encoders for processing images and text which can be seen on **Figure 8.6**.

Image Encoder: Utilized a **Vision Transformer** (ViT) which was mentioned in the **BLIP** family and **CLIP** model. This allows **GroupViT** to capture spatial hierarchies and features from images effectively.

Text Encoder: Based on a **Transformer network**, converting textual input into embeddings. The dual **Transformer architecture** enables **GroupViT** to independently encode features from both images and text, contributing to a comprehensive understanding of the vision-language relationship.

8.1.6.2 Group-wise Contrastive Learning

A distinctive aspect of **GroupViT** is its emphasis on **group-wise contrastive learning**.

Enhanced Contrastive Learning: **GroupViT** employs a **contrastive learning** approach similar to **CLIP** and **ALIGN** but introduces a group-wise strategy. Instead of considering individual image-text pairs, **GroupViT** groups similar samples together during **contrastive learning**. This group-wise approach enhances the model's ability to capture diverse visual and textual concepts simultaneously.

Improved Robustness: By considering groups of samples, **GroupViT** enhances its robustness to variations within the data. This strategy helps the model generalize better in the presence of diverse image and text representations, making it well-suited for real-world applications with varying data characteristics.

8.1.6.3 Adaptive Training Strategies

GroupViT incorporates adaptive training strategies to further enhance its performance.

Dynamic Learning Rates: **GroupViT** adjusts learning rates dynamically during training, allowing it to adapt to the complexity and characteristics of the data. This adaptive approach contributes to improved convergence and better handling of diverse visual and textual patterns.

8.1.6.4 Comprehensive Vision-Language Understanding

GroupViT's architectural choices and training strategies collectively contribute to a model with a comprehensive understanding of vision and language. The dual **Transformer encoders**, **group-wise contrastive learning**, and adaptive training strategies position **GroupViT** as a versatile vision-language model capable of capturing nuanced relationships between diverse images and textual content.

8.1.6.5 GroupViT Architecture Summary

In summary, **GroupViT's** dual encoder architecture, **group-wise contrastive learning**, and adaptive training strategies make it a distinctive and effective model for vision-language tasks, showcasing its ability to handle diverse data and enhance the understanding of complex visual and textual relationships.

8.1.7 Number of Parameters Comparison

Table 8.1
Number of parameters comparison

Model	Pretrained Model Identifier	Total Parameters
BLIP	Salesforce/blip-image-captioning-base	224 726 017
BLIP-2	Salesforce/blip2-opt-2.7b	2.7 billion
CLIP	openai/clip-vit-base-patch32	151 277 313
ALIGN	kakaobrain/align-base	172 117 841
BridgeTower	BridgeTower/bridgetower-base	326 347 008
GroupViT	nvidia/groupvit-gcc-yfcc	55 726 609

Huge number of **BLIP-2** parameters makes it very computationally expensive. Challenges arising from this are elaborated upon in **Section 8.2.1**.

The number of parameters across the vision-language models varies significantly, indicating diverse model complexities and capacities for capturing nuanced relationships between images and text.

This comparison highlights the substantial diversity in model sizes, with **BLIP-2** having the highest number of parameters, followed by **BridgeTower**. In contrast, **GroupViT** has a comparatively smaller parameter count, showcasing a range of model capacities to accommodate different use cases and computational resources.

8.1.8 Summary

In the landscape of vision-language models, several distinct approaches have emerged, each characterized by unique architectural choices and training strategies. **CLIP** and **ALIGN**, for instance, share similarities in their use of dual-encoder architectures and **contrastive learning** to align image and text representations. **BLIP** takes a more intricate approach, employing **bootstrapping** from pre-trained models and a two-stage training process with a **Querying Transformer**. **BLIP** and its successor **BLIP-2** are particularly focused on the captioning task, while **CLIP** and **ALIGN** showcase versatility in handling various image-text-related tasks.

GroupViT, on the other hand, stands out for its notable dual **Transformer architecture** and innovative **group-wise contrastive learning** strategy. Utilizing a **Vision Transformer** (ViT) for image encoding and a **Transformer network** for text encoding, **GroupViT** enhances robustness through **group-wise contrastive learning**, grouping similar samples together during training. This unique approach contributes to improved model generalization, making **GroupViT** well-suited for real-world applications with diverse data characteristics. Adaptive training strategies, such as dynamic learning rates, further enhance the model's performance, providing a comprehensive understanding of vision and language relationships.

