Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Self-Supervised Multi-Modal Text-Image Retrieval Methods to Improve Human Reading

Master Thesis
at Research Group Language Technology, LT
Prof. Dr. Chris Biemann

Department of Informatics
MIN-Faculty
Universität Hamburg

submitted by
Florian Schneider
Course of study: Computer Science
Matriculation number: 7089752
on
27.07.2021

Examiners: Prof. Dr. Chris Biemann
Dr. rer. nat. Özge Alaçam

Abstract

In primary school, children's books, as well as in modern language learning apps, multi-modal learning strategies like illustrations of terms and phrases are used to support reading comprehension. Also, several studies in educational psychology suggest that integrating cross-modal information will improve human reading.

In artificial intelligence, methods or models are called multi-modal or cross-modal if they jointly process and relate data of different modalities simultaneously. This thesis focuses on current self-supervised multi-modal methods for text-image retrieval to improve human reading within a language learner scenario. Specifically, state-of-the-art visio-linguistic transformers, which recently gained a lot of interest and outperformed traditional multi-modal text-image retrieval methods, are considered.

In our language learner scenario, the aim is to support a user's reading comprehension by providing context-specific visual cues for arbitrary text on demand. This work answers initial research questions and proposes solutions to engineering challenges to realize the scenario in a practical use case – to be exact, towards employing state-of-the-art multi-modal text-image retrieval models to improve human reading.

First, a new Wikipedia-based multi-modal dataset (WISMIR) is collected to assess the performance of current text-image retrieval models on complex textual data. Despite the computationally determined low performance on the WISMIR test set, two user studies where human raters assessed the quality of the top-ranked retrieved images suggest that the evaluated models are generally suitable for the language learner scenario. In the last part of the thesis, preceding results and findings are incorporated to develop a "real-time" capable text-image retrieval system powered by current visio-linguistic transformers. The user interface, which is realized through a browser plugin, represents a proof-of-concept solution for the practical use-case of the language learner scenario that meets the essential requirements and implements the basic functionality.

However, effectively improving human reading within a real-world language learner scenario is a comprehensive and challenging task that requires much future research, experiments, and engineering.

Contents

1 Introduction

As human beings, we navigate through the world in a manifold cross-modal manner. Most of the time, this is so natural that we do not notice how intertwined our brain processes sensory inputs of different modalities. Especially when we make use of one of the most outstanding skills of humans, the natural language, we almost always experience it in a multi-modal fashion. Since we were babies, we learned our native language by combining our parents' words and visual hints. For example, think of parents showing a round object to their child while repeatedly uttering the word "ball". Sooner or later, the baby will have an abstract image of a ball in her mind whenever she hears or reads the word. In primary school, children's books, as well as in modern language learning apps, like Babble[1] or Duolingo[2], this multi-modal learning strategy continues as illustrations of terms and phrases are used to support reading comprehension. Even in higher education, e.g., during a Master's degree in any STEM field, we learn complex phenomena or algorithms with the help of visualizations, e.g., by sketching functions or drawing graphs that abstract the problem.

In artificial intelligence, methods or models are called multi-modal if they jointly process and relate data of different modalities at the same time. This work focuses on visio-linguistic models, which simultaneously process textual and visual data, that is, written language and images.

During the past few years, there were significant breakthroughs in computer vision as models are constantly improving in pixel-level object detection, classification of thousands of object or attribute categories, or estimating 3D human poses (Kirillov et al. 2020; Güler et al. 2018; Anderson et al. 2018). Also, in natural language processing, especially with the dawn of transformers, models are increasingly capable of understanding semantics and dependencies even in long contexts (Vaswani et al. 2017; Devlin et al. 2019; Brown et al. 2020), which is essential downstream tasks. This progress in uni-modal methods also led to a great leap forward in multi-modal visio-linguistic models, which are starting to leverage the power of transformers to operate on textual and visual data (Qi et al. 2020; Chen et al. 2020; Li et al. 2020).

Besides their competitive performance, another significant advantage of both uni-modal and multi-modal transformers is that they can be trained in a self-supervised fashion. This means that, unlike traditional models, transformers do not require manually labeled training data but can be trained on a vast amount of unlabeled "in the wild" data crawled from the internet.

One of several tasks where these models pushed the boundaries is multi-modal text-image retrieval. The goal of this task is to find the best matching images according to a textual query, usually from a large pool of images. To do so, models compute a similarity score between each image and the query and finally return the resulting list in descending order so that the best matching image with the highest similarity score is ranked first. Computing these similarities obviously requires that the models understand the textual input as well as the visual input and, further, that they can compare or relate the two modalities.

This thesis presents initial research towards leveraging state-of-the-art visio-linguistic transformers for text-image retrieval to improve human reading within a language learner scenario described in the following.

1. https://babbel.com/
2. https://duolingo.com/

1.1 Language Learner Scenario

Multiple studies in educational psychology suggest that integrating cross-modal information will improve learning to read (Ecalle et al. 2009; Dalton and Grisham 2011; Hahn et al. 2014; Gerbier et al. 2018; Kabooha and Elyas 2018; H. Xie et al. 2019; Albahiri and Alhaj 2020). Hence, in our language learner scenario, the aim is to support a user's reading comprehension by providing context-specific visual cues for a text on demand. This text, referred to as context, can consist of a sentence or a paragraph as it appears on any website. Moreover, the user chooses a word, referred to as focus, with which she has particular difficulties understanding it. The visual cues that should support her reading comprehension are images that best match the context as well as the focus. A visio-linguistic text-image retrieval transformer is employed to discover these images from a large image pool.

1.1.1 Practical Use Case

A practical use case of the scenario could be realized by a browser plugin, where a user inputs the focus word and the context and is presented with images that support her understanding of the focus within the context. This browser plugin should ideally fulfill the following requirements:

- installing without dependencies
- easy to use
- "real-time" capable, i.e., only short latency to receive the best matching images for a query
- highlighting of the region that represents the focus word within an image

1.2 Research Questions

Realizing the language learner scenario in a practical use case, which in the case of this work should be an easy-to-use browser plugin, involves multiple steps of research and engineering, which are the central topic of this thesis. Hence, this works seeks to answer the following research questions, which subsume the overall aim of employing state-of-the-art multi-modal text-image retrieval models to improve human reading.

1.2.1 Research Question 1 (RQ1)

The most popular training and evaluation datasets for current models applied on text-image retrieval are MS COCO (Lin et al. 2014) and Flickr30k (Young et al. 2014; Plummer et al. 2015). To the best of my knowledge, every state-of-the-art model utilized for this task was trained (at least partially) on one or both of these datasets. Both COCO and Flickr30k were created by crowdsourcing workers with the task to find short, simple, and descriptive captions for images carefully selected from Flickr[3].

In the previously introduced language learner scenario, however, there are no constraints on the textual input. Hence, the sentences or paragraphs are presumably more complex than the captions from COCO or Flickr30k, which is why it is expected that state-of-the-art text-image retrieval models will perform poorly on more complex textual data.

This results in the first research question of this thesis, which is formulated as:

> *How do state-of-the-art multi-modal transformers perform in text-image retrieval with complex and lengthy textual queries?*

3. https://flickr.com/

1.2.2 Research Question 2 (RQ2)

Typically, current models are evaluated on text-image retrieval with the popular Recall@K – short R@K – metric with $K = 1, 5, 10$. This metric ranges from 0% to 100% and measures the fraction of text-image pairs or samples of the test set, for which the model retrieved the ground-truth image for the corresponding textual query in the top-K ranks. One problem with this method of measuring the performance of text-image retrieval models for real-world systems is that the R@K metric is binary. For example, if a model retrieved the ground-truth image for a particular text-image pair in the 6th rank, the R@1 and R@5 metric for this sample would be 0%, and only R@10 would be 100%.

However, this is often a misleading result, as the probability that the first few ranked images are similar, and therefore all relevant, is very high in a large pool of images. Especially in the language learning scenario, this metric is not satisfactory for evaluating the suitability of models since it might even be beneficial for the user's learning process to see different images of the same thing.

Therefore, alternative ways have to be found to answer the second research question stated as:

> *Are state-of-the-art multi-modal transformers applied on text-image retrieval suitable for our language learner scenario?*

1.2.3 Research Question 3 (RQ3)

The input to traditional text-image retrieval methods or state-of-the-art models utilized for this task is a single sentence, usually referred to as the query. However, for our language learner scenario, it is required that the textual query is not only a sentence but a pair consisting of a sentence or paragraph (the context) and a word (the focus) contained in that context. Moreover, the images retrieved by the multi-modal text-image retrieval method, which should be developed within the scope of this thesis, have to match the context and the focus at the same time. To the best of my knowledge, no current model or system employed for text-image retrieval supports queries consisting of a context and a focus.

Hence, the third research question evaluated in this work is as follows:

> *How can textual queries consisting of a context and a focus contained therein be supported in multi-modal text-image retrieval methods so that the retrieved images correspond to both the context and the focus?*

1.2.4 Research Question 4 (RQ4)

To find the best matching images for a given textual query from a pool of images, a similarity function between the query and every image in the pool must be computed. Multi-modal text-image retrieval models represent such a similarity function by a complex neural network. The problem with this is that evaluating a single similarity score for a text-image pair requires a large amount of computation. Hence, finding the best matching image from a large pool of images becomes infeasible for "real-time" applications – even on a modern GPU-powered system.

This problem gives rise to the fourth research question to be answered by this thesis, which is stated as follows:

> *How to leverage state-of-the-art multi-modal transformers in a practical application, i.e., a "real-time" text-image retrieval system with a large pool of images?*

1.3 Thesis Overview

In this chapter, an introduction to the general topic and the research questions of this thesis was given. The remaining thesis is structured as follows: In Chapter 2, the fundamentals of uni-modal natural language processing and computer vision models are concisely explained. These uni-modal models are the basis for the multi-modal models, which are focussed and utilized throughout this thesis. The principles and different architectures of these multi-modal models are described and compared in Chapter 3. In Chapter 4, a brief overview of existing multi-modal datasets is given. Next, in Chapter 5, WISMIR, a dataset collected within the scope of this work and based on Wikipedia data, gets introduced and compared to popular existing datasets. This dataset is required to evaluate the first research question (RQ1). Further, several experiments conducted to evaluate RQ1 are described in Chapter 6. Chapter 7 presents a new tool developed to conveniently conduct user studies that seek to evaluate text-image retrieval performance by human raters. In Chapter 8, two user studies to evaluate RQ2 are introduced, and their results are reported. A multi-modal text-image retrieval system (MMIRS) was developed to evaluate RQ3 and RQ4 and is introduced in Chapter 9. Moreover, MMIRS aims to realize the practical use-case of the language learner scenario introduced earlier. Finally, the research questions are answered in summary with their solution approach, and the thesis is concluded in Chapter 10. Additionally, this chapter provides ideas for future work.

2 Related Work

This work considers multi-modal models, i.e., models learning from textual and visual data simultaneously. Since those models are based on current models from Natural Language Processing (NLP) and Computer Vision (CV), this section briefly introduces these uni-modal model architectures essential for this thesis. State-of-the-art multi-modal models utilized in this work are introduced separately with more details in Chapter 3.

2.1 Language Models

As this work's title suggests, the primary model architecture considered throughout this thesis, is the popular and successful transformer architecture introduced by Vaswani et al. 2017. This type of model emerged from a long path of research, which is briefly described in the following.

Since language or textual data is sequential, i.e., the words (and the characters that make up the words) in a text make sense only if they are read sequentially from left to right or from right to left. Hence, earlier successful models utilized to solve natural language tasks, i.e., recurrent neural networks or RNNs (Rumelhart et al. 1985), processed textual data also sequentially. An RNN is very similar to classical feed-forward neural networks with the extension that the output of the last hidden layer – also known as the "hidden state" – is fed as additional input to the current step. This "hidden state" vector serves as a "memory" of what was processed previously. However, these models have difficulties with long-term dependencies in a text and "forget" contextual information if it is too far away from the currently processed token.This is due to the model's architecture, which only allows processing the current token in the input sequence and the last "hidden state" vector, which contains aggregated information from all previous steps. For long input sequences, it is infeasible to compress all essential information from all previous tokens in a single vector.

More sophisticated sequential models designed to overcome this long-term dependency problem – technically also known as "vanishing gradient problem" – are LSTMs (Hochreiter and Schmidhuber 1997) or GRUs (Cho et al. 2014). These models can memorize context much longer than vanilla RNNs via multiple internal neural networks, called gates, that learn which information to forget or keep in the hidden state vector. Despite the models' enhanced context memorization capabilities, they still have problems solving tasks with too long sequences, where contextual information is essential. For example, in machine translation, where an input sequence in the source language has to be mapped to an output sequence in the target language, the models' performance decreases with the input length.

Models to solve sequence-to-sequence tasks usually follow an encoder-decoder architecture (Sutskever et al. 2014). In this architecture, the encoder model, e.g., an LSTM, processes the input token sequence to produce a continuous intermediate representation in which essential information from the input sequence is encoded. The decoder model – usually also an LSTM or GRU – then processes this intermediate representation to produce the first token in the output sequence. In the next step of the decoder, the hidden state from the last step and the previously generated token are the used to predict the next output token. This continues until the decoder predicts an *[END]* token, indicating that the output sequence is generated completely.

However, since the building blocks of in these encoder-decoder architectures are usually RNNs, the performance of these models still decreases with the length of the input sequence. A breakthrough in sequence-to-sequence tasks was the attention mechanism introduced by Bahdanau et al. 2015 and Luong et al. 2015. In this extension to traditional encoder-decoder architectures, supplementary context vectors are stored for each step of the encoding process. These context vectors, which are basically the "hidden state" vectors of the encoder steps, are used to provide the decoder with information directly from the input sequence and tell it which parts of the input are essential for the current decoding step – which is why this mechanism is called attention. Besides several methods to realize the attention mechanism, self-attention implemented in transformer models described in the following is the most successful.

2.1.1 Transformer Models

In the following, the fundamental concepts of the transformer architecture introduced by Vaswani et al. 2017 are described in more detail since it covers the primary constituents of state-of-the-art uni-modal as well as multi-modal transformer models. Note that the following explanations of the self-attention mechanism and the transformer model, in general, are inspired by blog posts from Peter Bloem[1] and Jay Alammar[2].

Self-Attention

One of the key differences of self-attention to the attention mechanism and encoder-decoder architectures involving RNNs is that the transformer model processes all tokens in the input sequence at once and not sequentially token after token. This solves the long-term dependency issues and allows a much more efficient computation during training and inference time via parallelization.

Basically, self-attention is a learned function that transforms every input vector $x_i \in \mathbb{R}^d$ to an output vector $y_i \in \mathbb{R}^d$. The fundamental concept of the self-attention function is that each of the input vectors is transformed by three learned linear functions, namely the key, the value, and the query function. With these three functions, the self-attention mechanism, which can be understood as a soft-dictionary function, can control how much and which information of each input vector is contained in each of the resulting output vectors. Or in other words, every output vector y_i is computed from a weighted sum over every input token, where the learned key-, value-, and query-transformation matrices $\mathbf{W}_k$, $\mathbf{W}_v$, and $\mathbf{W}_q$, define these weights as shown by Equation 2.1

$$y_i = \sum_j \text{softmax}\left(\frac{q_i^T k_j}{\sqrt{k}}\right) v_j \tag{2.1}$$

where $k_i = \mathbf{W}_k x_i$; $v_i = \mathbf{W}_v x_i$; $q_i = \mathbf{W}_q x_i$; and $j \in [0, N]$ with N being the lengths of the input sequence. Note that all the matrices are of size $d \times d$. The softmax function is applied for numerical stability so that the weights of the input vectors contributing to the output vector sum up to one. Further, the dot-product between the query and key vectors are normalized to counteract eventual vanishing gradient issues introduced by the softmax function.

Instead of iterating over the input sequence, all operations are applied at once via large matrix multiplication to achieve maximum computational efficiency through parallelization, resulting in the well-known Equation 2.2 and visualization shown in the left part of Figure 2.1.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \tag{2.2}$$

1. http://peterbloem.nl/blog/transformers
2. https://jalammar.github.io/illustrated-transformer/

where Q, K, and V are matrices containing all stacked query, key, and value vectors computed from the input vectors.

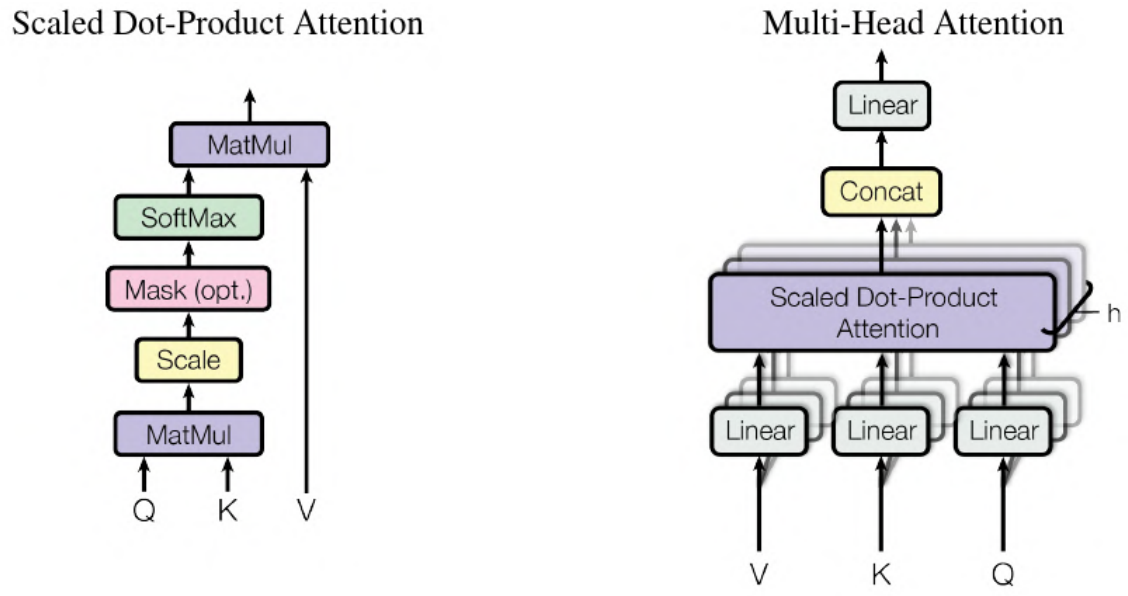


Figure 2.1: Left: The computation graph of the self-attention mechanism; Right: Schematic overview of the high-level computation graph of multi-headed attention. The image is taken from Vaswani et al. 2017.

Transformer Encoder Model

The complete transformer model introduced by Vaswani et al. 2017 and shown in Figure 2.2 is also an encoder-decoder model. However, the models considered throughout this thesis are only built from the encoder part, so the decoder part is not covered here. Further, the training of the model and task which it was designed to solve, i.e., machine translation is also not of interest for this work and therefore skipped in this explanation.

The transformer encoder shown in the left part of Figure 2.2 is made up of several different layers, described briefly in the following.

First, continuous representations of the input tokens are computed by the embedding layer. Since the input tokens are not processed sequentially but all at once, and the output of the dot-product and any other subsequent operation is invariant to the order of the elements, the token embeddings are combined by what is referred to as positional encodings. As the name suggests, these encodings contain information on the positions of the tokens in the input sequence. Next, the token embeddings are forwarded through a Multi-Head Attention layer shown on the right in Figure 2.1. This basically combines multiple self-attention layers with separate key, query, and value transformation matrices. Each of the self-attention layers is referred to as an attention head and is responsible for discovering different semantic dependencies between the input tokens. The outputs of the multi-head attention layer are combined with their corresponding inputs via a residual connection and normalized afterward. Layer normalization and residual connections are common strategies to improve the training of the large models via gradient-descent methods by preventing too small or large gradients. Finally, the normalized outputs are forwarded separately through the same fully connected feed-forward layer with ReLU activation to project them back to the input dimension d and normalized again afterward.

To sum up, a transformer encoder is a sophisticated transformation producing rich a representation per input token, containing information about dependencies between every token of the input sequence.

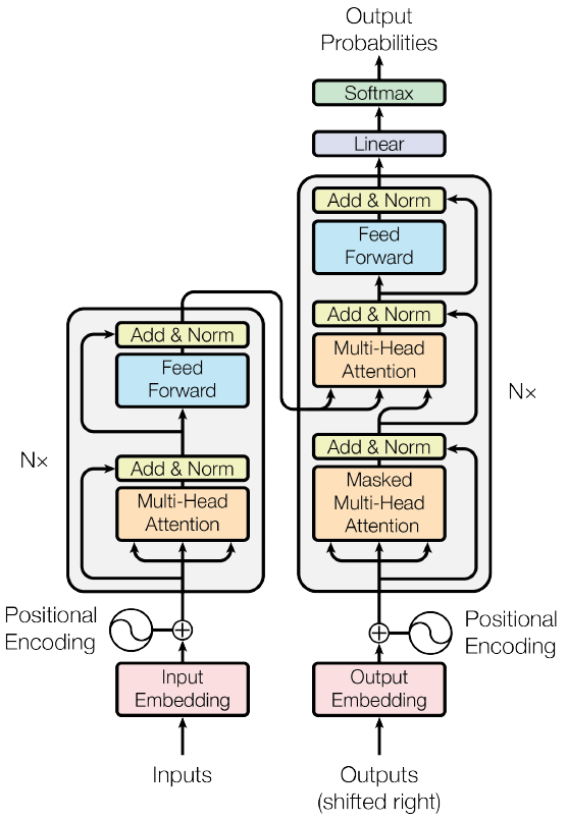


Figure 2.2: Schematic architecture of the transformer model introduced by Vaswani et al. 2017. The image is taken from the paper.

Bi-directional Encoder Representations from Transformers (BERT)

This transformer-based model revolutionized the field of natural language processing as it outperformed previous models in a total of eleven tasks – often by a large margin – which is why basically every succeeding language model is based on the ideas introduced in the BERT paper by Devlin et al. 2019.

Since the model is made up of stacked transformer encoders discussed in the previous section, the authors' main contribution and key to the success of BERT is not of architectural nature. Further, it was not the first (transformer-based) language model pre-trained to produce representations on large corpora. However, one major drawback of standard conditional language models, i.e., models trained to predict the current token given all previous tokens in a sequence, is that they can only be trained in one direction. This is because predicting a token in the sequence would be trivial if the models were fed the complete input sequence bi-directionally.

With BERT, which employs bi-directional self-attention and processes all tokens of a sequence simultaneously, the authors introduced a simple yet effective technique to train the model on massive datasets by two unsupervised tasks. The first task is called Masked Language Modelling (MLM), where 15% of the tokens in the input sequences are masked out, and the model has to predict the original tokens. The second task is Next Sentence Prediction (NSP), where BERT is trained to understand the relationship between sentences. This is achieved by feeding the model two concatenated sentences, which are two actually consecutive sentences in 50% of the time or two random sentences in the other 50% of the cases. During pre-training, BERT is trained on both of the tasks simultaneously by minimizing the combined loss function.

The model's strong performance is based on the contextualized token representations learned via unsupervised pre-training on a huge unlabeled corpus containing about 3B

words. When the BERT is applied on a downstream task, a new model instance is initialized with the pre-trained weights and then fine-tuned on the specific task. During fine-tuning, the weights are updated by training BERT on task-specific labeled data in a supervised fashion. Depending on the downstream task, the token embeddings computed by the model are employed differently or fed through an additional layer also trained during fine-tuning.