In contrast, the **BridgeTower** model represents an innovative approach in **multi-modal learning**, emphasizing the establishment of a robust connection between **uni-modal** and **cross-modal** encoders. Comprising visual, textual, and **cross-modal** encoders, the **BridgeTower** model employs advanced **Convolutional Neural Networks** (CNNs) and **Recurrent Neural Networks** (RNNs) or **Transformers** to process visual and textual inputs, respectively. The unique feature of multiple bridge layers within the **cross-modal encoder** enables nuanced interactions between **uni-modal** and **cross-modal** representations, promising a thorough integration of complex **multi-modal** data.

8.2 Results of Experiments

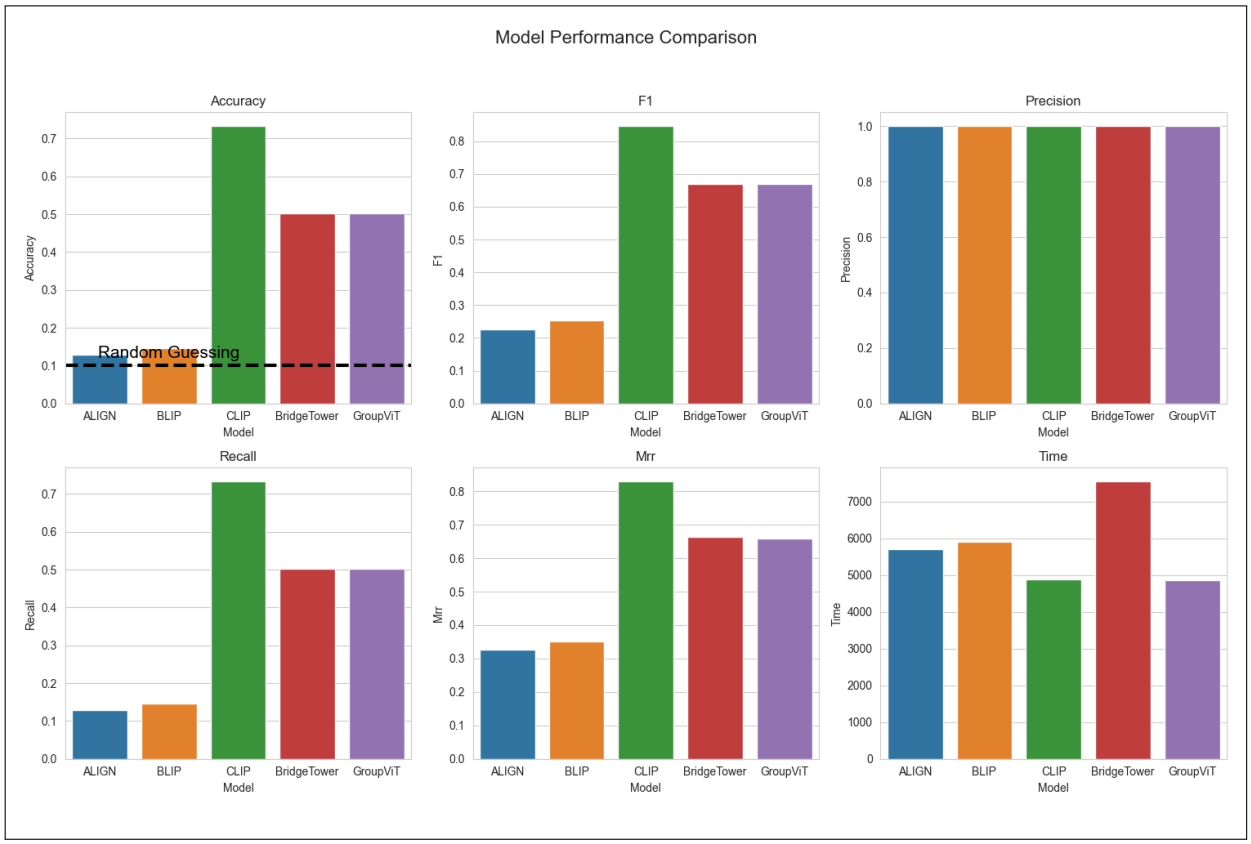
8.2.1 Problems Encountered

During the evaluation process, we encountered unforeseen challenges that hindered the assessment of the **BLIP-2** model. Unfortunately, we faced difficulties related to the installation of **BLIP-2** dependencies and associated libraries. Despite our efforts to troubleshoot and resolve these issues, we were unable to successfully integrate **BLIP-2** into our evaluation environment.

Given the importance of a consistent and reliable evaluation setup, we made the decision to skip the evaluation of **BLIP-2** in this comparative analysis. We acknowledge that this limitation prevents a comprehensive comparison including **BLIP-2**. Our evaluation primarily focuses on the other mentioned models, where we could ensure a stable testing environment and accurate assessment.

8.2.2 Performance Evaluation

Figure 8.7
Metric performances



In the comprehensive evaluation of vision-language models depicted in **Figure 8.2.2**, **CLIP** emerged as the top performer across multiple key metrics. Notably, **CLIP** exhibited exceptional accuracy, achieving high **F1** scores, **Recall** rates, and **Mean Reciprocal Rank** (MRR) scores consistently above 0.7. Furthermore, **CLIP** demonstrated impressive efficiency, showcasing favorable inference times. Notably, **Precision** was uniformly perfect (1.0) across all models, reflecting the models' ability to precisely classify and generate accurate predictions.

The closely matched performance of BLIP and ALIGN, as well as BridgeTower and GroupViT, underscores the comparable capabilities of these model pairs. However, it's worth highlighting that

while BLIP and ALIGN demonstrated similar performance metrics, BridgeTower and GroupViT showcased analogous strengths, especially in terms of precision. Despite their similarities, the inference times varied, with BridgeTower requiring the most.

8.3 Superior Performance of CLIP

Considering the overall exceptional performance of **CLIP**, particularly its consistent excellence in accuracy, **F1**, **Recall**, and **MRR**, coupled with its efficiency in terms of inference time, we made the strategic decision to choose **CLIP** for further steps such as **fine-tuning** and improvement. This selection is based on **CLIP's** comprehensive and robust performance across diverse evaluation metrics, positioning it as a promising candidate for subsequent model enhancements and refinements.

In addition to its performance metrics, the consideration of the number of parameters is pivotal in ensuring the feasibility and computational efficiency of further steps such as **fine-tuning**. The table comparing the number of parameters in **Table 8.1** among the models reveals that **CLIP** strikes a balance between model complexity and resource requirements. While other models like **BridgeTower** have more parameters, **CLIP's** performance efficiency makes it an attractive choice.

CLIP distinguishes itself through its novel dual-encoder architecture, utilizing a **Vision Transformer** for images and a **Transformer-based language model** for text, coupled with **contrastive learning**. This unique design allows CLIP to comprehensively align and understand the relationships between diverse images and text, potentially contributing to its superior performance in vision-language tasks compared to the above-mentioned models.

Chapter 9

Model fine-tuning

9.1 Contrastive and Triplet Losses

Figure 9.1
Contrastive loss training

```
Model CustomCLIPModel started training, device: cuda, loss: contrastive, learning rate:  0.0001

==> Before training results: Val Loss: 1.0229
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: 0.2673, Val Loss: 0.2063
Metrics: accuracy = 0.15, f1 = 0.261, prec = 1.0, rec = 0.15, mrr = 0.317

==> Epoch 2/20 - Train Loss: 0.2045, Val Loss: 0.2040
Metrics: accuracy = 0.1, f1 = 0.182, prec = 1.0, rec = 0.1, mrr = 0.361

==> Epoch 3/20 - Train Loss: 0.2040, Val Loss: 0.2035
Metrics: accuracy = 0.35, f1 = 0.519, prec = 1.0, rec = 0.35, mrr = 0.532

==> Epoch 4/20 - Train Loss: 0.2041, Val Loss: 0.2044
Metrics: accuracy = 0.16, f1 = 0.261, prec = 1.0, rec = 0.15, mrr = 0.421
```

Figure 9.2
Triplet loss training

```
Model CustomCLIPModel started training, device: cuda, loss: triplet, learning rate:  0.0001

==> Before training results: Val Loss: 1.0229
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: 1.0022, Val Loss: 0.9278
Metrics: accuracy = 0.2, f1 = 0.333, prec = 1.0, rec = 0.2, mrr = 0.366

==> Epoch 2/20 - Train Loss: 0.8383, Val Loss: 0.7101
Metrics: accuracy = 0.15, f1 = 0.261, prec = 1.0, rec = 0.15, mrr = 0.363

==> Epoch 3/20 - Train Loss: 0.7851, Val Loss: 0.9795
Metrics: accuracy = 0.0, f1 = 0.0, prec = 0.0, rec = 0.0, mrr = 0.274

==> Epoch 4/20 - Train Loss: 0.7640, Val Loss: 0.7283
Metrics: accuracy = 0.15, f1 = 0.261, prec = 1.0, rec = 0.15, mrr = 0.435

==> Epoch 5/20 - Train Loss: 0.6605, Val Loss: 0.6951
Metrics: accuracy = 0.2, f1 = 0.333, prec = 1.0, rec = 0.2, mrr = 0.403
```

After initiating fine-tuning for **CLIP** using **Contrastive and Triplet loss functions**, we meticulously monitored the model's performance across various metrics. Notably, the outcomes exhibited a nuanced pattern, with metric performance either showing degradation or remaining relatively stable, contingent upon the chosen learning rate. A visual representation of this observation is shown on **Figure 9.1** and **Figure 9.2**. This intricate relationship underscores the sensitivity of the **fine-tuning** process to hyperparameter tuning, especially with respect to learning rates.