Pre-training a model's parameters on massive datasets and then re-using these parameters to fine-tune a different model instance on a different task is often referred to as transfer learning, a common technique also applied in Computer Vision.

2.2 Computer Vision Models

The most common type of models employed in Computer Vision – regardless of the task – are Convolutional Neural Networks or CNNs (LeCun et al. 1998). However, CNN only describes a type of neural network, the exact architecture of the employed model depends on the task so that there exists a large variety of model architectures.

In this work, CNNs for object detection and classification are of interest since they extract the visual features necessary for multi-modal models described in Chapter 3. Further, their outputs are used in the multi-modal information retrieval system MMIRS developed in this work and described in Chapter 9. To be specific, the object detection and classification network considered and utilized throughout this work is a Faster R-CNN with ResNet-101 model (Ren et al. 2016; He et al. 2016; Anderson et al. 2018), briefly described in Section 2.2.2.

To understand this complex and sophisticated models, it is essential to know the fundamentals of CNNs described in the next section.

2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks try to mimic the human visual system (HVS) by modeling the receptive fields in our eyes and throughout the visual cortex with mathematical functions called filters. Note that CNNs can be used for various tasks like and even solve NLP tasks or generate images. Since this is irrelevant for this work, it is ignored, and only CNNs for object detection and classification are considered in the following. Usually, the input is an image, and the output is a list of labels of the detected objects in the image.

Like HVS, the structure of CNNs is hierarchical. On the lower levels, general visual features like contrasts and edges are recognized, which are summarized in the higher levels to more specific features like patterns or simple objects. The final layers of the CNNs classify these features, e.g., to recognize complex objects, characters, or structures in X-Ray images. The number of layers or depth of the CNN depends on the design of the researches. Usually, the performance of a CNN increases with the number of layers. However, so does the number of parameters and, therefore, the training time.

A CNN for object detection or classification consists of three different layer types stacked alternately, namely convolution layers, pooling layers, and fully connected layers. Since fully connected layers are very common in neural networks generally, they are not introduced here. However, convolution layers and pooling layers are fundamental for every CNN and briefly described in the following.

Convolution Layers

These layers are where the name of the CNN comes from and the main operation of these networks. As mentioned earlier, in CNNs, the receptive fields of the HVS are modeled by filter functions to detect useful visual features, e.g., for object detection. Typical filter functions like Gaussian or Gabor filters are generally continuous functions in $\mathbb{R}^n$. In CNNs,

however, the inputs are rasterized and discrete images with fixed and finite dimensions. Hence, also the filters are rasterized and discrete functions with typical dimensions of $k \times k \times c$, where usually is $k = \{3, 5, 7\}$ and c is the number of channels or the depth of the kernel. The number of channels c in a kernel is always equal to the number of channels of the image the kernel is applied on. For example, an RGB image has $c = 3$ channels, and filters applied on the image are also of shape $k \times k \times 3$. The filter functions are applied on an input image via 2D convolution as described by Equation 2.3.

$$\mathrm{O}[i, j] = \sum_{c=0}^{C} \sum_{u=0}^{k} \sum_{v=0}^{k} \mathrm{H}[u, v, c]\ \mathrm{I}[i - u, j - v, c] \tag{2.3}$$

where H is a discrete $k \times k \times c$ filter kernel; I is a discrete $m \times n \times c$ image or input volume; and c denotes the depth or the channels of the kernel or image.

Note that there are several essential things – like the padding of the image or the stride of the filter – to consider during convolution, which are not discussed here for brevity. In the following discussions, it is assumed that the inputs I are padded so that the resulting output O is of the same dimension – also known as "same" padding.

Since a single filter can only detect a single feature, many different kernels are applied per layer, each producing its own output or feature map, usually stacked into one matrix. So, for example, if 16 different filters are applied on an input image of size 28×28, the resulting feature map is of size $28 \times 28 \times 16$. Note that filters applied on this feature map would be of size $k \times k \times 16$. An example of this convolution process with the corresponding input and output sizes is shown in Figure 2.3 – the max-pooling operation can be ignored for now and is described next.

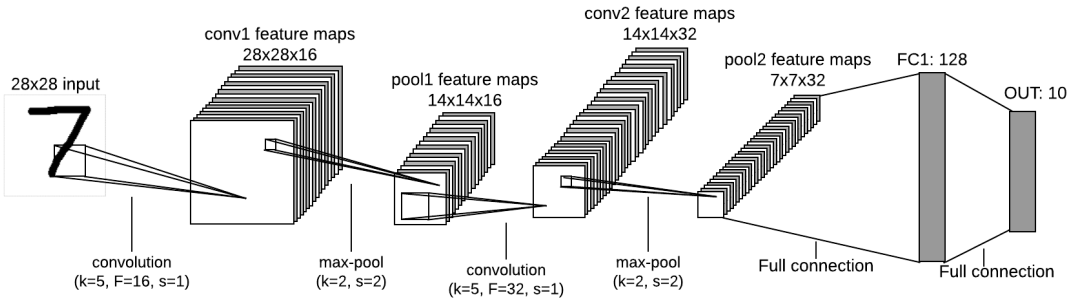


Figure 2.3: Example of a concrete CNN to classify handwritten digits. The input and output sizes, as well as the parameters of the convolution, pooling, and FC layers, are described. The parameter k describes the filter or pooling window size, s the stride, and F the number of kernels applied on the input volume. Source: https://www.easy-tensorflow.com/images/cnntext.png

The fundamental part responsible for the widespread success of CNNs is that these filter functions are not pre-defined and fixed but are learned by the model from the training process. In other words, the parameters of a CNN are the weights of the discrete filter kernels employed in the convolution layers of the network. Similar to other types of neural networks, these weights are trained via gradient-based optimization and backpropagation. Training details, however, are not relevant here and are therefore not covered here.

Pooling Layers

This layer is used to decrease the complexity of CNNs by reducing the dimensions – not the channels – of the feature maps. Further, pooling layers summarize a feature map, which can be thought of as creating more complex features from simple features.

Same as a filter kernel, a pooling kernel or window also is of fixed size $k \times k \times c$. However, instead of applying a convolution, the pooling window is slid over the input volume and summarizes the underlying values. Further, the window is slid over the input with a stride of $s \geq 2$. That is, the window coordinates are incremented by s, which has the wanted effect that the output dimensions are smaller than the input dimensions (see Figure 2.3). There are multiple methods to pool a feature map, but the most common is "max-pooling".

Note that pooling layers do not have any learnable parameters, i.e., they are not trained but constant.

2.2.2 Object Detection And Classification Models

The Faster R-CNN (Ren et al. 2016) model is a deep CNN to efficiently and accurately detect and classify objects in an image. In this work, a version with ResNet-101 (He et al. 2016) as the backbone model is used to extract visual features from images. To understand Faster R-CNN with ResNet-101 (Anderson et al. 2018), its predecessors Fast R-CNN Girshick 2015 and R-CNN (Girshick et al. 2014), and the general concept of object detection and classification networks are briefly introduced first.

General Object Classification Concept

To classify objects in an image, first, the regions the objects are contained in an image have to be located or detected. Second, features that adequately describe these regions of interest (ROIs) have to be extracted. In the third and final step, the ROI features are processed by a classifier to predict the corresponding class label.

R-CNN

The R-CNN model consists of three components, each responsible for one of the steps mentioned previously. Hence, the model is not trainable end-to-end, which is one of several significant limitations. Further, R-CNN is memory and computational very expensive compared to state-of-the-art models or Faster R-CNN and takes almost 50s to detect and classify objects in an image on average. This has two reasons: First, the ROI detection component, based on a traditional CV algorithm, extracts about 2000 region proposals and is relatively slow. Second and mainly, the feature extraction component has to be applied individually to every region. In the third component of R-CNN, the extracted ROI features are used to predict the respective class labels with an SVM classifier. Further, a regressor model predicts improved bounding boxes for the ROIs.

Fast R-CNN

As the name suggests, the successor of R-CNN improves the processing speed (and also the accuracy) of the model. In Fast R-CNN, an end-to-end trainable CNN, its predecessor's expensive feature extraction component is substituted and improved. Further, the SVM classifier of R-CNN is replaced by fully connected (FC) layers, and the object labels are predicted via softmax over the class vocabulary. Instead of extracting the features of each ROI individually, Fast R-CNN computes a global feature map of the image from which the features of the individual regions are cropped. Since the FC layers to predict the object classes and improved bounding boxes require fixed-size input vectors, a technique called ROI-pooling produces these vectors from the differently sized ROI feature maps. The difference between ROI-pooling and max-pooling is that the output size is constant and independent of the input size. These fixed-sized ROI feature vectors are fed through two separated paths of FC layers to predict the object labels and to regress improved bounding boxes.

Compared to R-CNN, the Fast R-CNN model is much faster and reduced test time for one image by a significant factor to about 2.3 seconds on average while being more accurate, too.

Faster R-CNN

From the 2.3 seconds needed by Fast R-CNN to process one image, about 2 seconds are taken to find ROIs in the image. In Faster R-CNN, its predecessors' traditional and expensive approach is replaced by a so-called Region Proposal Network (RPN). The exact details are not essential for this work, but conceptually the RPN, which is also a CNN, finds ROIs not on the original image but in the feature maps computed by the feature extraction CNN. Like in Fast R-CNN, an ROI-pooling layer produces ROI feature vectors of fixed size, which are forwarded through the FC layers to predict class labels and to regress bounding boxes for the detected objects.

The RPN is very efficient compared to the traditional approach and takes only about 100ms, which reduces the overall time to process an image to 0.3s. However, the accuracy of Faster R-CNN does not improve compared to Fast R-CNN.

Faster R-CNN with ResNet-101

To extract the global feature map from the raw input image, any CNN can be employed. In Faster R-CNN, the authors tested the two object detection networks ZF (Zeiler and Fergus 2014) and VGG-16 (Simonyan and Zisserman 2015), which differ in depth, speed, and accuracy. Both of these models were pre-trained on ImageNet (Deng et al. 2009) for object classification. As suggested by its name, the Faster R-CNN with ResNet-101 (He et al. 2016) model employs a ResNet-101 model instead, which is much deeper has 101 layers. Hence, it achieves better evaluation scores on different object detection benchmarks – at the cost of longer training and inference time.

The authors of Anderson et al. 2018 additionally pre-train the complete model on the Visual Genome dataset (see Section 4.1.3) for Object Detection and extended it so that it predicts object labels with additional attribute labels. Some attribute labels are, for example, "green", "long", or "smiling" and are combined with the corresponding object classes.

The resulting ROI features are considered information-rich and are therefore used in several downstream tasks or as input to other (multi-modal) models.

3 Multi-Modal Visio-Linguistic Models

There exists a large variety of multi-modal models for visio-linguistic tasks. From an architectural perspective, these models can be subdivided into two groups: non-transformer-based models and transformer-based models.

3.1 Non-transformer Models

Non-transformer-based models first process each modality separately by using Recurrent Neural Networks (RNNs) like LSTMs (Hochreiter and Schmidhuber 1997) or GRUs (Cho et al. 2014) for the text modality, and Convolutional Neural Networks (CNNs) like Faster-R-CNN with ResNet-101 (Ren et al. 2016; He et al. 2016; Anderson et al. 2018) or VGG (Simonyan and Zisserman 2015) for the imaging modality. After that, the representations or embeddings for textual tokens and image regions are in the same n-dimensional vector space. In this common vector space, it is possible to fuse the modalities and compute, for example, similarities between tokens and image regions, word-region-alignments (WRA), or loss functions utilizing information from both modalities.

Typical tasks of these models are visual question answering (VQA), image captioning, or text-image or image-text retrieval. The models indeed reach reasonable evaluation scores but can seldom catch up with transformer-based models, described in Section 3.2. State-of-the-art models of this type are, for instance, CAAN (Zhang et al. 2020), PFAN (Y. Wang et al. 2019), SCAN (Lee et al. 2018), SCG-Net (Shi et al. 2019), or DenseCap (Johnson et al. 2016). It should be noted that this type of model is commonly thought to be overtaken by transformer models, which represent the large majority of models published in or after 2020.

Nevertheless, it is worth mentioning that the general architecture of multi-modal non-transformer models is much more efficient while still producing fairly comparable. CAAN, for instance, reaches Recall@10 scores on Flickr30k image-retrieval of 87.9 while Unicoder-VL (G. Li et al. 2020), a state-of-the-art multi-modal transformer, reaches 94.9 R@10 on the same dataset. However, CAAN is the clear winner when it comes to performance or efficiency: to compute the similarity of a text-image pair, CAAN needs about 45μs, where Unicoder-VL needs 0.5s – a difference in 5 orders of magnitude.

3.2 Multi-Modal Transformers

During the last few years, there were significant breakthroughs in various computer vision (CV) tasks and models (Kirillov et al. 2020; Güler et al. 2018) as well as in the field of natural language processing (NLP). Especially with the recent dawn of the so-called transformer language models, the models are increasingly capable of understanding text's semantics (Brown et al. 2020; Devlin et al. 2019; Yang et al. 2019). This progress in uni-modal models also led to a great leap forward in multi-modal visio-linguistic models (VLMs), as scientists are starting to leverage the power of transformer models to work with text and images simultaneously (Chen et al. 2020; Li et al. 2020; Qi et al. 2020; L. H. Li et al. 2019; Su et al. 2020).

The input to multi-model transformers are textual tokens of a sentence and visual tokens of an image or their dense vector embeddings. The textual token embeddings are usually contextualized word-embeddings computed by pre-trained transformer language models

such as BERT (Devlin et al. 2019). Since sentences or text, in general, is sequential data, the textual tokens get additively combined with positional embeddings that reflect the ordering of words. The visual token embeddings are typically region-of-interest (ROI) features from the output of one of the last layers before the classification head in pre-trained bottom-up object detection and classification networks such as Faster-R-CNN with ResNet-101 (Ren et al. 2016; He et al. 2016; Anderson et al. 2018; Z. Yu et al. 2020). Because it is beneficial for the model to know the spatial relationships of the visual tokens, the location and the size of the bounding box the token originates from are also encoded into the visual token.

3.2.1 Early-Fusion Models

In early-fusion models such as UNITER (Chen et al. 2020), OSCAR (Li et al. 2020), ImageBERT (Qi et al. 2020), VisualBert (L. H. Li et al. 2019), or VL-BERT (Su et al. 2020), tokens of both modalities and special tokens indicating the modality form the input to the network. Multiple self-attention heads in the transformer encoder layers introduced by Vaswani et al. 2017 and described in Section 2.1.1, then produce a joint-representation of both modalities. That is, fine-grained word-region-alignments (WRA) of the tokens of the input text and the visual tokens of the input image, crucial for achieving state-of-the-art results.

Despite their remarkable evaluation scores across all visio-linguistic tasks, early-fusion models are not applicable in real-world information retrieval systems with large image pools. This is because computing the global similarity of a query sentence and an image requires forwarding the query tokens with the visual tokens through the complete transformer stack. In order to retrieve the best matching image from a large pool, this process has to be repeated for every image in the pool, and we cannot pre-compute any of the outputs. This would require tremendous computational power and therefore makes early-fusion models infeasible in "real-time" information-retrieval systems.

3.2.2 Late-Fusion Models

As opposed to early-fusion models, where both modalities get fused already in the first self-attention head, in late-fusion models, the textual and visual modalities first get forwarded through separate transformers. Later, the output of the textual transformer and the output of the visual transformer get fused depending on the model's specific implementation. For example, LXMERT (Tan and Bansal 2019) and VilBERT (Lu et al. 2019) compute the fused cross-modality output with a third cross-modal Transformer that takes the separate and uni-modal transformers' outputs as inputs. Other late-fusion models specially designed to solve multi-modal retrieval tasks like TERN (Messina et al. 2021) and TERAN (Nicola et al. 2020) use a more straightforward and computationally efficient way. In TERN, the two CLS token embeddings of the output of the separate transformers are representing the complete image and sentence, respectively. These tokens are then used to compute the cosine similarity that expresses the global similarity of a sentence and an image. On the other hand, TERAN uses all token embeddings in the output of the uni-modal transformers to produce a fine-grained word-region-alignment matrix, where each cell is the cosine-similarity of the respective textual and visual token. The resulting fine-grained word-region similarity matrix gets pooled to compute a global similarity score of the input sentence and image.

The significant advantage of late-fusion models over early-fusion models in information-retrieval systems is that the output embeddings of the uni-modal transformers can be pre-computed and indexed. In multi-modal text-image retrieval systems, with a large pool of images, this saves enormous amounts of time and computational power. The pre-computed image embeddings can be reused, and only the query sentence embedding

has to be computed to measure the similarity between the query sentence and all images in the pool. Especially in TERN and TERAN, where the multi-modal fusion is not a complex transformer, this attempt leads to short latency and makes real-world multi-modal information-retrieval systems possible.

3.3 TERAN

In this section, the TERAN model gets introduced. This multi-modal transformer is a late-fusion model, meaning that it has two separate transformer-encoder stacks (see Section 2.1.1) – one for the textual and one for the visual data. It is specially designed for text-image or image-text retrieval, i.e., to compute global similarity scores of sentences and images. The code of the model, written with PyTorch[1], and pre-trained weights are available on GitHub[2].

The inputs to the text transformer-encoder stack are the token embeddings of a pre-trained BERT tokenizer model. To be specific, the "bert-base-uncased" from the popular huggingface[3] library is used.

The inputs to the vision transformer encoder stack are region-of-interest (ROI) features extracted with the bottom-up object detection and classification network Faster R-CNN. To encode the region's position, the region feature vector gets combined with a 5-dimensional vector encoding position and area of the region through a simple fully-connected layer.

A schematic overview of the model is shown in Figure 3.1

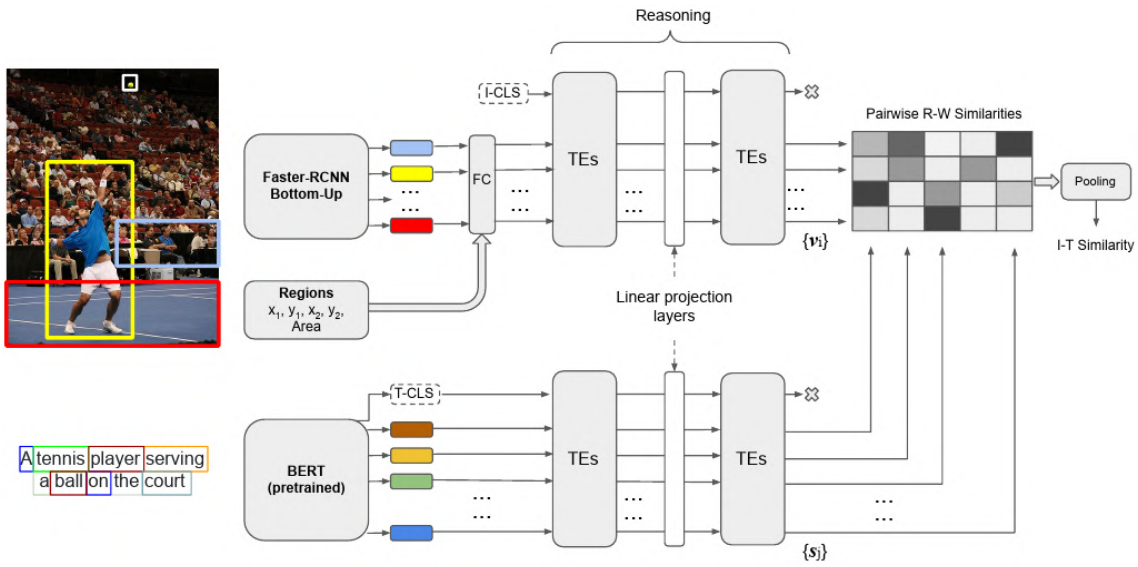


Figure 3.1: An overview of the TERAN model architecture taken from the paper.

3.3.1 Fusion of the modalities

To combine or fuse the textual and visual modalities, i.e., to compute joint-representations, linear layers in the separate transformer stacks project the embeddings into a common 1024-dimensional vector space. After that, the embeddings get forwarded further through the respective transformer stacks, which output an embedding for each input token. In these embeddings, the spatial relations and the semantic structure of the visual and the textual input are encoded, respectively. Fine-grained word-region-alignments are computed via

1. https://pytorch.org/
2. https://github.com/mesnico/TERAN
3. https://huggingface.co/

cosine similarity of the textual and visual embedding vectors and are used to construct an alignment matrix. Each cell of this matrix represents the similarity of a textual and visual token. To compute the global similarity of the original image and sentence, the alignment matrix gets pooled. The paper's authors evaluated multiple pooling strategies and came up with "*max-over-regions sum-over-words (MrSw)*" pooling to achieve the best results.

To express this formally, we define the word-region-alignment matrix as

$$\mathbf{A} \in \mathbb{R}^{|I| \times |S|} \tag{3.1}$$

where $|I|$ is the number of region features in the input image I and $|S|$ is the number of token features in the input sentence S.

The cells of $\mathbf{A}$, i.e., the cosine-similarities of the visual regions and textual tokens are defined as

$$\mathbf{A}_{i,j} = \frac{\mathbf{v}_i^T \mathbf{t}_j}{|\mathbf{v}_i||\mathbf{t}_j|} \tag{3.2}$$

where $\mathbf{v}_i \in I$ and $\mathbf{t}_j \in S$.

The global similarity of an image and a sentence through *MrSw* pooling of $\mathbf{A}$ is defined as

$$\Phi(S, I) = \sum_{j \in |S|} \max_{i \in |I|} \mathbf{A}_{ij} \tag{3.3}$$

3.3.2 Training of the model

The model is trained in an end-to-end fashion entirely via hinge-based triplet loss. This loss is a contrastive or ranking loss function, which gets explained in the following. As opposed to typical loss functions for classification tasks like (binary) cross-entropy loss or softmax loss, ranking loss functions are used to train networks with the objective to predict or compute a distance between two input samples.

Triplet-Loss Function

To compute the triplet-loss in general, we first need a notion of a distance function $\Phi(a, b)$. Additionally, we require three samples, which we refer to as "anchor", "positive", and "negative" samples, or s_a, s_p, and s_n. These samples can be randomly drawn or selected according to specific constraints from the dataset. Finally, we need to define a margin m that guarantees a certain minimum distance between s_a and s_n. Now, the objective of the network is to compute pairwise distances so that $\Phi(s_a, s_n) >= \Phi(s_a, s_p)$

Visually, this principle can be easily understood from Figure 3.2, where an example anchor, positive and negative sample are shown in a 2D space with Euclidean distance.

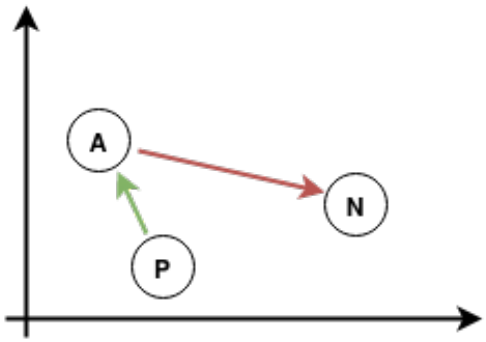


Figure 3.2: A schematic overview of the triplet-loss function. The objective of this loss function is that the model decreases the distance of the anchor sample A and the positive sample P and increases the distance of the anchor sample A and the negative sample N.

Formally, the triplet-loss function, that networks seeks to minimize, is defined as

$$\mathcal{L}(s_a, s_p, s_n) = max(0, m + \Phi(s_a, s_p) - \Phi(s_a, s_n)) \tag{3.4}$$

In other words, the objective is that the distance between the anchor and the negative sample is larger than the distance between the anchor and the positive sample plus the margin.

By analyzing the loss function, it can be noted that three different situations can happen depending on the three samples: Firstly, so-called "easy-triplets", where $\Phi(s_a, s_n) > \Phi(s_a, s_p) + m$, i.e., the negative sample already has a sufficient distance to the anchor sample concerning the distance of the anchor and the positive samples. For "easy-triplets", the loss is zero, and the model's weights do not need to be updated.

Further, there are "hard-triplets", where $\Phi(s_a, s_n) < \Phi(s_a, s_p)$, i.e., the negative sample is closer to the anchor than the positive sample. For "hard-triplets", the loss is positive and greater than the margin, which requires the model to update the parameters significantly.

Finally, there are the "semi-hard-triplets", where $\Phi(s_a, s_p) < \Phi(s_a, s_n) < \Phi(s_a, s_p) + m$, i.e., the negative sample is more distant to the anchor than the positive sample but the minimum-margin distance is not met. For "semi-hard-triplets", the loss is also positive but smaller than for "hard-triplets". The essential point to understand is that the model learns most from "hard-triplets".

This principle is visualized in Figure 3.3, where the anchor and positive samples are fixed, while the negative sample can lay in three different regions, creating either "hard", "semi-hard", or "easy" triplets.

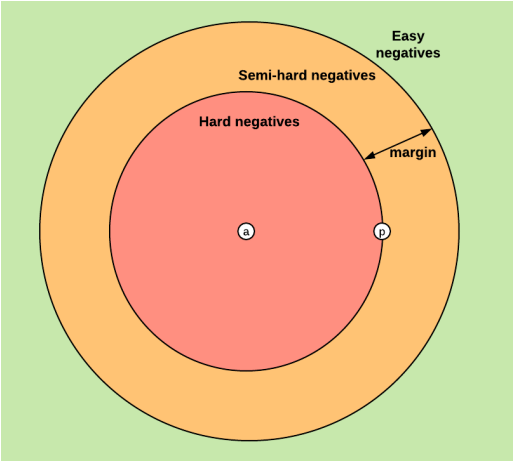


Figure 3.3: A schematic overview of the three different types of negative samples that can originate from the definition of the triple-loss function defined in Equation 3.4. Note that the anchor sample s_a and the positive sample s_p are fixed and depicted as a and p in the image. Source: https://omoindrot.github.io/triplet-loss

Triplet-loss in TERAN

In the case of TERAN, the required distance function is the global text-image-similarity $\Phi(S, I)$ defined in Equation 3.3 Since TERAN deals with samples of two modalities, the notion of the triplet-loss function needs to be slightly extended.

$$\mathcal{L}(I, S) = max(0, m + \Phi(I, S) - \Phi(I, S')) + max(0, m + \Phi(I, S) - \Phi(I', S)) \tag{3.5}$$

where S' is a hard-negative textual sample for the image I, I' is a hard-negative visual sample for the sentence S, and (I, S) is a positive pair.

These hard-negative samples are drawn per training batch B and not from the full training set for performance reasons as follows:

$$I' = \underset{I'' \neq I \in B}{\operatorname{argmax}} \Phi(I'', S) \tag{3.6}$$

$$S' = \underset{S'' \neq S \in B}{\operatorname{argmax}} \Phi(I, S'') \tag{3.7}$$

The triplet-loss function employed in TERAN consists of two basic triplet-loss functions added together. In the first function, the anchor sample is an image, while the positive

and negative samples are sentences. In the second function, it is the other way round: the anchor sample is a sentence, and the positive and negative samples are images. By adding the two segments in the final loss function, the model is forced to learn joint representations for the textual and visual modalities.

Training Data

The authors of TERAN provide two models, with "MrSw" pooling (see Equation 3.3) employed as global text-image similarity function. The models are solely trained on either COCO 2014 (see Section 4.1.2) or Flickr30k (see Section 4.1.1) training sets, respectively.

3.3.3 Precomputation of Embeddings

One significant advantage of late-fusion models and especially in TERAN is the possibility to precompute the outputs or embeddings of the separate transformer-encoder stacks. This is particularly profitable in a "real-time" information retrieval system like MMIRS, powered by TERAN (see Chapter 9). For MMIRS, a "real-time" text-image retrieval system, the embeddings of every image in the image pool of the system got computed before runtime by forwarding it through TERAN's visual transformer-encoder stack. After precomputing, the embeddings are persisted on disk. At runtime or inference time, only the query embeddings have to be computed by forwarding the textual input through the respective transformer. The best matching images are retrieved by computing the alignment matrices of the query embeddings and all precomputed image embeddings, followed by pooling the matrices to obtain the similarity scores of the images and the query, and finally, sorting the scores in descendant order. To make this precomputation of embeddings possible, the codebase of TERAN had to be modified since the authors did not provide this functionality out-of-the-box. To be precise, the TERAN model code had to be disentangled in order to bypass the visual transformer and to persist the outputs of the visual transformer, i.e., the visual embeddings for later use. Further, a new PyTorch Dataset primitive was implemented to load the persisted embeddings from disk and directly serve them, together with the query embedding to the similarity function. More implementation details can be read in Chapter 9.

3.4 UNITER

In the following, UNITER (Chen et al. 2020), an early-fusion transformer model, gets described. As the model's written-out name, "**UN**iversal **I**mage-**TE**xt **R**epresentation Learning", suggests, it is designed to create universal multi-modal embeddings, usable for different language- and vision-tasks. UNITER is based on the popular textual transformer model BERT's (Devlin et al. 2019) ideas and architecture. It employs self-attention-heads to compute word-region-alignments between tokens in sentences and regions in an image. Before the model can process the visual and textual input tokens, they are transformed by an Image Embedder and a Text Embedder, respectively. The Image Embedder's inputs are 2048-dimensional feature vectors computed by a Faster R-CNN (Ren et al. 2016) and a 7-dimensional vector that encodes the image-region position in the source image. The two vectors are then combined through a fully connected layer. The Text Embedder takes token embeddings from a pre-trained BERT tokenizer of the input sentence and sums them with their respective positional embedding. Note that the output embeddings of the Text Embedder and the Image Embedder are normalized via Layer Normalization (Ba et al. 2016) to reduce the training time of UNITER.

Sophisticated pre-training on a combination of popular multi-modal text-image datasets and the model's architecture are the reasons for the outstanding performances in current visio-linguistic tasks. UNITER beats all its competitors in Visual Question Answering

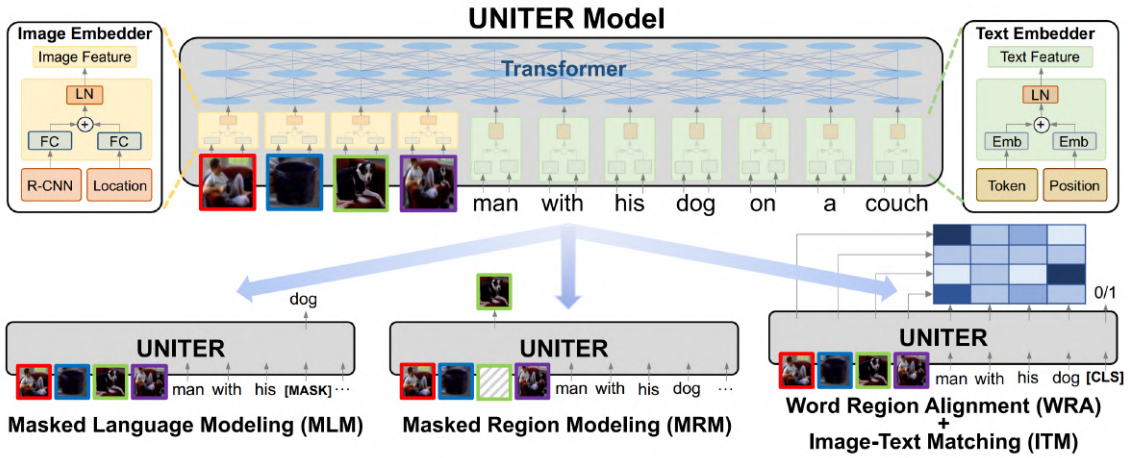


Figure 3.4: A visual overview of the UNITER architecture. taken from the official GitHub repository https://github.com/ChenRocks/UNITER

(VQA) (Antol et al. 2015), Visual Commonsense Reasoning (VCR) (Zellers et al. 2019), Natural Language for Visual Reasoning (NLVR) (Suhr et al. 2017; Suhr et al. 2018), Visual Entailment (SNLI-VE) (N. Xie et al. 2019), Text-Image and Image-Text retrieval on COCO and Flickr30k, Referring Expression Comprehension (L. Yu et al. 2016; Kazemzadeh et al. 2014).

3.5 Summary

In this chapter, different types of visio-linguistic multi-modal models were introduced. Their general architecture can be subdivided into two groups: traditional models and transformer models, subdivided further into early-fusion and late-fusion transformers. In visio-linguistic benchmarks, the state-of-the-art transformers outperform traditional models in terms of evaluation metrics of the respective tasks. However, a significant drawback, especially of late-fusion multi-modal transformers, is their computational complexity, which leads to slow inference times and makes them unsuitable for "real-time" multi-modal information retrieval systems. Nevertheless, early-fusion transformers allow pre-computation of textual or visual embeddings, which saves an enormous amount of computation at inference time, when, e.g., the similarities between a textual query to all images in a large pool have to be computed. Therefore, TERAN, a multi-modal early-fusion transformer, is focussed and utilized throughout experiments conducted and the text-image retrieval system developed for this thesis.

4 Existing Multi-Modal Datasets

Training multi-modal models designed to solve visio-linguistic tasks requires datasets containing textual and visual data. Multiple datasets are available with different sizes and content, all of them containing text-image pairs. Typically, one sample consists of a description or caption together with the corresponding image. There are also datasets including additional data such as fine-grained labeled regions or questions about the image's content and answers. Further, there are multi-modal datasets designed for specific visio-linguistic tasks like VQA (Antol et al. 2015), GQA (Hudson and Manning 2019), NLVR2 (Suhr et al. 2018), SNLI-VE (N. Xie et al. 2019) or VCR (Zellers et al. 2019), which are usually released together. Since this thesis is concerned primarily with the text-image retrieval task, which requires only text-image pairs, other task-specific datasets are not covered in this work.

Section 4.1 and Section 4.2 give an overview of existing multi-modal datasets for text-image retrieval.

4.1 Popular Datasets

Within this thesis, datasets utilized for pre-training or fine-tune visio-linguistic transformers on text-image retrieval are focussed. These datasets have in common that they are designed by researchers and handcrafted typically by crowdsourcing workers with particular tasks.

In the following, a brief overview of suitable and commonly-used datasets is given.

4.1.1 Flickr30k

First, it is worth mentioning that there are two datasets referred to as Flickr30k and are often used interchangeably within the context of multi-modal text-image retrieval.The original Flickr30k (Young et al. 2014) is the basis for the Flickr30k Entities dataset (Plummer et al. 2015) and holds approx. 32K photographs of everyday activities, events, and scenes taken from Flickr [1]. The captions are created via crowdsourcing, where each image is described by five different annotators, resulting in about 160K text-image pairs. The captions, therefore, vary in length and specificity.

Flicker30k Entities builds upon Flickr30k and extends it with approx. 244K co-reference chains that identify and link entities among the same image's captions and a total of about 275K bounding boxes to locate the image regions in which the entities are depicted.

More details and statistics about the Flickr30k dataset can be found in Section 5.3.

4.1.2 COCO 2014

MS COCO 2014 or simply COCO (Microsoft Common Objects in COntext) (Lin et al. 2014) is a well-known dataset originally thought for object detection and segmentation of everyday-objects in natural, non-iconic images. The dataset contains about 123K carefully selected mostly non-iconic images from Flickr with five descriptive captions each, resulting in approx. 616K text-image pairs. Additionally, a total of approx. 2.5M object instances of 91 common object categories are annotated with masks and bounding

1. https://www.flickr.com

boxes within the images. Within the scope of multi-modal image-text retrieval methods, typically, only the text-caption pairs are of interest.

The dataset was collected in multiple stages, utilizing over 70K working hours of human workers from Amazon's Mechanical Turk.

More details and statistics about the COCO dataset can be found in Section 5.3.

4.1.3 Visual Genome

Visual Genome (Krishna et al. 2017) is a complex and large dataset built for general visual understanding and grounding of visual concept to language. The dataset consists of multiple components rather than simple text-image pairs: region descriptions, objects, attributes, relationships, region graphs, scene graphs, and question-answer pairs. The dataset was collected and verified entirely by human workers from Amazon Mechanical Turk[2] and contains[3] about 108K images, 5.4M region descriptions, 1.7M visual question answers pairs, 3.8M object instances, 2.8M attributes, and 2.3M relationships between objects. The images are collected from an intersection of MS-COCO (Lin et al. 2014) and YFCC-100M (Thomee et al. 2016).

Global scene descriptions, i.e., complete sentences that describe the whole image, are generated from multiple image region descriptions using predefined templates.

4.1.4 SBU Captions

The SBU Captions (Stony Brook University) dataset comes in two versions: the original (Ordonez et al. 2011), which is the basis for the later generalized version (Kuznetsova et al. 2013). To create the original dataset, a huge number of Flickr photos were gathered along with their captions. The collection was then filtered to remove captions that do not refer to or describe the image's content, do not have a sufficient length, or do not indicate a visible spatial relationship between the entities in the image. This filtering process resulted in a dataset containing 1M photos with high-qualitative descriptive captions.

In the generalized version of SBU Captions, the images' captions are programmatically processed to reduce the information gap between the image's visual content and their description. For example, the captions should not contain entities that are not visible in the image or contain information that only the photographer could know. The authors propose several methods to generalize the captions, which result in different outcomes, all contained in the generalized SBU Caption dataset.

4.1.5 Conceptual Captions

The Conceptual Captions dataset (Sharma et al. 2018) is a very large-scale and large-variety dataset. It contains approx. 3.3M image-caption pairs from billions of websites, processed by a very sophisticated filtering and transformation pipeline that makes heavy use of state-of-the-art machine learning powered tools to purge the vast amount of input data. The captions originate from alt-text HTML attributes of image-tags from the websites. They are programmatically transformed, and named-entities are substituted by their respective hypernym nodes from a very-large Knowledge Graph to achieve clean, fluent, and informative descriptions of the images.

4.1.6 Focused Datasets

In this thesis, the Flickr30k and COCO datasets are focused for the following reasons: These two datasets are the most commonly used datasets for pre-training and fine-tuning

2. https://www.mturk.com/
3. https://visualgenome.org/

multi-modal models on the text-image retrieval task. To the best of my knowledge, every recently published model is trained and evaluated on one or both datasets or on data that includes Flickr30k and COCO. Further, this work focuses on the TERAN model due to its suitability for "real-time" information retrieval systems. The TERAN models introduced and published by Nicola et al. 2020 are trained solely on either Flickr30k or COCO and evaluated on both.

4.2 Wikipedia-based Datasets

Unlike the popular datasets mentioned in the previous section, the datasets described in the following contain "in the wild" data from Wikipedia. That is, they consist of non-constrained and heterogeneous samples. They are non-constrained because the captions and the images are not the outcomes of a particular (crowdsourcing) task, i.e., there are no constraints besides the terms of use of Wikipedia. They are heterogeneous since the data was randomly crawled from Wikipedia and does not follow any pattern, as opposed to, e.g., COCO or Flickr30k images and captions, which got carefully collected according to several sophisticated rules. Hence, these datasets of interest for our language learner scenario (see Section 1.1), where a user should be able to visualize arbitrary sentences, e.g., from blogs, news websites, or Wikipedia.

Another difference between Wikipedia-based datasets and the datasets from Section 4.1 is that the images of the former are used to support the corresponding text. In the other datasets, however, it is the other way round: The texts are created for the images, which is similar to image-captioning tasks. The Wikipedia-based data is more similar to the language learner scenario because the goal there is to support language learners by providing visual cues for a given text.

4.2.1 WikiCaps

The WikiCaps (Schamoni et al. 2018) dataset is designed to improve the multi-modal translation of image captions from Wikipedia. The dataset contains about 3.8M images of Wikipedia and Wikimedia articles with their respective English captions created by Wikimedia Commons users. It also contains additional 1000 multi-lingual and parallel image-caption pairs in German-English, French-English, and Russian-English, which are not of particular interest for this work.

As shown in Figure 4.1, WikiCaps is – due to its size – made up of a much more diverse vocabulary and contains many more tokens compared to COCO and Multi30k (Elliott et al. 2016) (a multi-lingual Flickr30k alternative).

Therefore, WikiCaps was chosen as the basis for the new WISMIR dataset collected within the scope of this thesis and described in Chapter 5.

4.2.2 ImageCLEF

This dataset (Tsikrika et al. 2011) is the primary resource for the ImageCLEF Wikipedia Image Retrieval task from 2010 and 2011. It consists of $237,434$ images from Wikipedia, with their user-generated captions in English, French, or German, the articles the images appear in, and some low-level visual features. All data originates from Wikipedia dumps from 2009. The number of English-only captions and their respective images is $70,127$.

Due to its rather small size compared to WikiCaps and the same source of data, i.e., Wikipedia, ImageCLEF is not further analyzed and considered in this thesis.

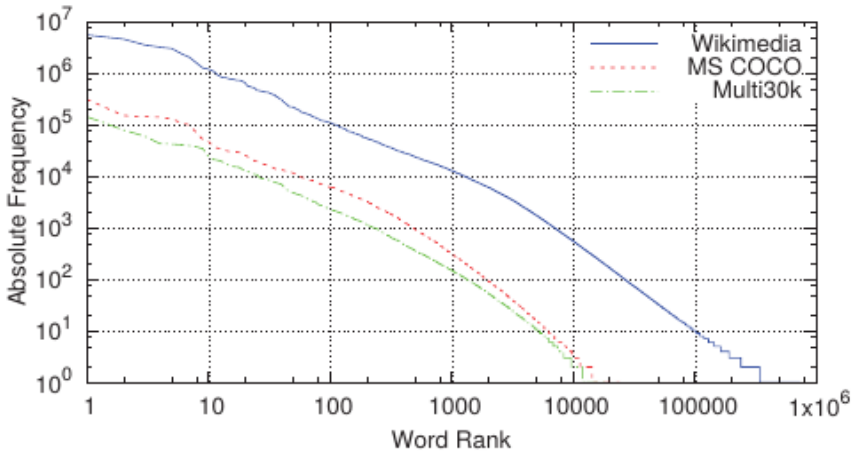


Figure 4.1: Lower-cased word frequency against word rank in the MS COCO, Multi30k and WikiCaps dataset. Source: Schamoni et al. 2018

5 WISMIR Dataset

This thesis aims to answer the research question of how good current text-image retrieval models perform on data containing more complex textual data than the training dataset commonly utilized to train these models (see Section 1.2). Hence, a dataset that fulfills those constraints and suitable for text-image retrieval had to be found. As described in Section 4.2.1, the WikiCaps dataset is a promising candidate since it contains non-constrained caption-image pairs from Wikipedia. However, only samples containing captions of a certain length and divergence from the captions of popular datasets described in Section 4.1 are of interest. To ensure that the new dataset consists only these samples, an ETL pipeline tool to filter WikiCaps was developed. The obtained subset, utilizable for our experiments, is referred to as WISMIR (**Wi**kiCaps **S**ubset for **M**ulti-Modal **I**mage **R**etrieval).

In Section 5.1, the tool gets introduced, and in Section 5.2, the data collection process gets elucidated.

5.1 ETL Pipeline Tool

The authors of WikiCaps only provide a tab-separated file containing the Wikimedia file IDs of the image and the respective captions together with a Perl script to download the images serially. To make the data more accessible, an efficient Python application that realizes an ETL pipeline was developed. The tool is designed to be flexible, i.e., the single steps can be customized by specifying the behavior in a configuration file. The default

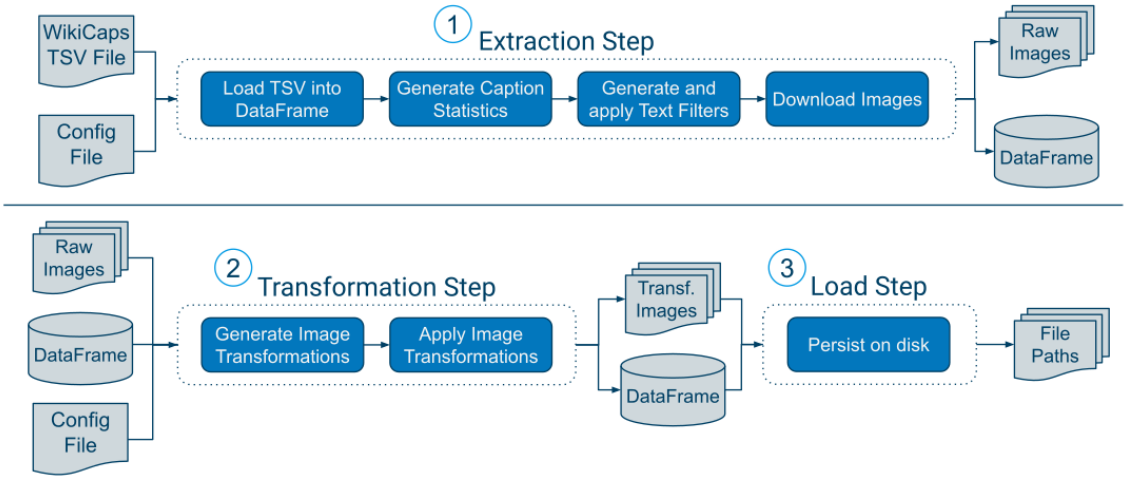


Figure 5.1: Default steps of the ETL pipeline tool to collect the WISMIR dataset.

steps are shown in Figure5.1 and described in the following.

In the extraction step of the pipeline, first, the provided WikiCaps TSV file gets loaded into a pandas [1] DataFrame. Next, different corpus statistics, e.g., the average number of tokens or the ratio of noun tokens, for each caption or row in the DataFrame are generated in parallel and appended to the DataFrame.

Three different frameworks are available to collect the statistics: spaCy[2], NLTK[3], and

1. https://pandas.pydata.org/
2. https://spacy.io/
3. https://www.nltk.org/

Polyglot[4]. This is because each framework uses different models for tokenization, segmentation, part-of-speech tagging, and named-entity-recognition and will therefore produce (slightly) different outputs.

After the statistics generation, the tool parses the configuration file to extract the user-defined filter specifications. These filter specifications consist of a column name and an interval described by a minimum and maximum value. The DataFrame containing the caption-based statistics is then filtered by checking if the number contained in the respective columns is in the specified range. If the input is within the range of min-max interval, it is kept, otherwise discarded. After filtering, the DataFrame gets shortened to a specified maximum number of samples. Finally, the corresponding images of the remaining rows get downloaded in parallel.

In the next pipeline step, the transformation step, the tool parses the configuration file to extract image transformation specifications. Afterward, the tool applies each transformation to each downloaded image in parallel.

In the final pipeline step, the load step, the final DataFrame containing all the statistics and links to the transformed images gets persisted on disk. Note that the DataFrame, log files, and images already get persisted between the substeps to be safe in the case of unexpected failures.