9.2 Cross-entropy Loss

Figure 9.3
Cross-Entropy loss with 0.001 Learning Rate

```
Model CustomCLIPModel started training, device: cuda, loss: cross_entropy, learning rate: 0.001

==> Before training results: Val Loss: -69.0733
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: -6345860.7821, Val Loss: -29064971.3333
Metrics: accuracy = 0.11, f1 = 0.182, prec = 1.0, rec = 0.1, mrr = 0.297

==> Epoch 2/20 - Train Loss: -4998836090.4348, Val Loss: -9199435093.3333
Metrics: accuracy = 0.41, f1 = 0.571, prec = 1.0, rec = 0.4, mrr = 0.249
                                .
                                .
                                .
==> Epoch 6/20 - Train Loss: -6060515.4776, Val Loss: -28012212.0000
Metrics: accuracy = 0.11, f1 = 0.182, prec = 1.0, rec = 0.1, mrr = 0.363

==> Epoch 7/20 - Train Loss: -1070460553.7391, Val Loss: -2297825365.3333
Metrics: accuracy = 0.0, f1 = 0.0, prec = 0.0, rec = 0.0, mrr = 0.268
```

Figure 9.4
Cross-Entropy loss with 0.0001 Learning Rate

```
Model CustomCLIPModel started training, device: cuda, loss: cross_entropy, learning rate: 0.0001

==> Before training results: Val Loss: -69.0733
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: -14401.1255, Val Loss: -33259.0070
Metrics: accuracy = 0.0, f1 = 0.0, prec = 0.0, rec = 0.0, mrr = 0.245

==> Epoch 2/20 - Train Loss: -305729.9035, Val Loss: -404187.2000
Metrics: accuracy = 0.1, f1 = 0.09, prec = 1.0, rec = 0.05, mrr = 0.237

==> Epoch 3/20 - Train Loss: -521366.7021, Val Loss: -656628.4750
Metrics: accuracy = 0.2, f1 = 0.33, prec = 1.0, rec = 0.2, mrr = 0.23

==> Epoch 4/20 - Train Loss: -2974468.2000, Val Loss: -3371378.2000
Metrics: accuracy = 0.3, f1 = 0.18, prec = 1.0, rec = 0.1, mrr = 0.174

==> Epoch 5/20 - Train Loss: -1186417.0333, Val Loss: -1412906.0250
Metrics: accuracy = 0.45, f1 = 0.62, prec = 1.0, rec = 0.45, mrr = 0.204

==> Epoch 6/20 - Train Loss: -63406.4032, Val Loss: -102497.0922
Metrics: accuracy = 0.40, f1 = 0.46, prec = 1.0, rec = 0.3, mrr = 0.223

==> Epoch 7/20 - Train Loss: -1671678.6111, Val Loss: -1961251.1750
Metrics: accuracy = 0.37, f1 = 0.4, prec = 1.0, rec = 0.25, mrr = 0.26
```

Figure 9.5
Cross-Entropy loss with 0.00001 Learning Rate

```
Model CustomCLIPModel started training, device: cuda, loss: cross_entropy, learning rate: 0.00001

==> Before training results: Val Loss: -69.0733
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: -850.1132, Val Loss: -1194.2816
Metrics: accuracy = 0.20, f1 = 0.333, prec = 1.0, rec = 0.2, mrr = 0.473

==> Epoch 2/20 - Train Loss: -1405.9185, Val Loss: -1752.5396
Metrics: accuracy = 0.20, f1 = 0.333, prec = 1.0, rec = 0.2, mrr = 0.368
                                .
                                .
                                .
==> Epoch 7/20 - Train Loss: -3534.7351, Val Loss: -3656.5725
Metrics: accuracy = 0.20, f1 = 0.333, prec = 1.0, rec = 0.2, mrr = 0.321
```

Fine-tuning CLIP using **Cross-entropy loss** involved a meticulous exploration of learning rates to optimize the model's performance. When using a learning rate of 0.001, the model exhibited erratic behavior, oscillating back and forth during training, yielding only moderate results. This behavior is observable on **Figure 9.3**. Subsequently, lowering the learning rate to 0.0001 showed improvement, with a discernible trend in the model's learning process, as depicted on **Figure 9.4**. However, it became evident that the model plateaued, settling into a relatively low local optimum.