Thanks to the clearly defined interfaces between each (sub)step of the pipeline, it is also possible to execute the (sub)steps isolated from each other. Therefore, the tool can be used to collect statistics for any DataFrame containing a captions column or to transform any set of images according to specified image transformations.

5.2 Dataset Collection

Using the developed ETL tool described in the previous section, two initial versions of WISMIR were collected.

The second version was collected after observing the poor performance of a TERAN model trained and evaluated on WISMIR v1 (see Section 6.2) to assess if more data would improve the model's evaluation results.

5.2.1 WISMIR v3 Collection

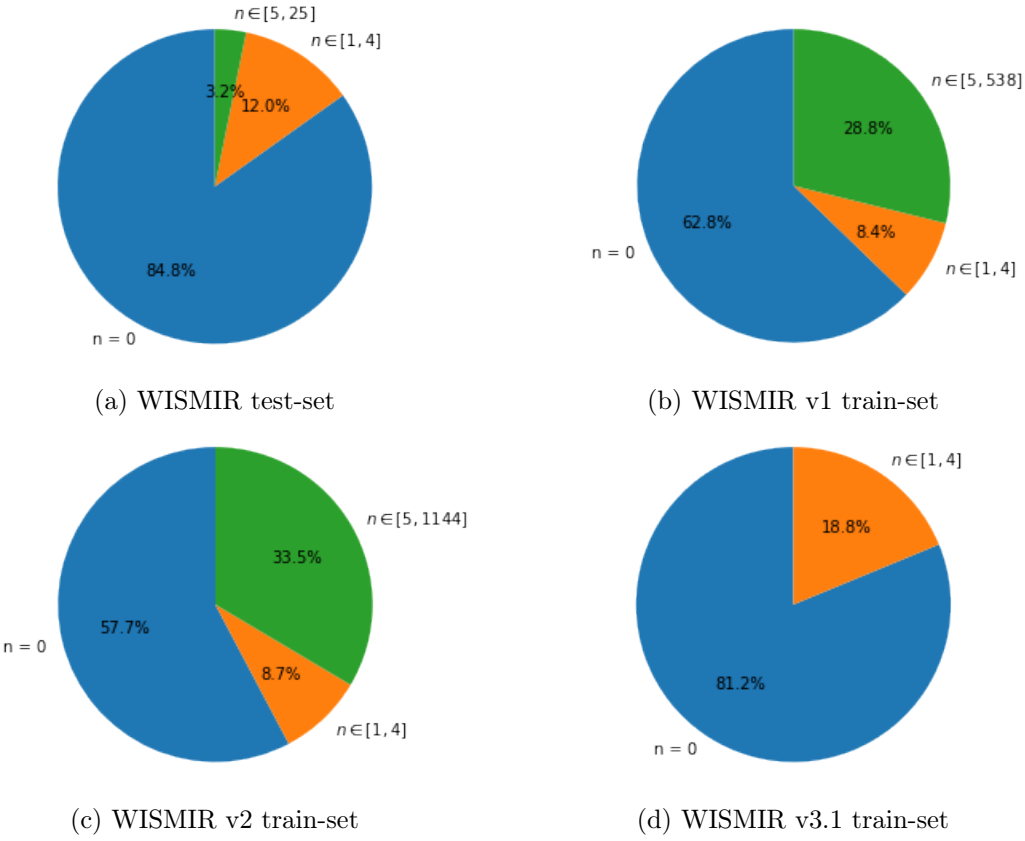
During an error analysis experiment to find systematic errors (see Section 6.3) in TERAN models trained and evaluated on the WISMIR v1 (W1) and WISMIR v2 (W2), it was found that a significant number of samples share the same caption for different images. While it is common in multi-modal datasets that an image has multiple different captions, in WISMIR, it is the other way round. In Flickr30k and COCO, for example, each image is described by five different captions. In W1 and W2, however, the same caption can describe multiple different images. To be precise, 37.2% and 42.4% of the samples of W1 and W2 train-sets, respectively, share the same caption with at least one other sample. In the test set of WISMIR, which is equal for all versions, 15% of the captions describe at least two different images. In Figure 5.2, the composition of samples of different versions of the dataset that have the same caption as n other samples is shown. Figure 5.3 show how many different images share the same caption with how many other images in WISMIR. Because the number of different images described by the same caption varies from 1 to over 1000 in the W2 train-set (see Figure 5.3c), it could be problematic for a model's generalization ability to unseen data.

Hence, two additional versions of the dataset were created: WISMIR v3 does not contain duplicated captions at all, and WISMIR v3.1, where one caption describes a maximum of

4. https://polyglot.readthedocs.io/

five different images.

Table 5.1 lists the sizes and splits of the different dataset versions. All versions of the datasets share the same test split of WISMIR v1. A detailed statistical analysis of WISMIR and a comparison with the popular COCO and Flickr30k datasets can be found in the next Section 5.3.



(a) WISMIR test-set

(b) WISMIR v1 train-set

(c) WISMIR v2 train-set

(d) WISMIR v3.1 train-set

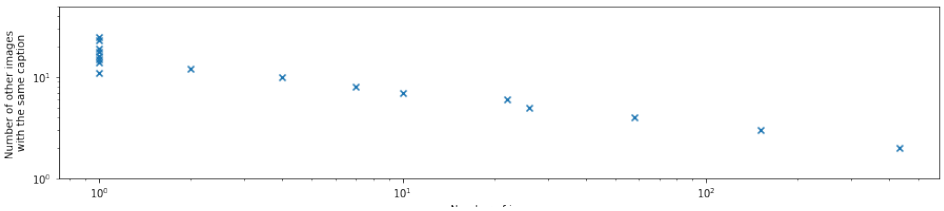
Figure 5.2: The composition of samples that share the same caption with n other samples for different versions of the WISMIR dataset.

Version	Size	Train Split	Test Split
v1	187,598	178,218	9380 (5%)
v2	395,874	386,494	9380 (2.4 %)
v3	232,530	223,150	9380 (4.2 %)
v3.1	284,110	274,730	9380 (3.3 %)

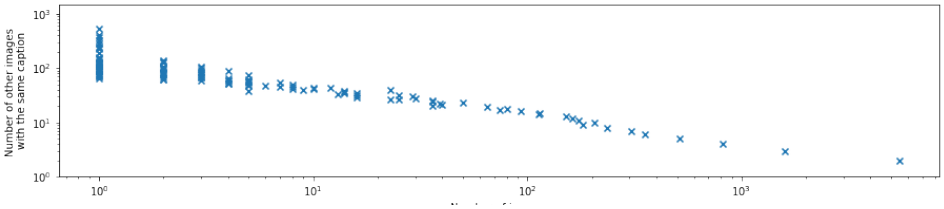
Table 5.1: Number of text-image pairs in WISMIR v1, v2 and v3 datasets and train-test splits. Note that the all the versions share the exact same test set.

5.3 Data Analysis and Comparison

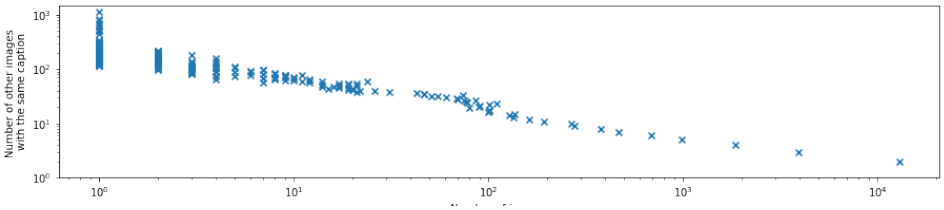
In order to verify the disparity between WISMIR and the popular datasets COCO and Flickr30k, which is a prerequisite to assess how current text-image retrieval models perform on datasets containing more complex textual data, a detailed data analysis and comparison was undertaken.



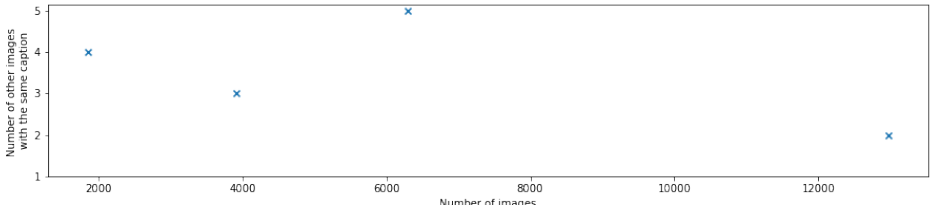
(a) WISMIR test-set



(b) WISMIR v1 train-set



(c) WISMIR v2 train-set



(d) WISMIR v3.1 train-set

Figure 5.3: Distribution of the number of images that are described by the same caption as n other images for different versions of the WISMIR dataset. Note that the axes in (a), (b), and (c) is logarithmic to highlight the long-tailed distribution characteristics. The diagrams can be read similar to horizontal bar charts.

5.3.1 Typical Samples from COCO, Flickr30k and WISMIR

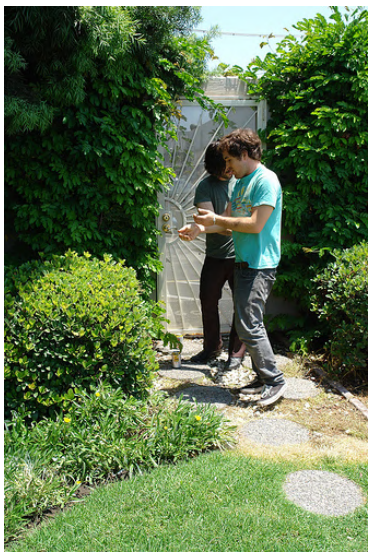
For a first impression on the differences of the three datasets, typical samples are shown in Figure 5.4, Figure 5.5, and Figure 5.4. For a first impression on the differences of the three datasets, typical samples are shown in Figure 5.4, Figure 5.5, and Figure 5.6. From the provided samples, it can be observed that the caption texts of COCO and Flickr30k are short, simple, and directly describe the prominent content of the images in one sentence. The captions of WISMIR, however, are much longer, do not follow a systematic pattern, contain multiple sentences, and do not necessarily describe the content of the image but also contain information like, e.g., the name of the photographer or the historical context. Another essential thing to notice is that the images from the three datasets do not differ from each other substantially, i.e., while the captions from Flickr30k and COCO show apparent dissimilarity to the captions from WISMIR, the images could be used interchangeably.

(a) *Two teenagers at a white sanded beach with surfboards.*

(b) *A very old and large plant vase next to others.*

(c) *Three teddy bears sit on a fake sled in fake snow.*

Figure 5.4: Three typical images with their corresponding captions from COCO dataset.



(a) *Two men in green shirts are standing in a yard .*

(b) *Three men on a large rig .*

(c) *A little girl climbing into a wooden playhouse*

Figure 5.5: Three typical images with their corresponding captions from Flickr30k dataset.

5.3.2 Statistical Analysis

First, the tool described in Section 5.1 was utilized to generate corpus statistics based on the COCO, Flickr30k, and WISMIR captions to examine patterns analytically. Note that the differences between the first and second versions of WISMIR are neglectable, which is why only WISMIR v2 and WISMIR v3 statistics are mentioned and discussed.

In the Figures 5.7, 5.8, 5.9, and 5.10, multiple box plots summarize statistics about various characteristics of the three datasets' captions. In all figures, it can be observed the resemblance of COCO and Flickr30k captions and the disparity to WISMIR captions, which verifies our first impression from the typical samples provided previously.

Number of tokens per caption One of the primary differences between captions in Flickr30k or COCO and WISMIR is the average number of tokens per caption. As shown in Figure 5.7, WISMIR captions are almost four times as long as captions from Flickr30k or COCO. This is an essential insight because having lots of tokens will most probably result in a looser coupling between the regions of an image and the words or tokens, making it more challenging for a multi-modal model to learn joint-representations or word-region-alignments.

It can be further observed from the interquartile ranges that the number of tokens

(a) View of the Kornmarkt in Trier, Rhineland-Palatinate, Germany. In the middle of the image is Stadtlesen, a mobile open air library; at left is the open air portion of the Bitburger Wirtshaus; the glass fronted building in the background at left is a bookshop.

(b) *The "Capon Lake Whipple Truss Bridge" (ən pən), formerly known as "South Branch Bridge" or "Romney Bridge", is a historic Whipple truss bridge in Capon Lake, West Virginia. It is located off Carpers Pike (West Virginia Route 259) and crosses the Cacapon River. The bridge formerly carried Capon Springs Road (West Virginia Secondary Route 16) over the Cacapon River, thus connecting the unincorporated communities of Capon Springs with Capon Lake. Photographed by Justin A. Wilcox of Washington, D.C. on October 25, 2015.*

(c) *Plaque above the entrance of the chapel of Notre-Dame de Vassivière (Puy-de-Dôme, France). The text in French is "Done the 6th day of june in the year 1555"*

Figure 5.6: Three typical images with their corresponding captions from WISMIR dataset.

per caption often varies a lot from sample to sample in WISMIR, whereas in COCO or Flickr30k, the number remains very similar for the majority of samples. This varying number of tokens per caption can also influence a model's performance since it needs to learn handling both short and extended captions.

Number of sentences per caption Figure 5.8 depicts the number of sentences contained in the datasets' captions. We can observe that WISMIR captions contain more than twice as many sentences on average than Flickr30k and COCO captions. More sentences per caption have the same effect on learning joint-representations as more tokens per caption: The model needs to learn which parts of a caption are represented in the image, and therefore need to be focused.

The box plots in Figure 5.8 for WISMIR show noticeably different numbers for maximum sentences depending on the framework used. An investigation of the samples containing these maximum number of sentences revealed that this is because some captions do not only contain English but also words or characters, e.g., from the Arabic or Cyrillic alphabets. This makes it much harder for the models employed in the frameworks and primarily trained on English data to tokenize and segment the captions. For the investigated samples, the non-English texts are translations of the English part of the captions. One explanation of why spaCy predicts a smaller number of tokens and sentences could be the size of the employed models: the spaCy models have an accumulated size of 829 MB, whereas NLTK and Polyglot models have accumulated sizes of 165 MB and 25 MB, respectively.

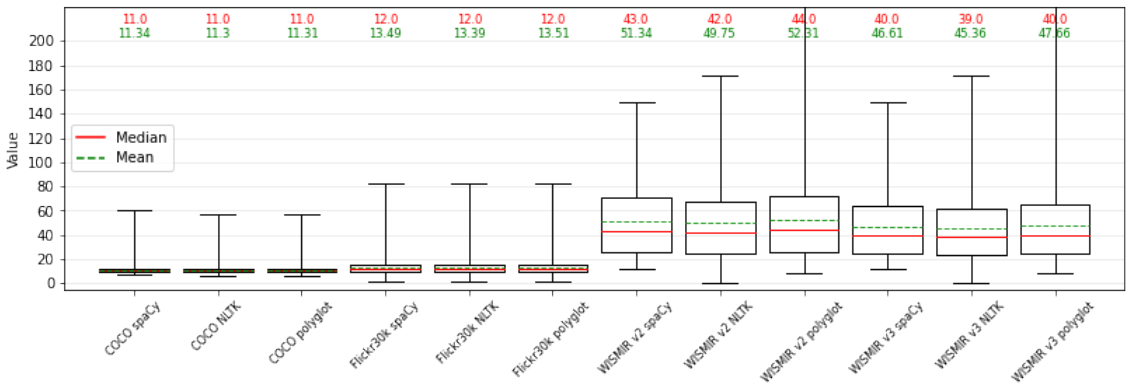


Figure 5.7: Box plot diagrams for the **number of tokens per caption in COCO**, Flickr30k, and WISMIR, generated by different tokenization models. The median and mean are depicted in red and green font, while their whiskers indicate the minima and maxima.

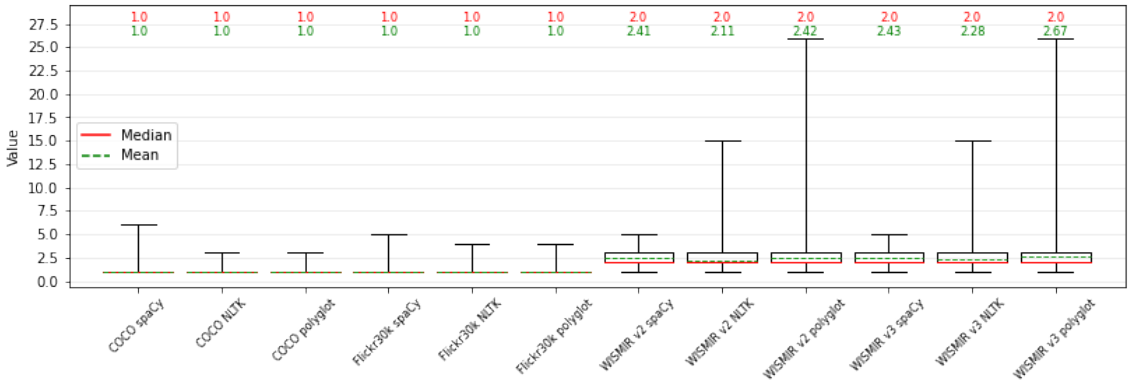


Figure 5.8: Box plot diagrams for the **number of sentences per caption** in COCO, Flickr30k, and WISMIR, generated by different tokenization models. The median and mean are depicted in red and green font, while their whiskers indicate the minima and maxima.

Further, the spaCy model, "en_core_web_lg"[5], used to generate the statistics, was trained with OntoNotes 5.0 (Weischedel et al. 2013), which contains texts in English, Mandarin Chinese, Arabic, and Chinese. Because language model sizes are generally proportional to their performance, and because the training data of the spaCy model contains non-English language, we argue that spaCy can handle non-English text better and is, therefore, more reliable than the other frameworks.

Ratio of noun tokens per caption In Figure 5.9, the ratio of tokens classified as NOUN or PROPN POS tags to all tokens of the datasets' captions is depicted.

POS tags or part-of-speech tags are labels of a word or token typically created in an automated fashion by a POS tagger model and describe the word's grammatical properties. Most European languages like English, German, or Italian and many other languages share the same or very similar set of parts of speech like nouns, verbs, adjectives, prepositions, articles, or conjunctions (Note that this is not a complete list). There exist different sets of POS tags used by POS tagger models called (POS-)tag-sets. The most popular are part of the Penn Tree Bank (Marcus et al. 1993), Universal Dependencies v2 (Nivre et al. 2020), or the Brown Corpus (Francis and Kucera 1979). The NOUN and PROPN POS tags are

5. https://spacy.io/models/en#en_core_web_md

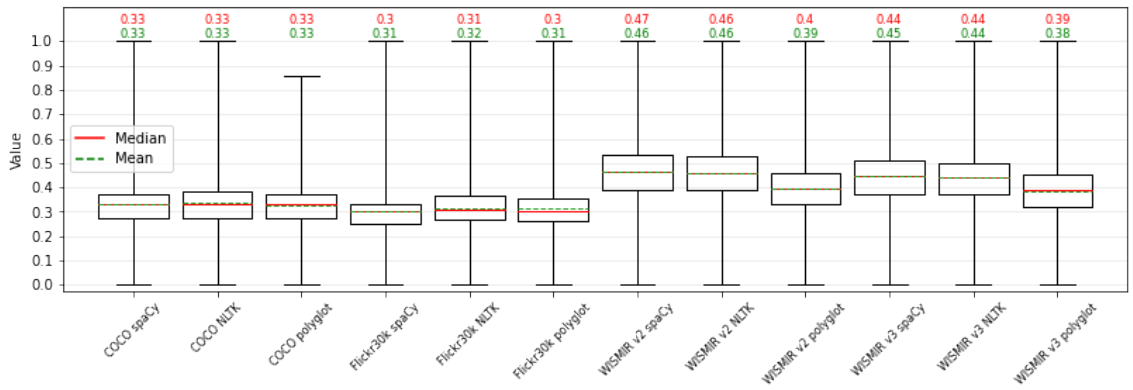


Figure 5.9: Box plot diagrams for the **ratio of tokens tagged with NOUN and PROPN POS tags and all tokens of a caption** in COCO, Flickr30k, and WISMIR, generated by different tokenization models. The median and mean are depicted in red and green font, while their whiskers indicate the minima and maxima.

reserved for nouns and proper nouns, respectively. Proper nouns are nouns that are names for specific places, objects, or persons, e.g., Hamburg, Bialetti, or Maria.

The difference between the ratio of nouns and proper nouns per caption is not as significant as the other discussed statistics, but is still 15% higher in WISMIR than in COCO or Flickr30k. This difference is worth mentioning because nouns are generally better depictable than other types of words such as prepositions, verbs, or conjunctions.

Ratio of named entities per caption The most significant difference between the captions of WISMIR and COCO or Flickr30k is the ratio of tokens part of named entities to all tokens of a caption. As it can be observed from Figure 5.10, COCO and Flickr30k captions

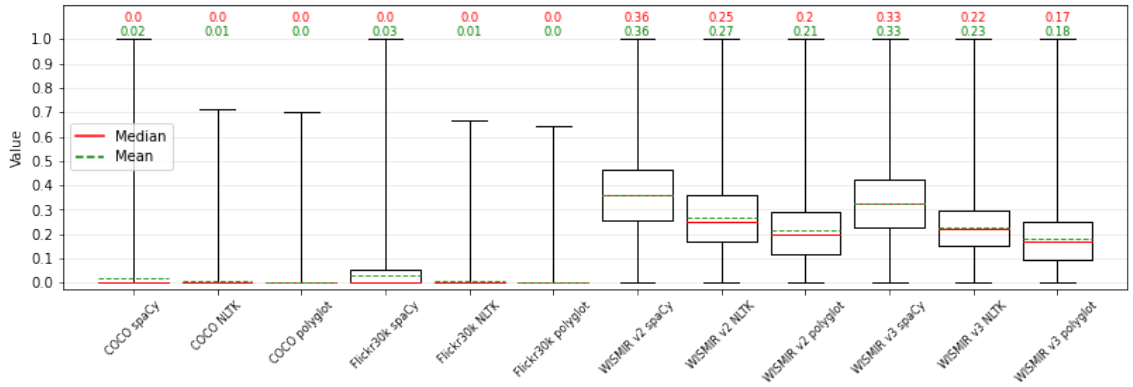


Figure 5.10: Box plot diagrams for the **ratio of tokens tagged with NOUN and PROPN POS tags and all tokens of a caption** in COCO, Flickr30k, and WISMIR, generated by different tokenization models. The median and mean are depicted in red and green font, while their whiskers indicate the minima and maxima.

have zero named entities at the median and close to zero on average. In WISMIR, however, between 18% and 36% of a caption's tokens are part of named entities.

We argue that this can greatly impact multi-modal models' performances because a model needs to recognize the entity type of the named entity to learn reliable joint representations for the respective tokens and corresponding image regions. To make this more clear, think of images of celebrities and captions containing the celebrities' names.

Although the token embeddings correlated with the celebrities' names might contain the information that it is a person's name, the multi-modal model would still need to learn to extract this information – if it is even contained – to align the textual with the corresponding visual embeddings. If the information about the entity type, in our example, a person's name, is not contained in the token embeddings, the multi-modal model additionally needs to learn it.

More information and discussion on this can be found in Section 10.6.1.

5.3.3 Readability Tests

To further underline the differences between COCO, Flickr30k, and WISMIR and to show the suitability of WISMIR in our language learner scenario, Flesch-Kincaid (Farr et al. 1951) (FK) and Dale-Chall (Chall and Dale 1995) (DC) readability scores were computed. The results shown in Figure 5.11 are calculated from random samples of the

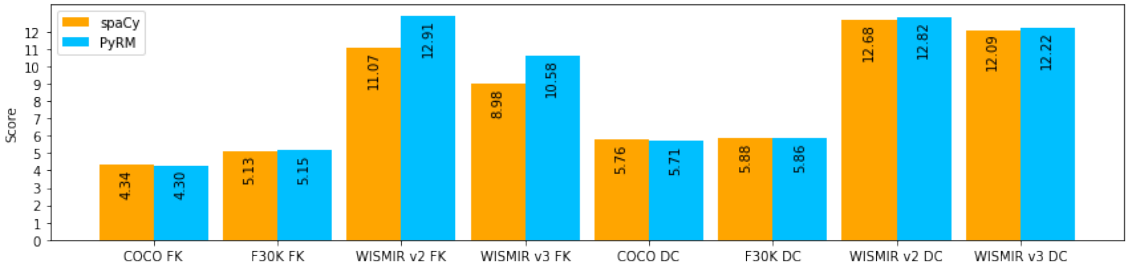


Figure 5.11: Comparison of Flesch-Kincaid (FK) and Dale-Chall (DC) readability scores of randomly sampled subsets of COCO, Flickr30k, and WISMIR captions containing $10^6 \pm 0.1\%$ characters computed by two different frameworks (spaCy and py-readability-metrics).

datasets' captions, with each sample containing $10^6 \pm 1000$ characters. Because these readability scores depend on the number of sentences, words, and syllables in the text, counted by imperfect models, two different implementations, namely spaCy-readability[6] and py-readability-metrics[7], were used to obtain more reliable results. Whereas the reported Flesch-Kincaid scores correspond already roughly to a US grade level, the Dale-Chall scores need to be interpreted according to Table 5.2.

Score	Notes
4.9 or lower	easily understood by an average 4th-grade student or lower
5.0–5.9	easily understood by an average 5th or 6th-grade student
6.0–6.9	easily understood by an average 7th or 8th-grade student
7.0–7.9	easily understood by an average 9th or 10th-grade student
8.0–8.9	easily understood by an average 11th or 12th-grade student
9.0–9.9	easily understood by an average 13th to 15th-grade (college) student

Table 5.2: Mapping of raw Dale-Chall readability scores to US grade levels. Source: https://en.wikipedia.org/wiki/Dale\T1\textendashChall_readability_formula

In Figure 5.11, it can be observed that the captions of COCO and Flickr30k should be easily understood by an average 4th to 6th-grade US student according to the Flesch-Kincaid readability test. According to the Dale-Chall scores, COCO and Flickr30k captions can be easily understood by a 5th or 6th-grade student. In contrast, WISMIR captions are understandable by college students or higher, according to Flesch-Kincaid and Dale-Chall scores.

6. https://github.com/mholtzscher/spacy_readability
7. https://github.com/cdimascio/py-readability-metrics

5.4 Challenges and Limitations

Unfortunately, most of the discoveries from this section were made towards the end of the thesis, when most of the experiments and studies already took place. Hence, not all of them were addressed in this work. Section 10.6.1 describes how they can be approached in future work.

5.4.1 Named Entities

The issue with named entities in WISMIR, as already alluded to in Section 5.3, is their high frequency in the dataset's captions. Aligning named entities in image regions is challenging because a model first needs to learn the type of the named entity to find regions in the images that represents it.

5.4.2 Loose Coupling Between Modalities

From the interquartile ranges in Figure 5.7, it can be observed that in WISMIR, 50% of the captions contain between approx. 20 and over 70 tokens depending on the tokenization model. Especially in TERAN, where a fixed number of 36 region features per image is used, this results in a loose coupling of the two modalities. Even in models like UNITER or VL-BERT, where between 10 up to 100 region features are used depending on the confidence thresholds, the extended captions in WISMIR will result in an unbalanced number of textual tokens versus image regions.

5.4.3 Language of the Captions

From visual inspections of the data, another issue with the captions of WISMIR was discovered. Some captions do not solely contain English text but also text from other languages, and even more critical, text written with characters from non-Latin alphabets like Arabic, Cyrillic, Mandarine, or other Asian alphabets. Transformer-based visio-linguistic models thematized in this thesis typically utilize pre-trained and frozen BERT-based models to tokenize the textual input forwarded through their encoder stacks. Although pre-trained multi-lingual and non-Latin BERT tokenizers models exist[8], popular models are pre-trained and evaluated with English tokenization models. Even though BERT tokenizers work on a sub-word level, non-English words and especially words of non-Latin characters will result in OOV (out-of-vocabulary) tokens, for which it is not possible to find consistent word-region-alignments. This, in turn, has a negative effect on every downstream task of visio-linguistic models, including text-image retrieval.

5.4.4 Unbalanced Data

Another issue with WISMIR data discovered by random inspections of several samples is that the data is unbalanced. Unbalanced, in this case, means that the data is not thoroughly heterogeneous but contains an unconstrained number of images described by the exact same caption, a varying number of identical images described by different captions, or many similar samples created by a single user.

Number of duplicated captions Since it is computationally easy and efficient to check the equality for textual data, the actual numbers of duplicated captions for different images in WISMIR were counted and shown in Figure 5.2 and Figure 5.3.

Number of duplicated images For the visual data, i.e., the images, finding duplicates in a large pool of images is computationally expensive due to their larger size compared to textual data. Because no servers where 300K+ images would fit into main memory were available, MD5 hashes for every image were computed and duplicated hashes were counted.

8. https://huggingface.co/bert-base-multilingual-cased

The problem with this approach is that cryptographic hash functions, like MD5, follow the avalanche criterion, which states that if a single bit in the input changes, each of the output bits changes with a 50% probability. So if only a single bit in an image is changed, the resulting hash will be very different from the hash of the original image. However, the changed image will still look the same to humans and carry the same semantic information (for typically sized images). Thus, counting duplicates of MD5 hashes of the images only reveals duplicates of exactly equal images. Note that this would also be the case if the original images were checked, i.e., their bytes without hashing, for equality.

A better approach than checking for equality would be using popular image similarity metrics like SSIM (Z. Wang et al. 2004) or mean-squared-error (MSE). Interpreting these scores and finding optimal thresholds to filter out similar images, however, is not straightforward and out of the scope of this thesis.

Table 5.3 shows the number of exactly equal images with described by different captions found via MD5 hashing. As it can be observed, the number of duplicated images in the

WISMIR Set	**Number of n-times duplicated images**	
	$n = 1$	$n = 2$
test-set	2	0
v1 train-set	83	0
v2 train-set	331	1
v3 train-set	189	0
v3.1 train-set	266	1

Table 5.3: Number of duplicated images in different versions of the WISMIR dataset. Note that for $n = 1$ and $n = 2$, the same image exists two or three times, respectively.

WISMIR datasets is rather low compared to the number of all samples (see Table 5.1).

Many similar samples Visual data inspection of multiple random samples also revealed that many images and their labels had a very similar structure. The caption of the referred samples consists of elements of a set of names of places or objects from the Philippines, and the images show either arbitrary sections of highways, roads, or shops beside the roads. With internet research to trace the origin of the named samples, it was found that there is one user, *Judgefloro*[9], who uploaded and captioned a staggering amount of $1,311,757$ photos showing "18 Philippine provinces including their 280+ Towns, Cities, Churches, Landmarks, Attractions, Monuments, Cultural heritage, Schools, Flora and Fauna, inter alia" to Wikimedia. In Figure 5.12, a random selection of the text-image pairs created by *Judgefloro* and included in the WISMIR v2 train-set is shown.

By computing the token type frequencies of the captions of WISMIR, terms like "Barangay", "Bulacan", or "Tarlac" were discovered in the first 30 places of the word-frequency lists. Usually, these places in word-frequency lists are occupied by stopwords and not by rare words mentioned above. Therefore, captions of different WISMIR versions containing at least one of the following terms were counted, of which all are frequent in the captions of the user *Judgefloro* but very rare or even non-existing terms English: "Barangay", "Tarlac", "Pampanga", "Bulacan", "Nueva Ecija", "Poblacion", "Pangasinan". As it can be observed in Table 5.4, the relative number of captions containing one of the terms mentioned above is similar in the WISMIR test-set, v1 train-set, and v2 train-set. As alluded to in previous sections and figures, those WISMIR versions contain a high number of duplicated captions. In WISMIR versions, where samples with duplicated captions were removed or limited, the number of captions containing one of *Judgefloro* frequent terms drops significantly. From this, it can be concluded that most of the captions containing these terms are duplicated a lot in the samples of the respective dataset versions. The last row of Table 5.4 shows

9. https://commons.wikimedia.org/wiki/User:Judgefloro

Dataset	Absolute	Relative
WISMIR test-set	1613	17.20%
WISMIR v1 train-set	31720	17.80%
WISMIR v2 train-set	68764	17.80%
WISMIR v3 train-set	2284	1.02%
WISMIR v3.1 train-set	10091	3.67%
WikiCaps	430972	11.26%

Table 5.4: Absolute and relative numbers of caption of different Wikipedia-based text-image datasets containing the following terms of which all are frequently used in captions of the Wikimedia user *Judgefloro* but very rare or even non-existing terms English: “Barangay”, “Tarlac”, “Bulacan”, “Nueva Ecija”, “Poblacion”, “Pangasinan”.

that also 10% of the captions in the unfiltered WikiCaps dataset contain these rare English terms. While some of the terms are most likely also taken from captions of other Wikimedia users, the high number of text-image pairs uploaded by *Judgefloro* gives strong confidence that he has created the majority of these samples.

The issue that results from having large fractions of similar samples in the training and test datasets is that models will not be able to generalize well to different data.

5.4.5 Incomplete Sentences

From visual inspections of random WISMIR samples, it was found that some of the captions are not proper English sentences but consist of listings of keywords that describe the image and its metadata or incomplete sentences. In the language learner scenario (see Section 1.1), where the aim is to support a user’s reading comprehension of natural sentences by visual cues obtained by an image retrieval model, training the model on data containing too many incomplete sentences could be problematic.

5.5 Summary

In our language learner scenario, the aim is to support human reading by providing visual cues for arbitrary texts, which is why we claim that models trained on the datasets of the first group will not perform well. Hence, a new dataset (WISMIR) for text-image retrieval tasks based on Wikipedia data to train text-image retrieval models was collected. WISMIR is a subset of WikiCaps, a dataset from the second group defined above. Different versions of WISMIR were collected with an ETL pipeline tool developed for this purpose. Further, an in-depth data analysis was conducted to compare WISMIR to popular image-retrieval datasets from the first group. Finally, the challenges and limitations of WISMIR were reported.

In the following chapter, several experiments of models trained and evaluated on the WISMIR dataset are reported.

(a) *Alat San Jose del Monte City, Bulacan Bridge in Barangay Tungkong Mangga, City of San Jose del Monte (Colinas Verdes Residential Estates and Country Club, of Sta. Lucia Realty & Development, Inc., Araneta Properties, Inc. and OPMC, North East 17 beside SM City San Jose Del Monte, connected by the Alat San Jose del Monte City, Bulacan Bridge K00 28+ 892 Load Limit 15 Tons load limit and River to Creek, connecting the Barangay Tungkong Mangga and San Jose del Monte City, Bulacan Welcome Road Signs in Barangays of Caloocan Barangay 185 16, Tala (Malaria), Caloocan City, along Quirino Highway formerly called the Manila-del Monte Garay Road or Ipo Road).*

(b) *Barangay Balingcanaway, Tarlac City along La Paz-Tarlac Road (Santa Rosa Junction-Tarlac Road from La Paz, Tarlac to Tarlac City where the new Tarlac-Pangasinan-La Union Expressway (TPLEx) NLEx Extension Phase 2 North Luzon West Expressway (NLWE) Phase 2 R-8 Extension 2).*

(c) *Timog and Panay Avenues, Barangay South Triangle, Quezon City, District 4 (Barangays of Quezon City South Triangle beside Paligsahan or Roces District 1, Laging Handa District 4 or Boy Scout Area, in Mother Ignacia Avenue, Sgt. Esguerra Avenue, by the Pedestrian footbridge (Circle, West Avenue corner Quezon Avenue, Quezon City) Ninoy Aquino monument in Quezon City; Timog Avenue, Timog Avenue, and Panay Avenue, Torre Venezia Hotel-Suites, Saint Paul the Apostle Parish Church, Blessed on January 26,1991, Camelot Hotel, Quezon City).*

(d) *Barangay Landing, Limay, Bataan, Bataan Province beside Barangays Poblacion, Wawa, Townsite, Duale, and Kitang-I, Barangays Bo. Luz & Kitang-II, Limay, Bataan, Bataan Province (along Limay, Bataan National Road, interconnecting with the Orion-Pilar-Bagac-Morong Bataan National Road gateway to the Bataan Provincial Expressway (Mariveles-Limay, Bataan section) beside the Bataan Provincial Expressway (Orion-Pilar, Bataan section) of the Bataan Provincial Expressway also known as the Roman Expressway or the Roman Superhighway, interconnecting with and into Olongapo-Gapan Road).*

Figure 5.12: Randomly chosen samples of WISMIR v2 train-set created by the Wikimedia user *Judgefloro*.

6 Experiments

In this chapter, experiments conducted within the scope of this thesis considering TERAN models are reported and discussed.

6.1 TERAN Training

This section covers details about the training processes of TERAN models trained on different versions of the WISMIR dataset. The following sections are structured similarly and report graphs of the training loss, Recall@K scores on the WISMIR test set, and a discussion of the training process results. The models' final evaluation scores on several datasets are compared and discussed in Section 6.2.

Each version of the WISMIR training sets has a different number of text-image pairs (see Table 5.1) and a varying number of duplicated captions (see Figure 5.2 and Figure 5.3). The test set of WISMIR is the same for every version and contains 9830 text-image pairs. More details on the WISMIR dataset are described in Chapter 5 and are not repeated here.

Following previous work (Messina et al. 2021; Nicola et al. 2020), visual features of images in the datasets were extracted using a Faster R-CNN with ResNet-101 (Ren et al. 2016; Anderson et al. 2018; Z. Yu et al. 2020) model with the number of extracted regions and features per image fixed to 36. The textual features are the token embeddings from the same BERT tokenizer model (Devlin et al. 2019; Wolf et al. 2020) used by the TERAN authors.

Again, following the TERAN authors, only the batch-wise loss got computed every iteration of the training loop. To check if the models were overfitting, they get evaluated on the test set via Recall@K metrics. Since the complexity of the evaluation is $O(N^2)$ (with $N = 9380$ for the WISMIR test set), this was only done after every epoch. Likewise, after every epoch, the models got persisted on disk, and the model with the best evaluation score so far was saved explicitly. This ensures that the best model is saved and serves as a replacement strategy for the missing early-stopping.

If not stated explicitly otherwise, the same set of hyperparameters was used in all the following training processes and is the same as the TERAN authors introduced:

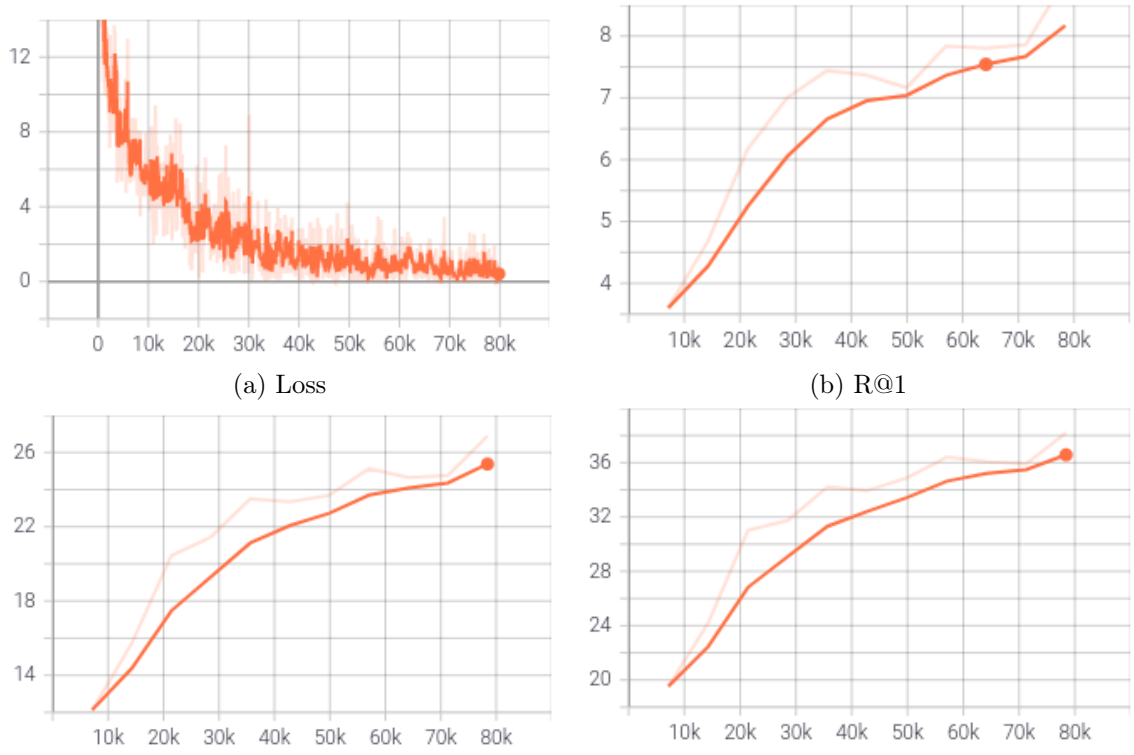
- The learning rate was set to 10^{-5} with a learning rate scheduler, which decreases the learning rate to 10^{-6} after 20 epochs.
- The margin m used for the hinge-based triplet-loss function (see Section 3.3.2) was set to 0.2.
- The batch size is set to 40 by the TERAN authors initially. However, since the available GPU memory is too small, it was reduced to 20.
- The maximum number of epochs was set to 30

Note that having the same batch size across the different training processes makes a comparison of the models' performance or loss curves on a step level possible. When talking about the training of (deep) neural networks, one step refers to one gradient update where batch-size-many samples are involved. An epoch refers to one complete iteration of the training dataset, with a varying number of steps depending on the batch size and the number of samples in the training set.

All models were trained on either a GeForce GTX 1080 Ti with 11GB memory or a GeForce GTX 2080 Ti with 12GB memory.

6.1.1 WISMIR v1

In this first training of a TERAN model on WISMIR data, the maximum number of epochs was limited to 10 since it was thought of as a first test. As shown in Figure 6.1, the model did not overfit at all and would most probably have reached better evaluation scores in successive epochs. This training experiment concluded that the WISMIR data is generally learnable by TERAN and that further training processes with more epochs and data are expected to reach better evaluation scores.



(a) Loss (b) R@1

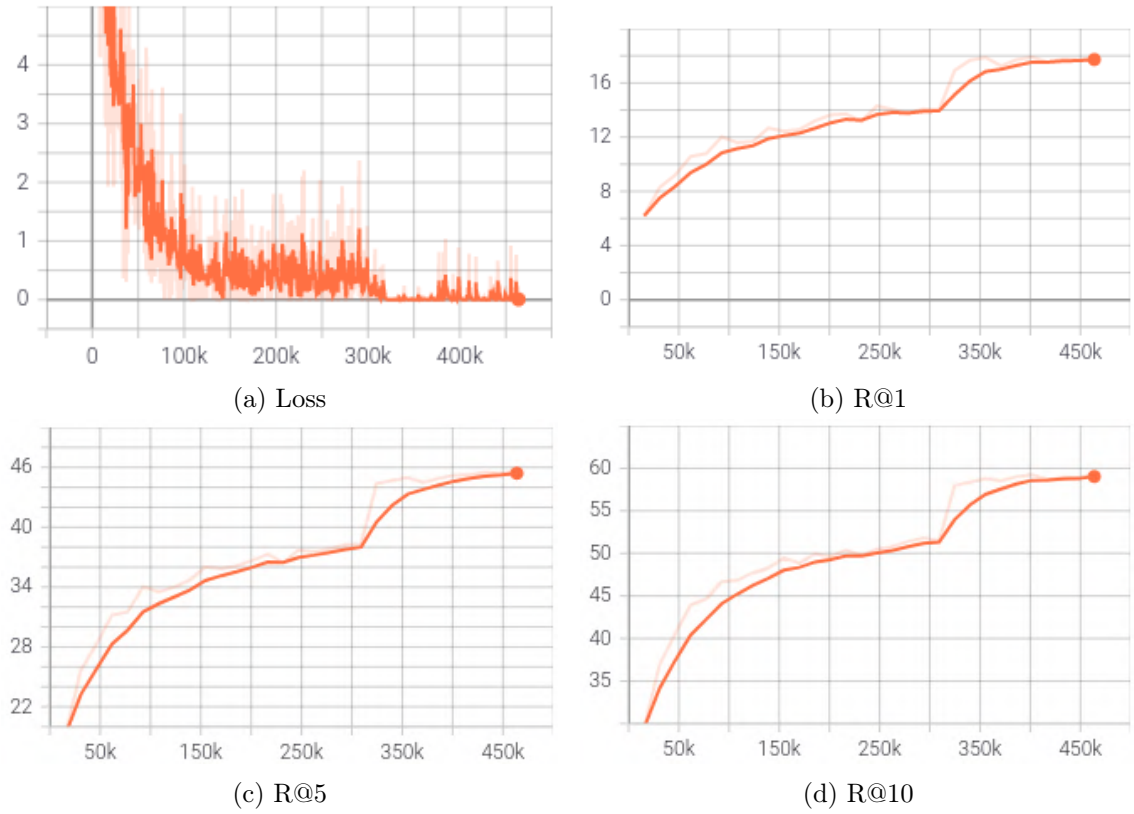
(c) R@5 (d) R@10

Figure 6.1: The loss, R@1, R@5, and R@10 curves generated during the training of TERAN on the WISMIR v1 dataset. The graphs show the number of steps on the x-axis and the corresponding value of the loss or the evaluation metrics on the y-axis.

6.1.2 WISMIR v2

The training process on the largest version of WISMIR took 4 days and about 12 hours until the maximum number of 30 epochs was reached. As observable from the loss-graph and R@k-graphs shown in Figure 6.2, the model is still not overfitted but seems to have reached its limit concerning the evaluation metrics asymptotically. Nevertheless, this also seemed to be the case around step number 300K. Shortly thereafter, however, the model's performance on the test set increased dramatically in a relatively small number of steps until the metrics flattened out again.

The finally achieved R@k evaluation scores of the trained TERAN model are relatively low compared to the achieved scores of TERAN trained and evaluated on COCO or Flickr30k (see Table6.1), and indicate that the data is challenging to learn. Further discussion on eventual reasons for the low evaluation results can be found in Section 6.2.