After the 5th epoch, there was a decreasing trend, culminating in an accuracy of 0.0 by the 10th epoch.

9.3 Cosine Annealing

Figure 9.6
Cross-Entropy loss with Cosine Annealing

```
Model CustomCLIPModel started training, device: cuda, loss: cross_entropy, learning rate: cosine annealing starting with 0.0001

==> Before training results: Val Loss: -69.0733
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: -14394.3037, Val Loss: -33071.2078
Metrics: accuracy = 0.1, f1 = 0.095, prec = 1.0, rec = 0.05, mrr = 0.252

==> Epoch 2/20 - Train Loss: -62595.3227, Val Loss: -100611.4750
Metrics: accuracy = 0.3, f1 = 0.181, prec = 1.0, rec = 0.1, mrr = 0.239
                                    •
                                    •
                                    •
==> Epoch 7/20 - Train Loss: -955219.1347, Val Loss: -1095305.5250
Metrics: accuracy = 0.76, f1 = 0.837, prec = 1.0, rec = 0.72, mrr = 0.291
```

Further reducing the learning rate to 0.00001 proved counterproductive, hindering the model's ability to learn effectively. Metrics appear to be fixed at a single value, as indicated on **Figure 9.5**. To address these challenges, the introduction of **cosine annealing**, along with an appropriate scheduler, provided a significant improvement. The model demonstrated enhanced learning, surpassing the baseline and consistently achieving an **Accuracy** range of 0.76-0.78 across multiple runs with different splits. The main trend is visible on **Figure 9.6**.

9.4 Summary

It's noteworthy that while **Accuracy** saw consistent improvement, the use of **Cross-entropy** loss had implications for the stability of **Mean Reciprocal Rank** (MRR), particularly in the context of ranking. The model's performance in terms of **Accuracy** did not entirely translate to stable **MRR** values, as Cross-entropy primarily focuses on correctly identifying matches without explicitly considering ranking details. This observation emphasizes the need for an understanding of the interplay between loss functions, learning rates, and the specific evaluation metrics employed during **fine-tuning** to achieve optimal performance.

Chapter 10

Project Realization

10.1 Steps of Model Evaluation

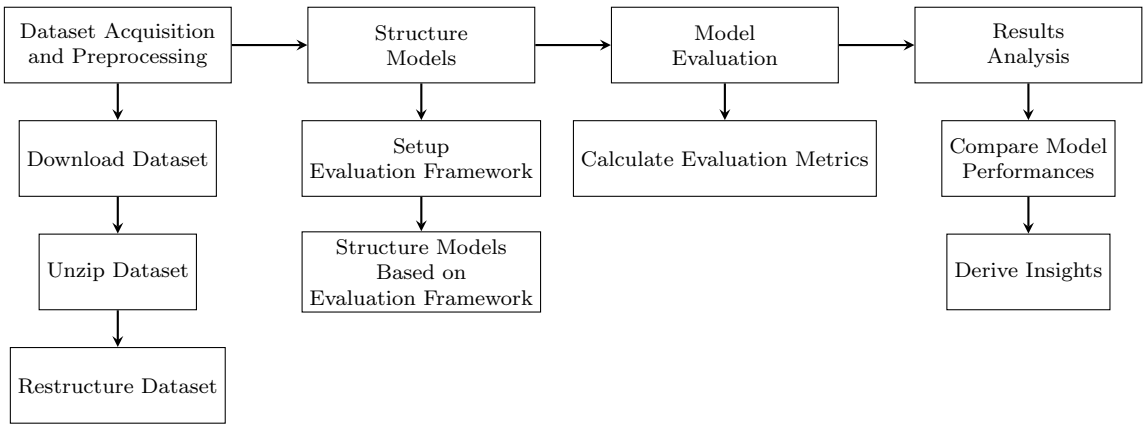


Figure 10.1
Steps of Model Evaluation

This chapter outlines the systematic approach adopted for evaluating multiple models in the context of Visual Word Sense Disambiguation (Visual-WSD). The process encompasses several stages, from dataset preparation to model evaluation and fine-tuning, ensuring a comprehensive assessment of each model's capabilities. The chapter gives an overview of each step outlined in **Figure 10.1**.