(a) Loss (b) R@1

(c) R@5 (d) R@10

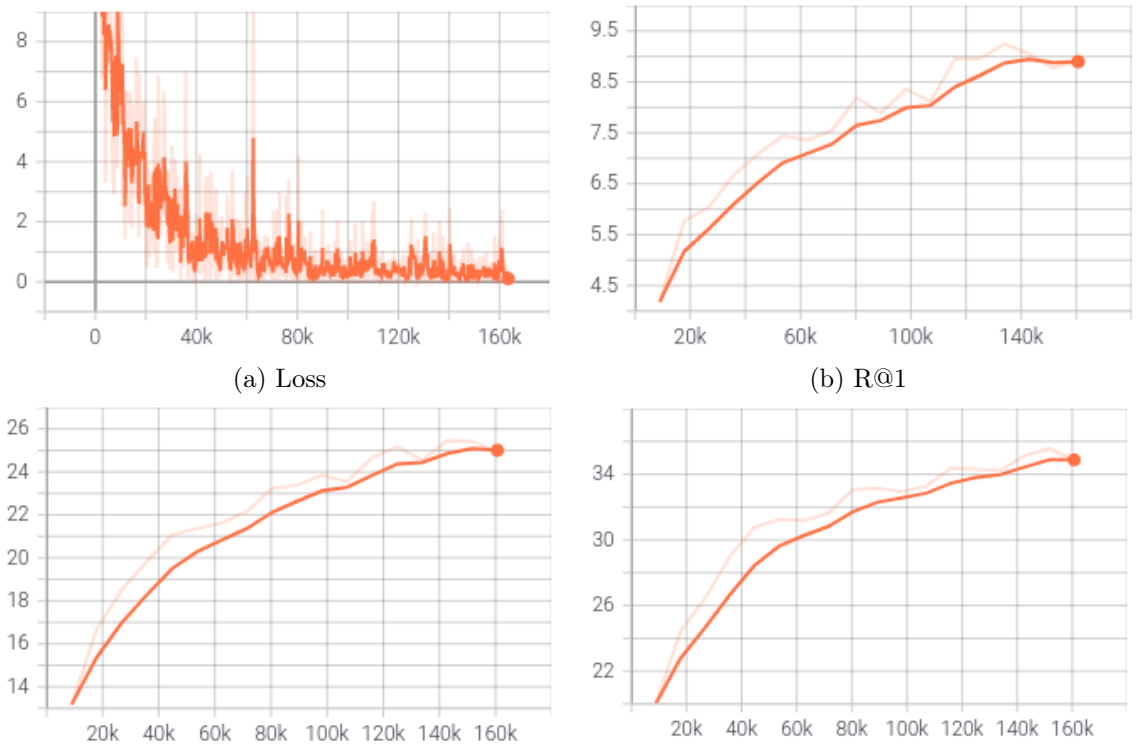
Figure 6.2: The loss, R@1, R@5, and R@10 curves generated during the training of TERAN on the WISMIR v2 dataset. The graphs show the number of steps on the x-axis and the corresponding value of the loss or the evaluation metrics on the y-axis.

6.1.3 WISMIR v3

After the large number of equal captions describing a varying number of different images (see Section 5.2.1) was discovered in the second version of WISMIR, a TERAN model was trained on WISMIR v3 to investigate the issue's influence on the model's performance. As a reminder: In WISMIR v3, all text-image pairs, where the same caption was used in another pair for a different image, were filtered out (see Section 5.2.1). In other words, in WISMIR v3, each caption is unique, whereas in WISMIR v2, a caption may be contained in up to 1000 of other text-image pairs with different images. For more information about this issue, see Section 5.2.1.

As it can be noticed by comparing the evaluation score progress of WISMIR v2 in Figure 6.2 and of WISMIR v3 shown in Figure 6.3, the filtered out samples in WISMIR v3 had a relatively large influence. After about 140K steps, the evaluation metrics of the TERAN model trained on WISMIR v3 flattened out. After about 160K steps (or 18 epochs), when the experiment was aborted, the model reached only 8.8 R@1, 25.4 R@5, and 35.6 R@10 scores. These scores were beaten by the TERAN model trained on WISMIR v2 after already after only about 60K (or 4 epochs) steps.

From the discussed results of this experiment, it can be concluded that the duplicated captions of WISMIR v2 seem to be beneficial for the model's training process. However, the size of WISMIR v3, which is only 57.74% of the second version's size, will most likely also have a non-marginal influence on the achieved performance. For more information and a comparison of the sizes of the different versions of WISMIR, see Table 5.1.



(a) Loss (b) R@1

(c) R@5 (d) R@10

Figure 6.3: The loss, R@1, R@5, and R@10 curves generated during the training of TERAN on the WISMIR v3 dataset. The graphs show the number of steps on the x-axis and the corresponding value of the loss or the evaluation metrics on the y-axis.

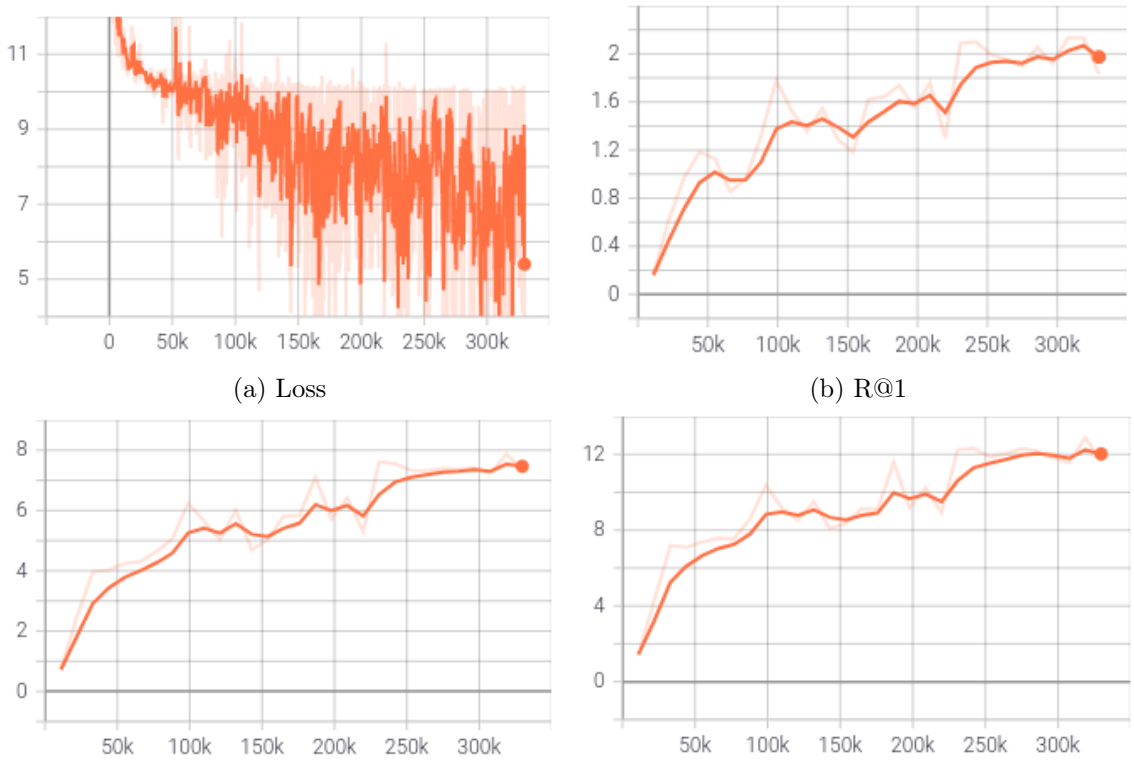
6.1.4 WISMIR v3.1

In this experiment, a TERAN model was trained on the WISMIR v3.1 dataset, a subset of WISMIR v2 similarly filtered to WISMIR v3. Instead of filtering every duplicated caption like in WISMIR v3, captions are kept so that a maximum of five duplicates remained in WISMIR v3.1 (see Section 5.2.1). In other words, in WISMIR v3.1, a caption may describe a maximum of five different images. Like the third version of the dataset, this version was created to investigate how duplicated captions affect the model's performance.

The experiment results were unexpected and chaotic, as can be seen especially in the loss curve shown in Figure 6.4. While the loss is slightly decreasing, it oscillates with a large amplitude, which increases with the number of steps. Usually, an oscillating loss curve indicates a too high learning rate, which is why the experiment was not aborted earlier since the learning rate scheduler decreases the learning rate by one order of magnitude after 20 epochs or in this case after about 220K steps. However, it can be observed from the loss curve shown in Figure 6.4a that this decrease did not have the expected effect, i.e., that the loss curve stabilizes.

Further, the graphs showing the R@K progress during the model's training process indicate that this version is especially tough to learn for TERAN. Even the final R@10 score did not surpass 12.9, whereas, in the other training experiments, the models achieved a minimum R@10 of 35.6.

From the significant differences in the resulting graphs of this training process compared to the graphs generated during the training with WISMIR v3 shown in Figure 6.3, I claim that it is very likely, that the different number of duplicated captions are not the sole cause of the observed phenomena. To examine why and what of this WISMIR version exactly caused the oscillating loss and the meager evaluation performance on the test set, successive experiments are necessary but out of the scope of this thesis.



(a) Loss (b) R@1

(c) R@5 (d) R@10

Figure 6.4: The loss, R@1, R@5, and R@10 curves generated during the training of TERAN on the WISMIR v3.1 dataset. The graphs show the number of steps on the x-axis and the corresponding value of the loss or the evaluation metrics on the y-axis.

6.2 Model Evaluations

In this experiment, the TERAN models trained on the different versions of WISMIR, COCO, or Flickr30k are evaluated on the test splits of the named datasets. Further, the performance of TERAN is compared to the performance of the UNITER base model (see Section 3.4), which was not The experiment aimed to compare the performance of the models considering the training data and investigate if the models can generalize from their training data to different test datasets. How the TERAN models were trained on WISMIR is described in Section 6.1, and the COCO and Flickr30k trained models are provided by the TERAN authors.

The evaluation results measured with the Recall@K metric are reported in Table 6.1. In the following sections, these results are discussed and interpreted.

6.2.1 Result Discussions

The nomenclature introduced with Table 6.2 is used in the following discussions to refer to the TERAN models trained on the different datasets to decrease the wordiness.

Evaluation Of TWX On WISMIR

From the upper part of Table 6.1 showing the evaluation results of the models on the WISMIR test set, it can be observed that TW2 performed best by a large margin. This is probably due to the large size of this version and that the training process was not aborted, although the evaluation metrics seemed to flatten out as discussed in the previous section and shown in Figure 6.2. However, by looking at the loss and performance curves of TW1,

Training Set	Test Set	R@1	R@5	R@10
WISMIR v1	WISMIR	8.9	26.9	38.2
WISMIR v2	**WISMIR**	**17.8**	**45.7**	**59.4**
WISMIR v3	WISMIR	8.8	25.4	35.6
WISMIR v3.1	WISMIR	2.1	7.9	12.9
Flickr30k	WISMIR	1.1	3.7	5.6
COCO	WISMIR	0.9	2.7	4.4
WISMIR v1	COCO 5k	2.0	6.9	11.5
WISMIR v2	COCO 5k	3.1	10.9	17.6
WISMIR v3	COCO 5k	3.2	11.3	18.0
WISMIR v3.1	COCO 5k	1.3	5.7	10.0
Flickr30k	COCO 5k	19.9	41.9	53.4
COCO	**COCO 5k**	**42.6**	**72.5**	**82.9**
WISMIR v1	Flickr30k 1k	4.6	14.5	22.6
WISMIR v2	Flickr30k 1k	8.1	22.9	33.1
WISMIR v3	Flickr30k 1k	8.8	25.8	36.1
WISMIR v3.1	Flickr30k 1k	3.7	12.9	20.4
Flickr30k	**Flickr30k 1k**	**59.4**	**84.8**	**90.5**
COCO	Flickr30k 1k	50.3	76.9	84.6

Table 6.1: Text-image retrieval results on several datasets evaluated via Recall@K metric for multiple TERAN models trained on different datasets.

Abbreviation	Model	Training Set
TW1	TERAN	WISMIR v1
TW2	TERAN	WISMIR v2
TW3	TERAN	WISMIR v3
TW3.1	TERAN	WISMIR v3.1
TWX	TW1, TW2, TW3, TW3.1	Any version of WISMIR
TC	TERAN	COCO
TF	TERAN	Flickr30k

Table 6.2: Abbreviations to refer to TERAN models trained on different datasets to decrease wordiness. Note that TWX refers to all of TW1, TW2, TW3, and TW3.1 models.

TW3, and TW3.1 shown in Figures 6.1, 6.3, and 6.4, respectively, it can be concluded that they most likely would never have exceeded the performance of TW2.

Although TW2's performance on WISMIR is best by a large margin, it is poor compared to what TC and TF achieved on COCO and Flickr30k test sets, respectively. From this, it can be concluded that it is hard for TERAN models to learn tight word-region-alignments and, consequently, also global text-image similarities from WISMIR data. To ensure that this comparably poor performance of TW2 on WISMIR does not originate from an eventual imbalance between the training set and test set, differences in the principal characteristics of the splits are investigated. As shown in Table 6.3, these differences between the WISMIR v2 training set and the test set are neglectable. The same is true for the differences between samples correctly ranked and incorrectly ranked by TW2 according to R@1, R@5, and R@10. Further, it was found that the model has seen that 84% of the token types, 72% of the noun token types, and 80% of the named entity types of the test set during training. From these findings, it can be concluded that the model's difficulties with WISMIR do not originate from surface forms of the dataset's captions, but from a deeper semantic or discourse level.

Sample Set	Avg. number of tokens	Ratio of noun tokens	Ratio of named entity tokens
test	51.00	0.4618	0.3542
train	51.35	0.4637	0.3598
R@1	51.05	0.4617	0.3559
not R@1	50.99	0.4618	0.3538
R@5	50.85	0.4600	0.3533
not R@5	51.13	0.4632	0.3549
R@10	50.76	0.4607	0.3536
not R@10	51.35	0.4633	0.3549

Table 6.3: A comparison of average properties per caption from different subsets of WISMIR v2 samples. In the samples column, "R@k" refers to the set of samples where the TW2 (see Table 6.2 correctly ranked the respective image in the first k positions. Samples, referred to as "not R@k", are samples, where the model did not retrieve the correct image in the first k ranks.

Additional problems could be introduced by the large number of tokens per caption on average. Most of the words in a lengthy caption are probably not grounded in an image region and can therefore be regarded as noise for word-region-alignments. When too many words are not depictable or are not grounded in the image regions, it leads to loose coupling between the caption and the image, which is not beneficial for the models' training and final performance in text-image retrieval tasks.

Further, for extended captions with many words and a limited number of 36 visual tokens per image, it could be challenging to sample good (anchor, positive, negative) triplets required by TERAN's loss function (see Section 3.3.2) and finally cause problems while training the model. Although it most probably plays a minor role, the smaller batch size used to train TW2, which is 20 compared to 40 in TC and TF training, also influences the quality of the triplets since they are sampled batch-wise.

Evaluation Of TWX On COCO And Flickr30k

By comparing the evaluation scores of TWX with the scores of TC or TF on COCO or Flickr30k (in the middle and bottom part of Table 6.1), it can be noticed that the training on WISMIR did not contribute much to the performance of TWX on COCO and Flickr30k. The same is also true the other way round, i.e., TC and TF performed very poorly on the WISMIR dataset. While TC performs better than TF on COCO and TF performs better than TC on Flickr30k, this difference is comparably small concerning the immense gaps of TWX on COCO and Flickr30k or TF and TC on WISMIR. These findings highlight the resemblance of COCO and Flickr30k and the disparity of the two datasets compared to WISMIR data.

However, it is worth pointing out that TW3 performed better on COCO and Flickr30k than TW2, although TW3 performed much worse on WISMIR than TW2. Moreover, both, TW3 and TW3.1 performed better on Flickr30k than on WISMIR. Also, the evaluation scores of TW3.1 on COCO – although they are deficient – are comparable to those of TW3.1 on WISMIR. These observations were unexpected since TW3.1 performed the worst on WISMIR by a large margin compared to TW1, TW2, and TW3. Similarly, the performance of TW3 on COCO is very similar or better compared to TW2 or TW1 on COCO, although TW3 performed worse on WISMIR. From this, it can be concluded that the removal or restriction of duplicated captions in WISMIR v3 and WISMIR v3.1 is beneficial for the performance of TWX on COCO and Flickr30k. Further, it indicates that the test set of WISMIR might be too similar to the WISMIR v2 train split and requires further investigation.

6.2.2 UNITER VS. TERAN Performance

In this experiment, the text-image retrieval performance of the UNITER$_{base}$ model, an early fusion model, is compared to the performance of different TERAN models, which are late fusion models.

Model	Test Set	R@1	R@5	R@10
TC	WISMIR	1.1	3.7	5.6
TF	WISMIR	0.9	2.7	4.4
UNITER$_{base}$	**WISMIR**	**5.31**	**13.28**	**18.75**
TC	COCO 5k	42.6	72.5	82.9
UNITER$_{base}$	**COCO 5k**	**50.33**	**78.52**	**87.16**
TF	Flickr30k 1k	59.4	84.8	90.5
UNITER$_{base}$	**Flickr30k 1k**	**72.52**	**92.36**	**96.08**

Table 6.4: Recall@K evaluation results of UNITER$_{base}$ and TERAN models (see Table 6.2) on text-image retrieval on multiple test sets.

As is can be observed from Table 6.4, which shows the evaluation results, UNITER$_{base}$ outperforms TC on COCO and TF on Flickr30k by a large margin. Nevertheless, the performance of UNITER$_{base}$ on WISMIR is deficient compared to the model's performance on COCO or Flickr30k. However, the difference between UNITER's performance and TC and TF on WISMIR is significant. As can be observed from Table 6.1, UNITER$_{base}$ even outperforms the TW3.1 on WISMIR despite having seen any WISMIR data during training.

From these results, the advantages of UNITER to TERAN models, i.e., the elaborate training process and the early-fusion architecture, can be clearly observed. The UNITER model was pre-trained on a much larger and diverse training set, which is a combination of 5.6M samples from COCO, Flickr30k, Visual Genome, SBU Captions, and Conceptual Captions (see Section 4.1). Further, the model was trained by several sophisticated self-supervised training tasks (see Section A.1) as opposed to contrastive loss training (see Section 3.3.2) of TERAN models.

The reason why TERAN models are still preferred over UNITER models despite their inferior performance is the computational efficiency of TERAN.

6.3 Word-Region-Alignment Matrix Analysis

Throughout this section, the same abbreviations of TERAN models trained on different datasets introduced in Table 6.2 are used to decrease wordiness. In this experiment, the word-region-alignment (WRA) matrices generated by different TERAN models are examined and compared to get an idea of what the models and their separate visual and textual transformer stacks learned from the respective datasets. For this, MMIRS (see Chapter 9) was utilized to retrieve images from COCO for two queries of different lengths shown in Table 6.5 and plot the corresponding WRA matrices. For a more formal definition or detailed explanation of WRA matrices see Equation 3.2 or Section 3.3.1, respectively.

From Figure 6.5, which shows the WRA matrices computed by TC, TF, and TW2 for query $Q1$ and the same image, it can be observed that the WRA matrices computed by TW2 (see Figure 6.5c) is very different from the other matrices. Remember that a cell of the WRA matrices represents the similarity between a textual token and a visual region of the corresponding query and image, respectively. So, the mth row represents the similarities between every textual token with the mth region, and the nth column represents the similarities between every region and the nth textual token. Hence, the vertical-stripe-like pattern in Figure 6.5c is unexpected because it suggests that all regions in the image are

Query ID	Number Of Tokens	Query Text
$Q1$	191	I decided I really wanted to bake a cake so I hopped into the car and drove to the grocery store. I bought flour, sugar, eggs, and some cocoa. All the other ingredients I had at home already. Once I got home I got out my big mixing bowl and measuring cups and spoons. I pre-heated the oven to 400 degrees and started pouring ingredients into the mixing bowl. After all the ingredients were mixed, I greased a cake pan and poured in the batter. I put the pan in the oven and started mixing the ingredients for the frosting in a new bowl. After baking for a while, I took the cake out and stuck in a toothpick to make sure the center was done. I let the cake cool for a bit and then put the frosting on it. After frosting went on, I cut it into pieces and put one on a plate for myself and another for my friend
$Q2$	14	My friend lives on a farm with his family and some sheep and goose
$Q3$	16	One problem with motorboats is that they're very harmful for animals living underwater

Table 6.5: Different queries used in the WRA analysis experiment described in Section 6.3.

almost identically similar to the textual tokens – at least from the first glance. Or in other words, a textual token is almost identically similar to every region in the image. From closer visual inspection, however, minor differences can be noticed. Further, it was tested numerically with particular attention to common floating-point equality issues to ensure that the cells per column are indeed not equal.

However, it was expected that the WRA matrices computed by all TERAN models would look similar to each other – as do the matrices computed by TC and TF. Note that vertical-stripe-pattern in the WRA matrix of TW2 shown in Figure 6.5c is not a coincidence, but is prominent in almost all generated WRA matrices inspected so far.

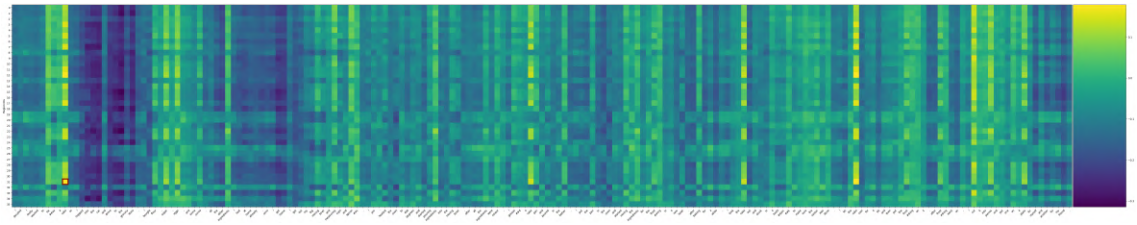
Nevertheless, it is essential to note that the rank of the image corresponding to the WRA matrices shown in Figure 6.5 only differs by three places in the ranks predicted by the different TERAN models. TC and TF ranked the image first, while TW2 ranked the image fourth. From this, it can be followed that the information contained in the WRA matrix computed by TW2 is de facto meaningful for the model, despite the only slightly different similarities between a token and the regions of the image.

In the following discussions and figure captions, this image is referred to as I.

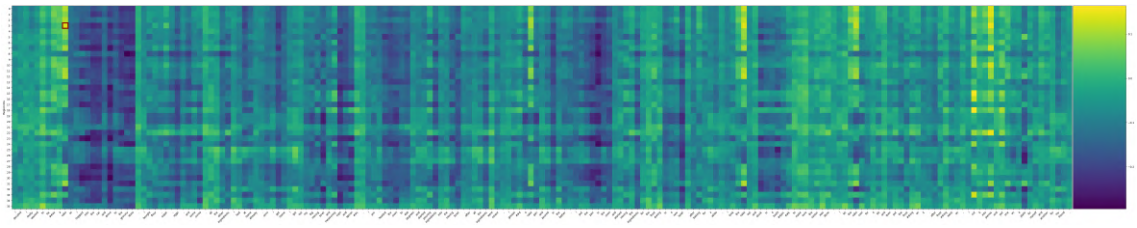
To see if the vertical-stripe pattern also reoccurs for shorter queries, $Q2$ and $Q3$ were used to compute additional WRA matrices with TW2. As shown by the WRA matrix plots in Figure 6.6, the pattern does reoccur, although it is less prominent for the first and second-ranked images of $Q2$ shown in Figure 6.6a and Figure 6.6b, respectively. From this, it can be followed that the length of the query does not influence the vertical-stripe-patterns in the WRA matrices computed by TW2 – at least at inference time.

6.3.1 Deeper Investigations

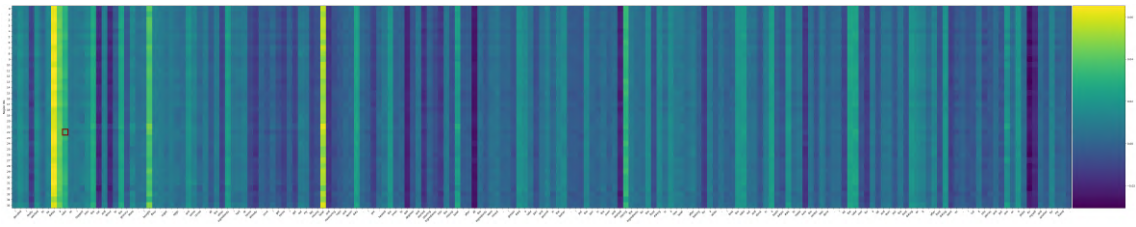
In this section, further investigation of the origins of the stripe-like pattern in the WRA matrices computed by TW2 is reported and discussed. Therefore it is essential to under-



(a) Computed by TC. Ranked 1st.



(b) Computed by TF. Ranked 1st.



(c) Computed by TW2. Ranked 4th.

Figure 6.5: Three word-region-alignment (WRA) matrices computed by different TERAN models for query $Q1$ and the retrieved image I. Note that the focus is on the general patterns and not the details of the WRA matrices.

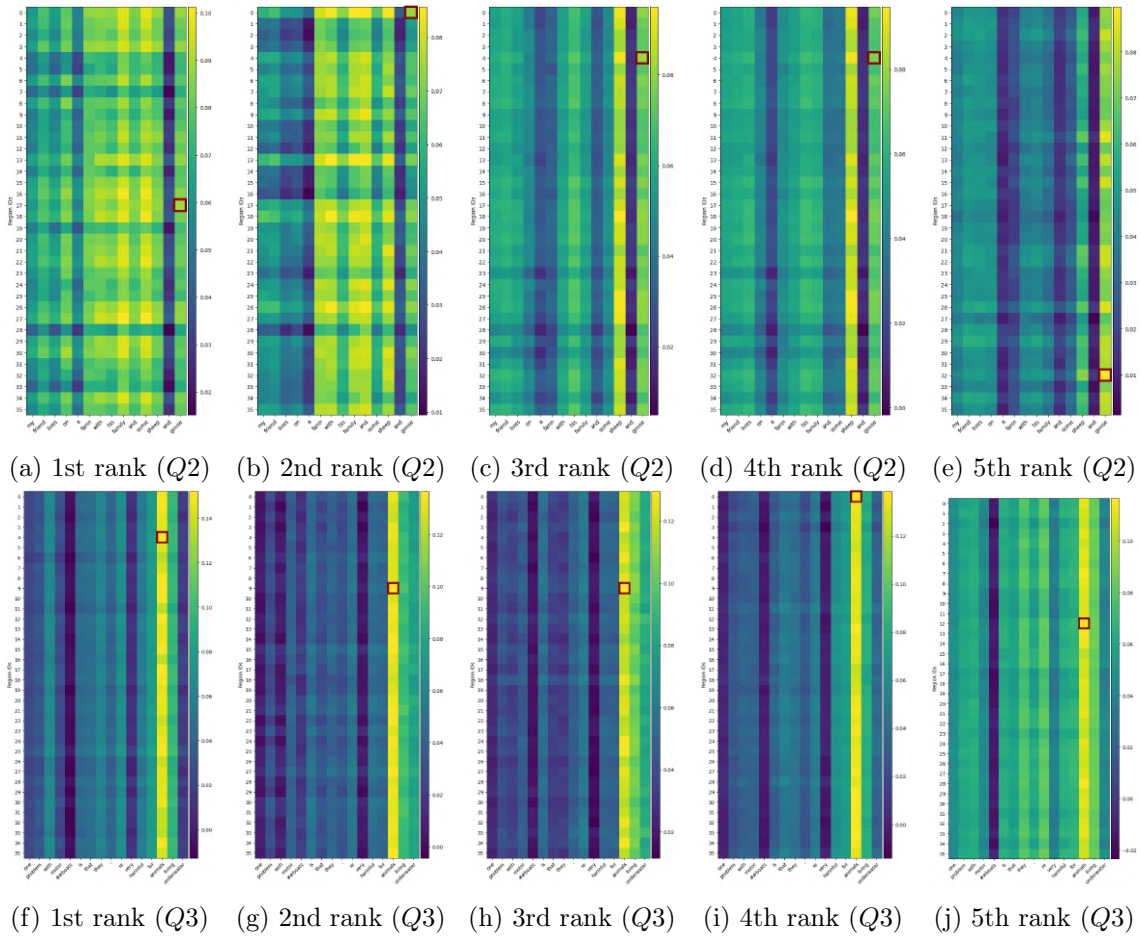
stand the architecture of TERAN and how the model computes the WRA matrices, which is described in detail in Section 3.3 and Section 3.3.1. As a reminder: The cells of the WRA matrices are the cosine-similarities of the embedding vectors of the textual features of a query and the visual features of an image computed by forwarding these vectors through the transformer stack of the respective modality.

The textual and visual features are computed from the same pretrained models for all considered TERAN models, i.e., from a pretrained BERT tokenizer model and a pertained Faster R-CNN, respectively. Since the striped pattern only appears in TW2 and not in TF or TC, it cannot originate from the feature vectors. Hence, the only possible origins are the embeddings computed from the feature vectors by either one or both of TERAN's uni-modal transformer encoder stacks.

To examine these 1024 dimensional embedding vectors, they are visualized as one-dimensional horizontal heatmaps in the following figures. Note that the single values of the vectors are not of importance here. Instead, high-level patterns emerging by vertically stacking the visualizations of the vectors are essential for the following visual inspections.

With the first visual inspection, the textual and visual token embeddings used by TW2 to compute the WRA matrix shown in Figure 6.5c and Figure 6.7a are examined.

From Figure 6.7b showing visualizations of the 191 textual token embeddings from $Q1$ computed by the textual transformer stack of TW2, a horizontally orientated striped pattern can be observed. This indicates that the values of the single dimensions of the individual query token embeddings do not vary much for most of the vectors. That is, for some embedding vectors, all the values are generally smaller than for other embedding vectors. Or, more formally put, all of the vectors have a small standard deviation across the dimensions, but some have higher, and some have lower means than others. However, from closer observation, it can be noticed that some embedding vectors do not follow this



(a) 1st rank ($Q2$) (b) 2nd rank ($Q2$) (c) 3rd rank ($Q2$) (d) 4th rank ($Q2$) (e) 5th rank ($Q2$)

(f) 1st rank ($Q3$) (g) 2nd rank ($Q3$) (h) 3rd rank ($Q3$) (i) 4th rank ($Q3$) (j) 5th rank ($Q3$)

Figure 6.6: Word-region-alignment (WRA) matrices of the top-5 ranked images computed by TW2 for query $Q2$ and $Q3$. Note that the focus is on the general patterns and not the details of the WRA matrices.

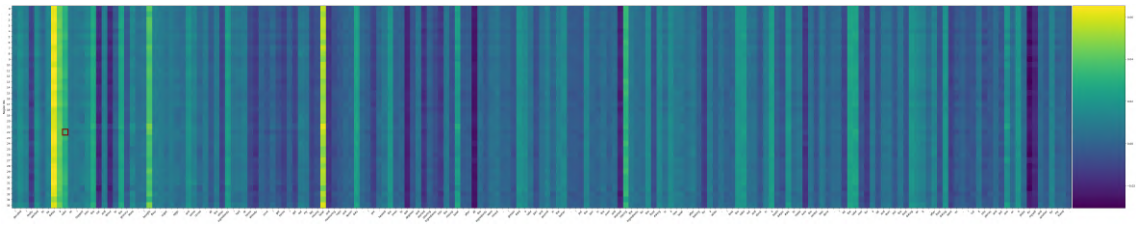
pattern, i.e., the values of the single dimensions differ much more. Another interesting structure that emerges is the prominent vertical line that visually goes through the middle of all embeddings. In this line, the values of the corresponding dimensions in the embedding vectors are either very low or very high compared to the rest of the vectors. There is a similar but not as prominent vertical line noticeable left to the other, more dominant line. While it is evident that the transformer encoded certain information in these lines and the emerging patterns generally, they can currently not be explained thoroughly and need further investigation.

By visually inspecting the 36 visual embeddings shown in Figure 6.7c and computed by the respective transformer stack of TW2, another emerging pattern is prominent. Here it looks as if all the 36 embedding vectors are identical. However, closer visual and exact numerical inspection showed that this is not the case. That is, the values of the single dimensions across the vectors differ, albeit only marginally.

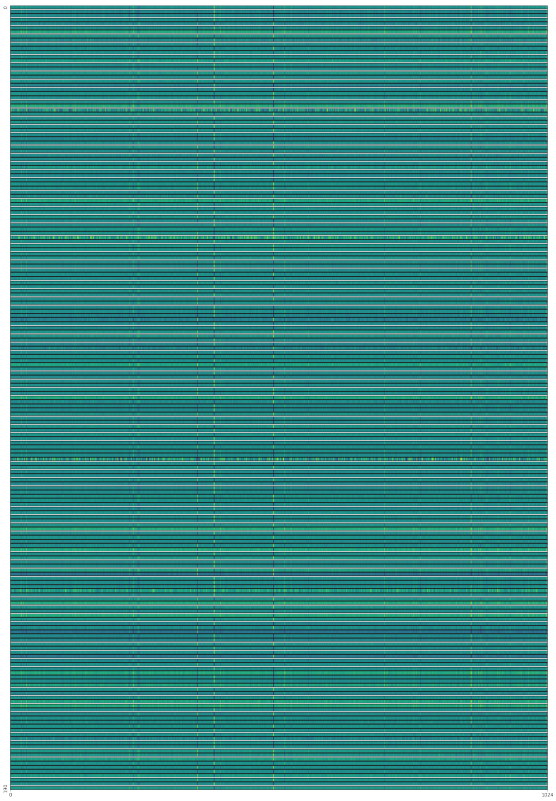
From these reported findings in the textual and visual embeddings, the prominent vertical stripe-like pattern in the resulting WRA matrix shown in Figure 6.5c and Figure 6.7a becomes explainable: Since the visual token embeddings are almost identical to each other, the similarities to the textual token embeddings are also similar, which is reflected by the vertical stripes in the WRA matrix.

To ensure that the patterns found in the textual and visual embeddings computed by TW2 are not characteristically for all TERAN models, the embeddings computed by TC and TF were also visualized in Figure 6.8 and Figure 6.9, respectively.

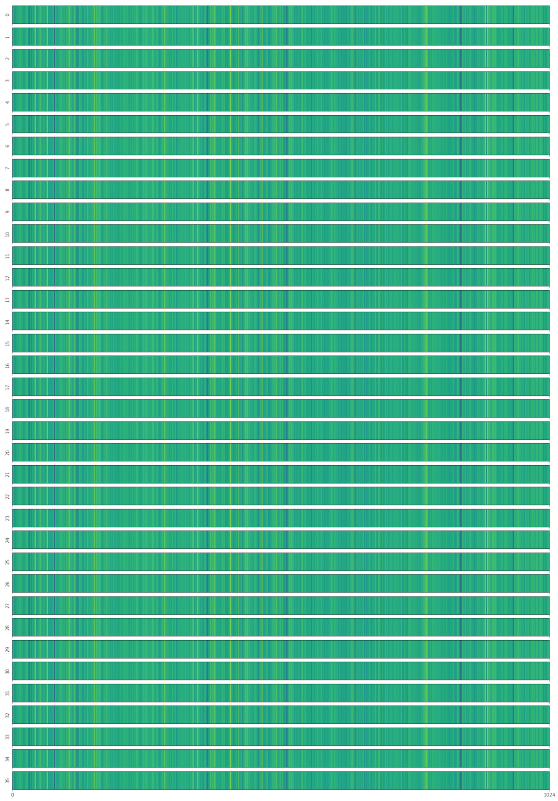
None of the previously emerged and discussed patterns show up in any of the figures – neither in the visualizations of the textual or visual embeddings nor in the resulting WRA matrices. This underlines the untypical behavior of TW2, which could be a reason for the comparably poor performance compared to TC and TF as reported in Section 6.2, Section 8.2, and Section 8.3.



(a) WRA Matrix for $Q1$ and I computed by TW2.

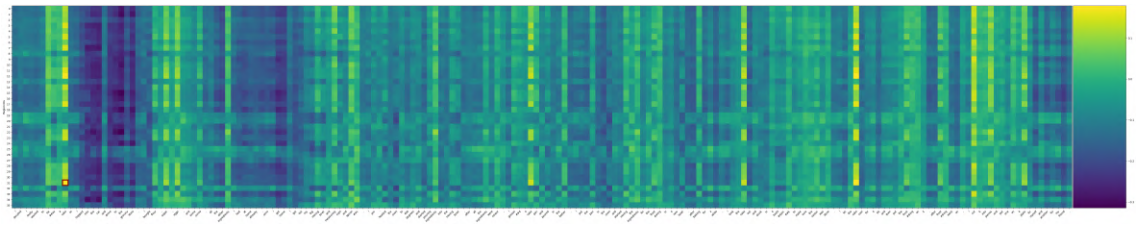


(b) Visualization of the embeddings of every textual token in $Q1$ computed by TW2.

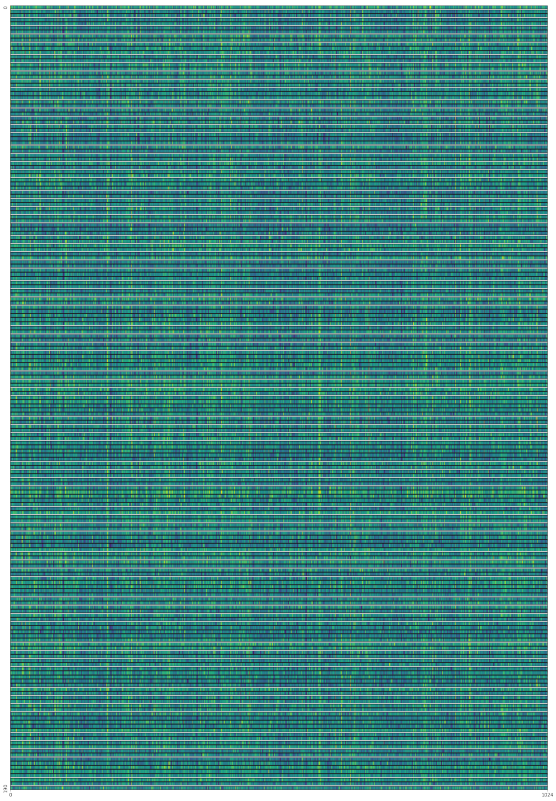


(c) Visualization of the embeddings of every visual token in I computed by TW2.

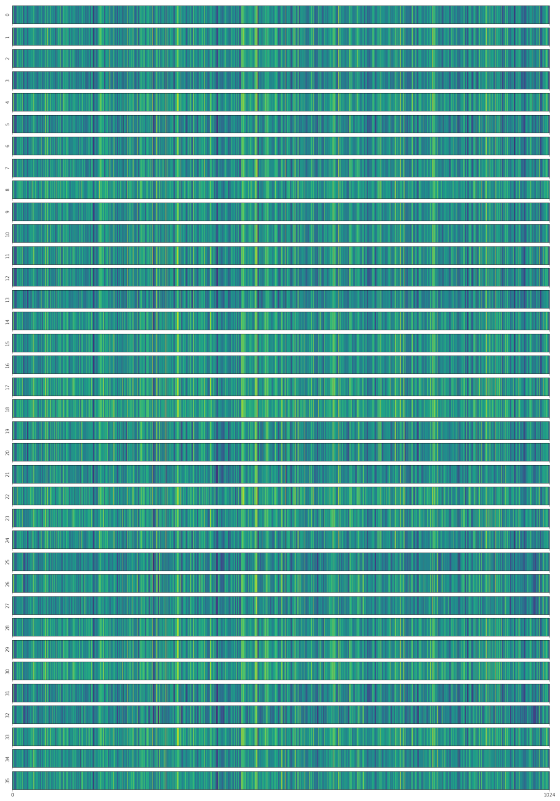
Figure 6.7: Visualizations of the query embeddings and the image embeddings used to calculate the WRA matrix of query $Q1$ and image I computed by TW2. Each row in the embedding visualizations shows visualization of a single 1024 dimensional embedding vector computed by either the textual or visual transformer stack of TW2. Note that the focus is on the high-level patterns and not the details of the plots.



(a) WRA matrix for $Q1$ and I computed by TC.

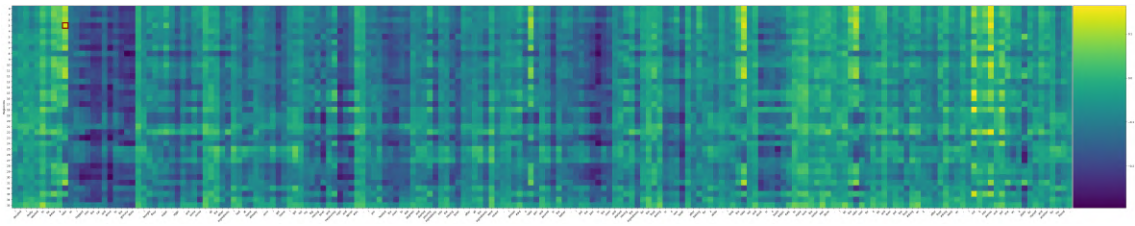


(b) Visualization of the embeddings of every textual token in $Q1$ computed by TC.

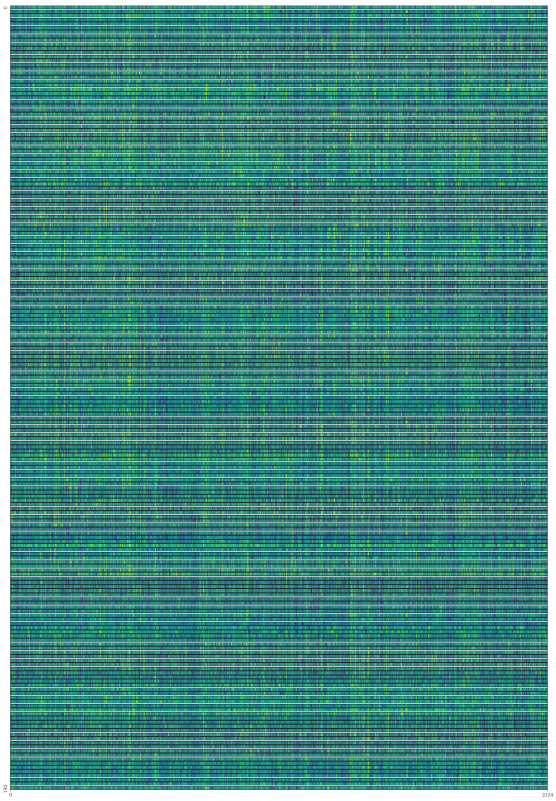


(c) Visualization of the embeddings of every visual token in I computed by TC.

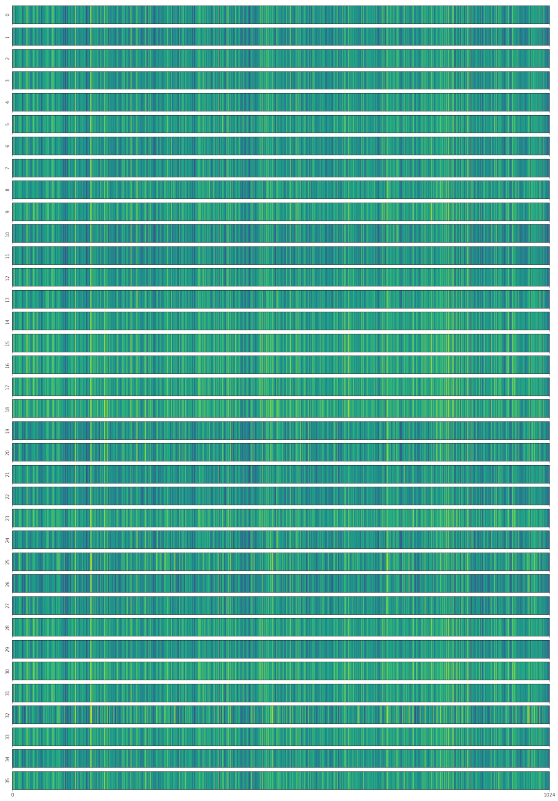
Figure 6.8: Visualizations of the query embeddings and the image embeddings used to calculate the WRA matrix of query $Q1$ and image I computed by TC. Each row in the embedding visualizations shows visualization of a single 1024 dimensional embedding vector computed by either the textual or visual transformer stack of TC. Note that the focus is on the high-level patterns and not the details of the plots.



(a) WRA Matrix for $Q1$ and I computed by TF.



(b) Visualization of the embeddings of every textual token in $Q1$ computed by TF.



(c) Visualization of the embeddings of every visual token in I computed by TF.

Figure 6.9: Visualizations of the query embeddings and the image embeddings used to calculate the WRA matrix of query $Q1$ and image I computed by TF. Each row in the embedding visualizations shows visualization of a single 1024 dimensional embedding vector computed by either the textual or visual transformer stack of TF. Note that the focus is on the high-level patterns and not the details of the plots.

7 IRST: Image Ranking Study Tool

To conduct the user studies described in Chapter 8, conveniently without the need to write new code for each (pilot) study, a multi-purpose tool was developed. As the name of the tool, **I**mage **R**anking **S**tudy **T**ool or IRST, suggests, its main objective is to conduct user studies to evaluate image rankings. A ranking is the output of a text-image retrieval model and consists of a text or query and a sorted list of k images. The first element of the list is the most similar and the last element is the least similar image to the text according to the model.

The tool currently offers three different ways, referred to as study types, to assess the quality of such rankings by human raters. Those are "image ranking", "image rating", and "Likert", and are described in the following section.

Moreover, IRST can be used as a standalone application or, in combination with Amazon's crowdsourcing platform, MTurk[1]. An introduction to MTurk is given in Section 8.1.

More information about the tool's features is provided in Section 7.2. Details about the tool design and software architecture are explained in Section A.3.

7.1 Study Types

In the following, the three study types or methods supported by IRST are introduced from a non-technical perspective.

7.1.1 Ranking Study

One of the three ways an image-retrieval result can be evaluated with IRST is through a *Ranking Study*. In this kind of study, the rater's objective is to rank her best-matching images according to a caption text from a pool of images. An example screenshot of a task from a *Ranking Study* is shown in Figure 7.1.

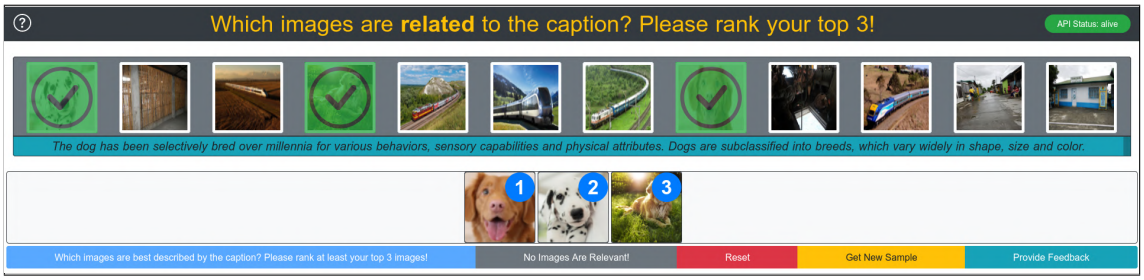


Figure 7.1: The user interface of a task from a *Ranking Study* in the IRST application. Here, a user has to rank her best matching images according to a text from the pool of provided images. This is done by dragging an image from the center into the ranking area on the bottom of the application. Note that the screenshot was cropped and slightly edited to better fit this document.