10.1.1 Dataset Acquisition and Preprocessing

Our initial step involved the acquisition of the Visual-WSD dataset. The dataset was *downloaded*, *unzipped*, and *restructured* for ease of use. This restructuring included renaming trial images for consistency and parsing text files into a singular ***CSV*** format. The final dataset structure was streamlined to facilitate efficient processing, with a focus on the ambiguous words, their contextual phrases, and candidate images. Then, data loaders were created based on the restructured dataset for both training and validation. Data loaders description can be found in **subsection 7.4.2**.

10.1.2 Structure Models

To maintain consistency in evaluation across different models, we defined so-called **Experimental Evaluation Framework**, described in **subsection 6.2.1**. Such **BaseModel** served as the architectural foundation for all the selected models, that were built and structured upon it.

10.1.3 Model Evaluation

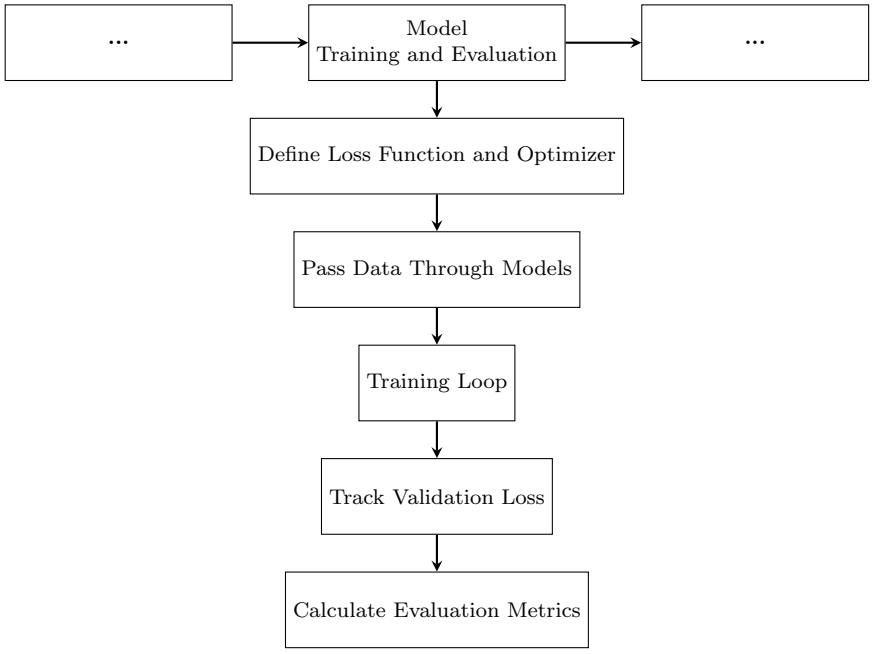
The models were then subjected to an evaluation process. This involved model evaluation and various metrics calculation to assess their performance. Metrics and their application is described in **section 6.3**. The evaluation highlighted each model's ability to correctly interpret and rank the visual context of ambiguous words. Detailed model evaluation step description can be found in **chapter 7**.

10.1.4 Results Analysis

The final stage involved compiling the results from the evaluations. This encompassed analyzing the performance metrics for each model, drawing comparisons, and deriving insights regarding their strengths and weaknesses in the Visual-WSD context. The model comparison is described in **chapter 8**.

10.2 Process of Fine-tuning

Figure 10.2
Adjustments to Model Evaluation Structure for Fine-tuning



Fine-tuning a pre-trained model involves performing specific adaptations to the model in order to make it more suited for a specific task - in our case, Visual Word Sense Disambiguation (Visual-WSD). The fine-tuning step is described in **chapter 9**. It is worth highlighting that our work concentrated on refining a model for the Visual-WSD task while maintaining the original functionalities for which the model was initially designed. Generally, the structure and work schema remain the same, with the addition of fine-tuning process in the model training to it. This involves taking a pre-trained model and further training it on a more specific dataset - Visual-WSD one in our case. Consequently, the fine-tuning process included the following key adjustments:

- **Loss Function and Optimizer:** Selecting and applying an appropriate loss function and optimizer for the fine-tuning task **(section 3.7)**

- **Training Loop:** Training the model on Visual-WSD dataset
- **Parameter Adjustments:** Continuously tracking metrics, making slight adjustments to the model's parameters based on the loss and performance on the validation set **(section 7.7)**

Thus, the overall **Model Evaluation Steps** depicted on **Figure 10.1**, incorporating these adjustments, is depicted as in **Figure 10.2** presented at the beginning of the section. **Model Evaluation** was replaced with **Model Training and Evaluation**, while the stages preceding and following this remained unchanged.

10.3 Reproducibility and Scalability

Ensuring the **reproducibility** of our experiments was a priority. We meticulously *documented* our environment setups, model configurations, and evaluation parameters. This practice was crucial for scientific rigor and facilitated peer reviews and collaborative improvements.

The environment was designed with **scalability** in mind, allowing for the seamless integration of additional computational resources as required by the growing complexity of our models and dataset size.

10.4 Final Product

10.4.1 Introduction

To leverage the capabilities of the project, users are provided with a streamlined command-line interface for initiating models' training and evaluation. The process is straightforward and can be executed on any system where the necessary Python environment and dependencies have been set up. In this section, exemplary execution of *CLIP* model evaluation and training is shown. The same instructions apply for the rest of the models, examined in terms of the project (CLIP, BLIP, BridgeTower, ALIGN, and GroupViT).

10.4.2 Prerequisites

To successfully execute the project, the user must ensure the following setup:

- Ensure that **Python 3** (version 3.10 or higher) is installed
- Install all the required libraries and dependencies as listed in the project's **requirements.txt** file. To achieve this, execute the following command: `pip install -r requirements.txt`

The project script automates the installation and preprocessing of datasets, ensuring that training and evaluation are ready for use, with no further actions needed from the user's side.

10.4.3 Evaluating a Model

Figure 10.3
Example Usage via Command Line: Model Evaluation

```
[15:35:09] [~/Desktop/visual-wsd] [refactoring ✖] ❱❱❱ python3 src/clip_eval.py

Model Clip started evaluation, device: cuda
100%|████████████████████████████████████████████████████████| 625/625 [1:00:27<00:00,  5.8s/it]
accuracy = 0.733
f1 = 0.845
precision = 1.0
recall = 0.733
mrr = 0.829
```

Executing the command displayed on **Figure 10.3** initiates the evaluation process for the model selected by a user on the chosen device, which in this instance is a CUDA-enabled GPU. During this process, the script provides updates on the evaluation's progress and ultimately presents crucial performance metrics for the model, such as accuracy, F1 score, precision, recall, and MRR.

10.4.4 Training a Model

Figure 10.4
Example Usage via Command Line: Model Training

```
[15:34:49] [~/Desktop/visual-wsd] [refactoring ✖] ❱❱❱ python3 src/clip_train.py

Model CustomCLIPModel started training, device: cuda, loss: cross_entropy, learning rate: cosine annealing starting with 0.0001

==> Before training results: Val Loss: -69.0733
Metrics: accuracy = 0.73, f1 = 0.857, prec = 1.0, rec = 0.75, mrr = 0.816

==> Epoch 1/20 - Train Loss: -14394.3037, Val Loss: -33071.2078
Metrics: accuracy = 0.1, f1 = 0.095, prec = 1.0, rec = 0.05, mrr = 0.252
```

The command line depicted on **Figure 10.4** initiates the training of a model of a user's choice. Throughout the training process, the model's parameters are modified according to the designated loss function, which is cross-entropy in this case, and guided by a learning rate schedule that employs cosine annealing. The console output provides immediate feedback by displaying the loss and metrics such as accuracy, F1 score, precision, recall, and Mean Reciprocal Rank (MRR) before training, and for each epoch thereafter.

10.5 Used Tools and Environment

In our project, a variety of tools and technologies were employed to ensure efficient and effective model training, evaluation, and fine-tuning. These tools were selected for their robustness, technical support, and alignment with our project's goals.

Central to our work was the **Python** programming language, renowned for its extensive support in Machine Learning and data processing. Python's rich ecosystem of libraries greatly facilitated our tasks from model development to data analysis.

We heavily utilized **Jupyter Notebook**, an interactive environment that excellently combined code execution with documentation and visualization. This choice was instrumental in making our experimentation process more transparent and accessible, allowing for a seamless blend of coding, visual outputs, and explanatory notes. Cloud platforms such as **Google Colab** and **Kaggle** were employed for their scalable compute resources and high availability. These platforms enabled us to

dynamically allocate resources based on the computational needs of different models and datasets as they provide **GPU** acceleration. In addition to GPUs, Google's Tensor Processing Units **(TPUs)** were tested in our project. TPUs, designed specifically for high-performance machine learning tasks, provided significant boosts in processing speed, particularly beneficial for training large, complex models and handling extensive datasets.

PyTorch framework was chosen for its dynamic computation capabilities. It enabled us to efficiently prototype and iterate over various Neural Network models. The **Transformers** library from **Hugging Face** was another cornerstone, providing us with access to the pre-trained models selected. This library was particularly useful for our project.

Data handling and manipulation were expertly managed using **NumPy** and **Pandas**. NumPy's powerful array manipulation capabilities were used in handling numerical computations, while Pandas offered an intuitive interface for manipulating and analyzing structured data, such as our dataset in *CSV* format.

Visualization tools like **Matplotlib** and **Seaborn** were employed to glean insights from our data and model outputs, allowing us to present our findings in a clear and visually appealing manner. For the image processing aspects of our project, **OpenCV** was the tool of choice, aiding us significantly in manipulating and processing the visual content of our dataset.

Evaluating our models' performances called for precise and reliable metrics, for which we turned to **Scikit-learn**. This library offered a comprehensive suite of tools for Machine Learning, including functions to calculate selected evaluation metrics. To monitor the progress of our model training and evaluation, we integrated **TQDM**, a simple yet effective progress bar library.

To ensure consistency and reproducibility, we utilized virtual environments and containerization tools like **Docker**. These tools helped in creating isolated environments with specific configurations, ensuring that our models ran consistently across different computing platforms.

Lastly, our computational infrastructure leveraged **GPU acceleration**, utilizing **CUDA** for efficient computation and significantly speeding up our model training and evaluation processes. For version control and collaboration, **Git** and **GitHub** were used to efficiently track changes and facilitate collaboration among team members.

Chapter 11

Conclusions and Future Work

11.1 Conclusions

In this thesis, we developed a comprehensive framework for evaluating Visual Word Sense Disambiguation (Visual-WSD) models, a small step in integrating visual contexts with natural language processing. Our approach involved creating a robust evaluation framework, assessing various Deep Learning models, and fine-tuning a model for enhanced performance.

When it comes to **Evaluation Framework**, we successfully established an evaluation framework tailored for Visual-WSD, setting a precedent for future research in this area. This framework not only tested model accuracy but also assessed their ability to interpret complex visual cues in conjunction with textual data.

In regards to **Model Evaluation**, our assessment of existing models revealed significant insights into their capabilities and limitations. We observed that while these models show promise in understanding visual contexts, there are notable gaps in their ability to consistently interpret nuanced language.

Furthermore, we conducted comparisons among various top-tier models in the field of image-captioning, analyzing why a particular architecture yields specific performance outcomes for the Visual-WSD challenge. Through this analysis, it was established that the **CLIP** (Learning Transferable Visual Models From Natural Language Supervision) model, as described by *Radford et al.* [9], significantly surpasses other chosen multimodal models such as BLIP, ALIGN, BridgeTower, and GroupViT in terms of performance, providing a consistency across all evaluated metrics, including F1-Score, MRR, Precision, Recall, and Accuracy.

Lastly, our efforts in **Fine-tuning** a particular model showcased the significant influence of this process in Visual-WSD. This process involved adjusting various parameters and algorithms to better align the model with the unique characteristics of our dataset. This fine-tuning not only improved the accuracy of the model in interpreting visual-textual data but also shed light on the intricate balance between generalization and specificity in Machine Learning models.

11.2 Future Work

Due to time and resources constraints, numerous adaptations, tests, and experiments are slated for future exploration. This is particularly relevant since experiments with Deep Learning models are often time-intensive, taking hours to complete a single experiment. This issue is compounded by the limited availability of essential resources, such as GPU. Future work will focus on a more thorough investigation of specific mechanisms and architectural specifications, exploring new approaches to the existing problem of Visual Word Sense Disambiguation.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[2] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021.

[5] Sunjae Kwon, Rishabh Garodia, Minhwa Lee, Zhichao Yang, and Hong Yu. Vision meets definitions: Unsupervised visual word sense disambiguation incorporating gloss information. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1583–1598, Toronto, Canada, July 2023. Association for Computational Linguistics.

[6] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[7] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[8] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models, 2016.

[9] 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, 2021.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[11] Alessandro Raganato, Iacer Calixto, Asahi Ushio, Jose Camacho-Collados, and Mohammad Taher Pilehvar. SemEval-2023 task 1: Visual word sense disambiguation. In Atul Kr. Ojha, A. Seza Doğruöz, Giovanni Da San Martino, Harish Tayyar Madabushi, Ritesh Kumar, and Elisa Sartori, editors, *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2227–2234, Toronto, Canada, July 2023. Association for Computational Linguistics.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[13] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and X. Wang. Groupvit: Semantic segmentation emerges from text supervision. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18113–18123, 2022.

[14] Xiao Xu, Chenfei Wu, Shachar Rosenman, Vasudev Lal, Wanxiang Che, and Nan Duan. Bridgetower: Building bridges between encoders in vision-language representation learning, 2023.

[15] Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions, 2023.

Poznan University of Technology
Faculty of Computing and Telecommunication
Institute of Computing Science

Typeset using LaTeX