To rank the images, users have to drag and drop the images from the center into the ranking area at the bottom part of the application. An image is ranked if a green checkmark-overlay appears on the respective image in the application center. The image's rank is

1. https://www.mturk.com/

indicated by the number in the blue badge in the top-right corner of the respective image in the ranking area. A ranked image can also be reordered per drag and drop if a rater decides to change its rank. A user can click on any image to enlarge it, which opens a modal window containing a larger version with the caption beneath. If users need help or further instructions, they can click on the quotation mark in the top-left, and a detailed instructions page shows up.

This study method aims to examine how much the users and the model agree on the ranking of images.

7.1.2 Rating Study

In a Rating Study, the tasks' objective is to rate how well images are related to a caption text on a star scale. By default, the scale ranges from 0 to 5 stars with 0.5 steps. An example screenshot of a task from a Rating Study is shown in Figure 7.2.

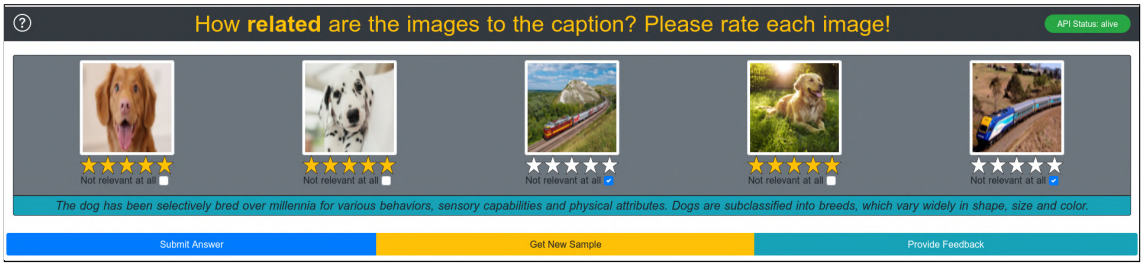


Figure 7.2: User interface of the IRST application when working on a *Rating Study*. Here a user has to rate images on a 5 star scale according to the relatedness to a caption text. Note that the screenshot was cropped and slightly edited to better fit this document.

To select a rating for an image, the users have to click on the desired number of stars beneath the respective image. If an image is entirely unrelated to the caption, the users can select the intended checkbox below the star scale.

7.1.3 Likert Study

In a *Likert Study* task, users have to answer questions concerning the image ranking on a Likert Scale. An example of a *Likert Study* task is depicted in Figure 7.3.

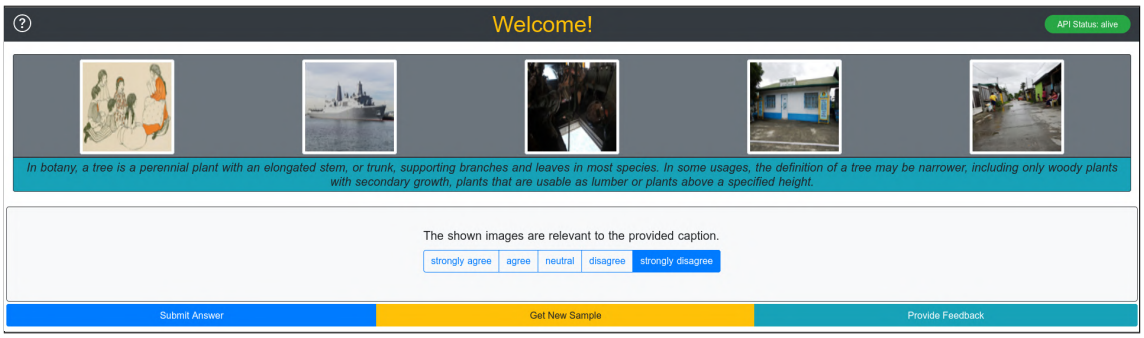


Figure 7.3: User interface of the IRST application when working on a `LikertSample`. In this task, the users have to answer the question in the bottom part of the application by selecting one of the provided answers. Note that the screenshot was cropped and slightly edited to better fit this document.

A user selects her preferred answer on the Likert Scale at the bottom of the application to answer the question. By default, the scale contains the answers "strongly agree", "agree",

"neutral", "disagree", and "strongly disagree" with respective weights 2, 1, 0, −1, and 2. A typical question could be "The shown images are relevant to the provided caption." for example.

7.2 Features

Convenient Data Input

IRST imports the image rankings that are the subject of a study from a pandas[2] DataFrame. The DataFrame requires three mandatory columns and can contain any number of additional columns (see Figure 7.4). Column "sample_id" contains the ID of the sam-

	sample_id	caption	top_k_matches	image_dataset	model
0	inscript_0	" After playing football with my friends , I ...	[128106, 154600, 566931, 348359, 260857, 38193...	coco	teran_coco
1	inscript_4	I like taking a bath instead of a shower somet...	[240111, 507274, 080200, 502422, 050521, 19579...	coco	teran_coco
2	inscript_6	Last night , I took a bath . I take baths when...	[204606, 379672, 164602, 288215, 190863, 40763...	coco	teran_coco
3	inscript_7	I went into my bathroom and started to run the...	[348359, 034760, 507274, 560644, 151567, 32276...	coco	teran_coco
4	inscript_8	Each night after we finish eating our dinner ,...	[507274, 415201, 228435, 379672, 078035, 45451...	coco	teran_coco
...	...	...	...	...	...
195	inscript_852	My husband and I love trees . In the spring we...	[147049, 449613, 239436, 200678, 302290, 17046...	coco	teran_coco
196	inscript_856	Earth day is a very important day in America	[444982, 203868, 062831, 239436, 368293, 30474...	coco	teran_coco
197	inscript_858	In the morning , I woke up to prepare for the ...	[429807, 226147, 020671, 349017, 304385, 17063...	coco	teran_coco
198	inscript_859	I travel to the local nursery to select a tree...	[357930, 217228, 444982, 203868, 167843, 46393...	coco	teran_coco
199	inscript_860	A few years ago , I did some volunteer work in...	[463932, 167843, 494555, 179997, 232160, 14704...	coco	teran_coco

Figure 7.4: Example DataFrame containing an image ranking result of a text-image retrieval model in each row. In the three mandatory columns "sample_id", "caption", and "top_k_matches", the ID of the sample from the original dataset, the query of the text-image retrieval task, and the list of the top-k images are contained, respectively.

ple in the original dataset, "caption" is the text, for which the top-k images in column "top_k_matches" are retrieved. In Figure 7.4, "image_dataset" and "model" columns are optional columns that can, in this case, provide information, from which dataset the images originate and which model produced the rankings, respectively. This can be useful, e.g., when evaluating the submitted results.

User Feedback

A user or worker can provide or is asked to provide feedback when working on a specific task. Users or workers can decide to anonymize the feedback so that only the message and no information about the user get logged and persisted.

Standalone Mode

As opposed to MTurk mode (see Section 7.2), in standalone mode, the researcher has to find the raters to conduct the study on her own. Once the raters are found, they have to register and log in at IRST to access the tasks and submit their results. See Section 7.2 for more information about the multi-user support). The tool keeps track of the tasks and submissions and ensures coordinated task distribution and results collection.

2. https://pandas.pydata.org/

MTurk Mode

Understanding MTurk mode requires knowing the basic principles of Amazon Mechanical or MTurk, explained briefly in the following. More detailed information about MTurk is described in Section 8.1.

In MTurk mode, the raters who conduct the study are workers from MTurk marketplace, a web application that runs in any standard web browser. In this marketplace, workers can browse through and work on tasks referred to as HITs (Human Intelligence Task).

In MTurk mode, IRST offers an easy-to-use function to publish user studies described in 7.1 in the MTurk Marketplace. When a worker submits her result, it gets persisted in IRST, and MTurk gets notified to register the assignment necessary for the worker to get paid. Additionally, the tool provides several other functions to manage HITs, approve assignments, and manage qualification requirements. With qualifications, requesters can control which Workers can accept and work on published HITs. There are custom and pre-defined qualifications, both of which IRST can manage. Further, the tool offers functionality to communicate with Workers who participated in previous studies. MTurk provides a sandbox version where HITs can be tested and published without payment, and a live version, to conduct the actual study. IRST supports both sandbox and live version.

Note that every administrative MTurk functionality is only available for "admin" users.

Multi-User Support

There are two types of users, namely "admin" and "basic" users. Admin users have to be defined in the configuration file and cannot register at IRST as opposed to basic users. Note that basic users can only register and login in standalone mode; in MTurk mode, this functionality is not available. In standalone mode, a user must be logged in to participate in a study and submit results. This is mandatory to compute inter-rater agreements of the results, which in turn is essential to evaluate study results.

Secure Authentication and Authorization

For authentication and authorization, IRST works with a combination of PBKDF2[3] and JSON Web Tokens (JWT)[4]. This layer of security is required to protect administrative functions against unauthorized requests, especially when the tool has to be accessible from the public internet in MTurk mode. Details about the processes of authentication and authorization are described in Section A.3.6.

SwaggerUI

Thanks to the usage of the FastAPI framework, IRST's REST API automatically follows the OpenAPI[5] specification, formerly known as Swagger. Additionally, the framework generates SwaggerUI[6], a web application to visualize and consume the API endpoints. In the current version of IRST, all administrative functions are available via SwaggerUI.

Configuration System

To keep IRST as flexible as possible, almost all of the tool's functionality is configurable in the respective section of a YAML configuration file.

The configuration system is build with OmegaConf[7], to support specification via environment variables, which is helpful for the deployment of the application.

3. https://www.ietf.org/rfc/rfc2898.txt
4. https://jwt.io/
5. https://www.openapis.org/
6. https://swagger.io/tools/swagger-ui/
7. https://omegaconf.readthedocs.io

7.3 Summary

This chapter introduced the Image Ranking Study Tool (IRST) which was developed to conduct the user studies planned in this work. With the help of IRST, user studies to evaluate the rankings of a text-image retrieval model can be conducted conveniently using three different methods without the need to write new code for each study. While the tool can be run in a standalone mode where raters have to be found by the researcher manually, it also offers out-of-the-box support for Amazon's crowdsourcing platform Mechanical Turk (MTurk).

Details about the software architecture and employed technologies are elucidated in Section A.3 In the next chapter, the user studies conducted on MTurk using IRST are reported.

8 User Studies

Two user studies were planned and conducted to answer the research questions of this thesis described in Section 1.2.

The objective of the first study is to let humans evaluate the results of a TERAN model trained on the WISMIR dataset (see Section 8.2). With the second study, we assess and compare how TERAN models trained on different datasets perform on L2 language learner data (see Section 8.3). Both studies were conducted on Amazon's crowdsourcing platform MTurk with the IRST application.

8.1 Amazon Mechanical Turk

Amazon's Mechanical Turk, or short MTurk is a digital crowdsourcing platform. On MTurk, researchers or companies can have crowdsourcing tasks like conducting studies or collecting datasets done by over 250K (Robinson et al. 2019) diverse workers from all over the world.

There are two roles on MTurk: requesters and workers. Requesters create and publish their research projects, chunked into multiple small tasks referred to as HITs (Human Intelligence Tasks) on the MTurk marketplace. Workers browse through the marketplace and accept HITs they find interesting and attractive to earn money.

Further, it is possible to constrain the HITs with qualification requirements so that only workers, which meet the requirements, can work on the HIT. This comes in especially handy when pilot studies need to be conducted to ensure only genuine workers can accept HITs of the main study.

When a worker completes a HIT and submits her results, requesters can review and decide to either approve or reject the work. If a result gets rejected, the respective worker does not receive any money.

There are several HIT templates with predefined layouts and controls available, but it is also possible to use custom applications hosted by the requester. To conduct the studies for this thesis, a tool was developed and used to create and manage custom HITs (see Chapter 7).

8.2 Model Evaluation Study

Evaluating multi-modal models on text-image retrieval is usually done with the standard information retrieval metric Recall@k. This is an exact and binary metric and measures the proportion of retrieved relevant documents, i.e., images in the case of text-image retrieval, in the top-k ranked images. Recall@k is defined as:

$$\mathrm{R@}k = \frac{|RI \cap TKI|}{|RI|} \tag{8.1}$$

where RI is the set of relevant images for a query q and TKI is the set of top-k retrieved images by the model for q.

The metric is called exact or binary because an image can only be regarded as relevant or irrelevant – nothing in between. So in the case of text-image retrieval, Recall@k measures how often the relevant image was retrieved in the top-k results for the whole test set. In

other words: It measures the percentage of queries of the test set where the model could retrieve the relevant image in the top-k results.

For each textual query in a typical evaluation or test set, e.g., in COCO, Flickr30k, or WISMIR, only a single image is defined relevant. That is, $|RI| = 1$ and contains the image of the respective sample. The problem with the metric in an information retrieval system is that other images from the pool of images might also be relevant. For example, consider the query "A dog playing in the grass" and the top-5 retrieved images shown in Figure 8.1, where only the last image with the green frame is the correct image from the test set. The Recall@k metric will be zero for all $k <= 4$ and only be 1 for $k >= 5$, resulting in misleading evaluation scores.

Figure 8.1: Possible ranking of images of a text-image retrieval model for the query "A dog playing in the grass". All images are taken from https://unsplash.com/.

However, all of the shown images are relevant to the query. This problem is usually solved using non-binary metrics like Discounted Cumulative Gain (DCG) or Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2002) to evaluate information retrieval systems. These metrics can consider continuous or binary relevance scores for the documents or images, as well as their position in the retrieved list of top-k documents for a query. The DCG@k is defined as:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{rel_i}{ld(i+1)} \tag{8.2}$$

where rel_i is the relevance score of the i-th ranked image for the query. The discounting factor $ld(i+1) = log_2(i+1)$ accounts for the position of the retrieved image, i.e., lower-ranked images have a higher penalty on their relevance score. The NDCG@k is defined as:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{8.3}$$

where IDCG@k is the DCG@k of the ideal ranking.

When evaluating text-image retrieval models on the WISMIR test set, the issue is that neither the relevance scores of the images for the queries nor the ideal rankings are known. In Nicola et al. 2020 and Messina et al. 2021, where text-image retrieval models are evaluated on COCO and Flickr30k test sets, the authors introduced a way to compute NDCG by computing relevance scores based on the captions of an image. However, this approach is only possible because, in COCO and Flickr30k, each image has five associated captions. In WISMIR, this is not the case, as one image only has one associated caption.

The aim of the user study described in the following is to let humans assess the performance of a TERAN model trained on our WISMIR dataset in addition to the exact evaluation as described in Section 6.2. Further, the study seeks to assess the suitability of the evaluated TERAN model in a multi-modal text-image retrieval system as described, e.g., in our language learner use case in Section 1.1.

8.2.1 Pilot Tasks

Since the main study is planned to be conducted on MTurk, a pilot study was done to find genuine workers that understand the tool and the task well. This is a necessary and

common step for MTurk-based studies to obtain qualitative results.

Control Questions

A worker's task in the main study is to rank her top images according to a provided caption text. Therefore, three control questions were designed.

In the first control question, 12 images from which 5 depict dogs, and 7 depict trains, with the caption "*The dog has been selectively bred over millennia for various behaviors, sensory capabilities and physical attributes. Dogs are subclassified into breeds, which vary widely in shape, size and color.*" are shown to a worker (see Figure 8.2). The task then is to rank the top-5 images related to the caption. The perfect expected result is that a worker only ranks the 5 images depicting dogs.

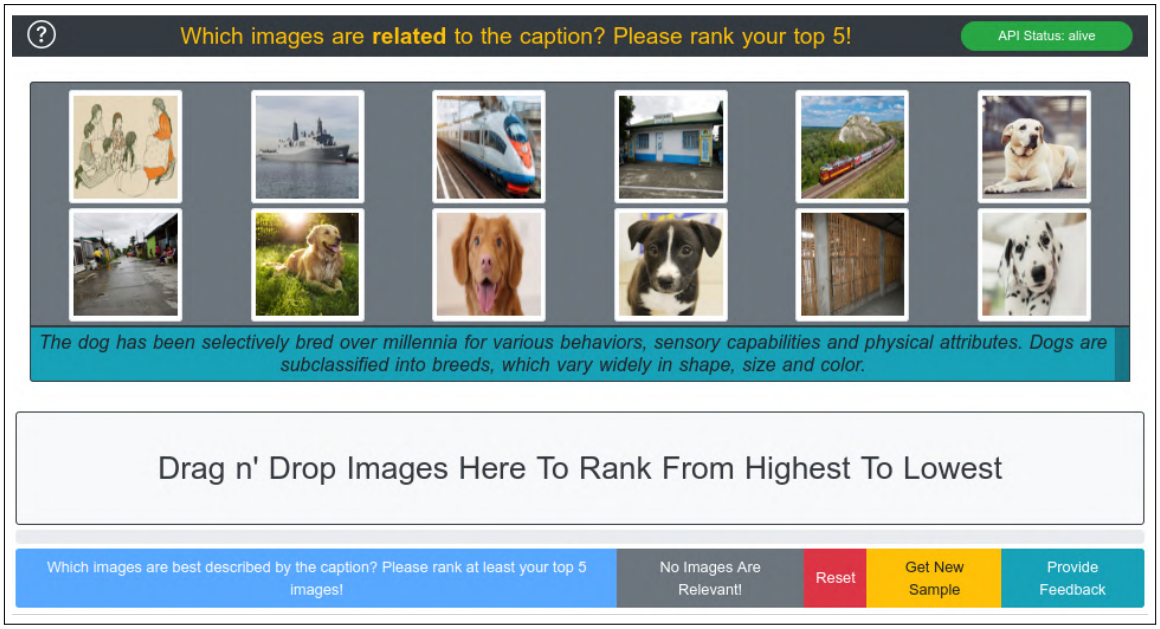


Figure 8.2: Screenshot of the IRST application showing the "dog" control question of the model evaluation pilot study. Here, a worker is expected to rank only the 5 images depicting dogs in an arbitrary order.

The second control question, shown in Figure 8.3, has the same task but is the other way round. From the 12 provided images, 5 images depict trains, and 7 depict dogs. The caption is: "*A train is a form of rail transport consisting of a series of connected vehicles that generally run along a railroad (or railway) track to transport passengers or cargo (also known as freightör "goods"). The word "train" comes from the Old French trahiner, derived from the Latin trahere meaning "to pull" or "to draw".*" The perfect expected result is that a worker only ranks the 5 images depicting trains.

The task of the third control question (see Figure 8.4) is the same as in the others, but the 12 images depict dogs or random objects excluding trees or things related to trees. The corresponding caption is: "*In botany, a tree is a perennial plant with an elongated stem, or trunk, supporting branches and leaves in most species. In some usages, the definition of a tree may be narrower, including only woody plants with secondary growth, plants that are usable as lumber or plants above a specified height.*". Here, I expected the workers to use the "No Images Are Relevant!" button to submit an empty ranking, indicating that none of the images is relevant to the provided caption. Unfortunately, this third control question was not published and used in the pilot study.

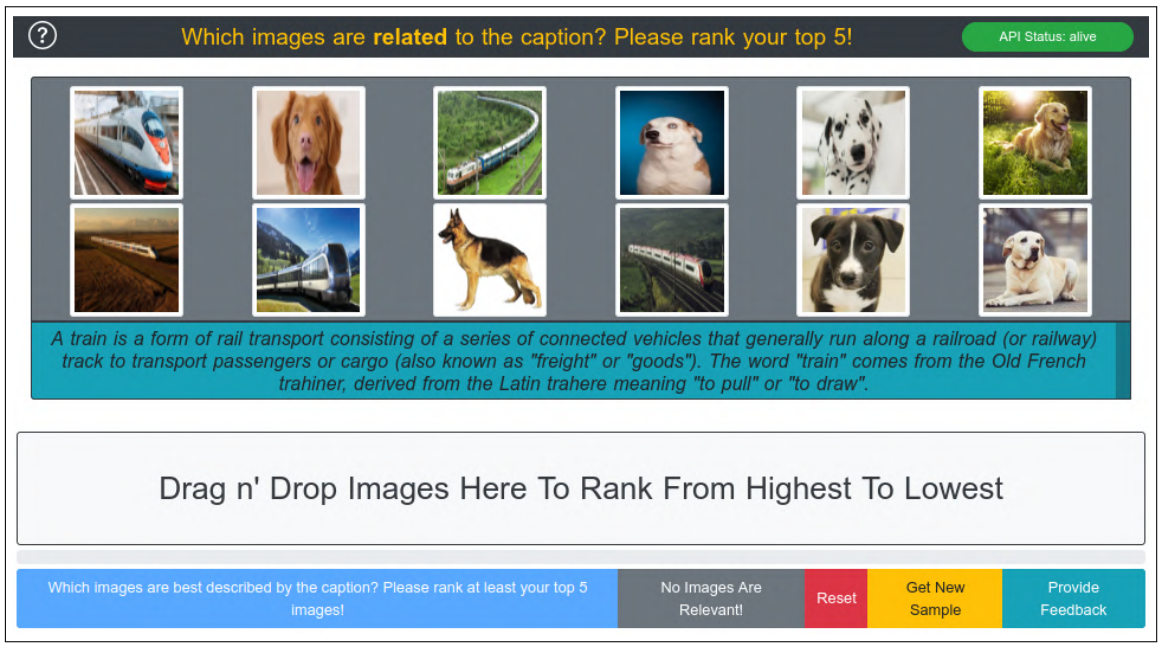


Figure 8.3: Screenshot of the IRST application showing the "train" control question of the model evaluation pilot study. Here, a worker is expected to rank only the 5 images depicting trains in an arbitrary order.

MTurk HIT Configuration

Using IRST, the "dog" and "train" control questions were published as HITs on MTurk with 100 Assignments each, resulting in a total of 200 HITs. To accept and work on one of the HITs, a user needs to have at least 500 approved Assignments with an approval rate of 80% With the reward per HIT set to 0.15$ and the MTurk fee of 20%, the total cost of the pilot study was 36.00$.

8.2.2 Pilot Results

To identify suitable workers for the main study, four classes of results described in Table 8.1 were defined.

Class	Description
A	The ranking contains only the five 5 images depicting dogs or trains, depending on the control question.
B	The ranking contains all 12 images, but the top-5 images depict the dog or trains, depending on the control question.
C	The ranking contains less than 12 images, but the top-5 images depict the dog or trains, depending on the control question.
D	The ranking does not contain the 5 images depicting the dogs or trains, depending on the control question.

Table 8.1: Four defined classes for results of the model evaluation pilot study to identify genuine workers for the main study.

From 114 unique workers, who submitted results, 86 worked on both of the tasks. Figure 8.5 shows how those 86 workers performed on the two control questions.

The large majority of workers, that is 60 workers, submitted class A results, and 4 submitted class A or B results for both of the control questions. Hence, one pilot study run was enough, and the 64 workers were chosen for the main study.

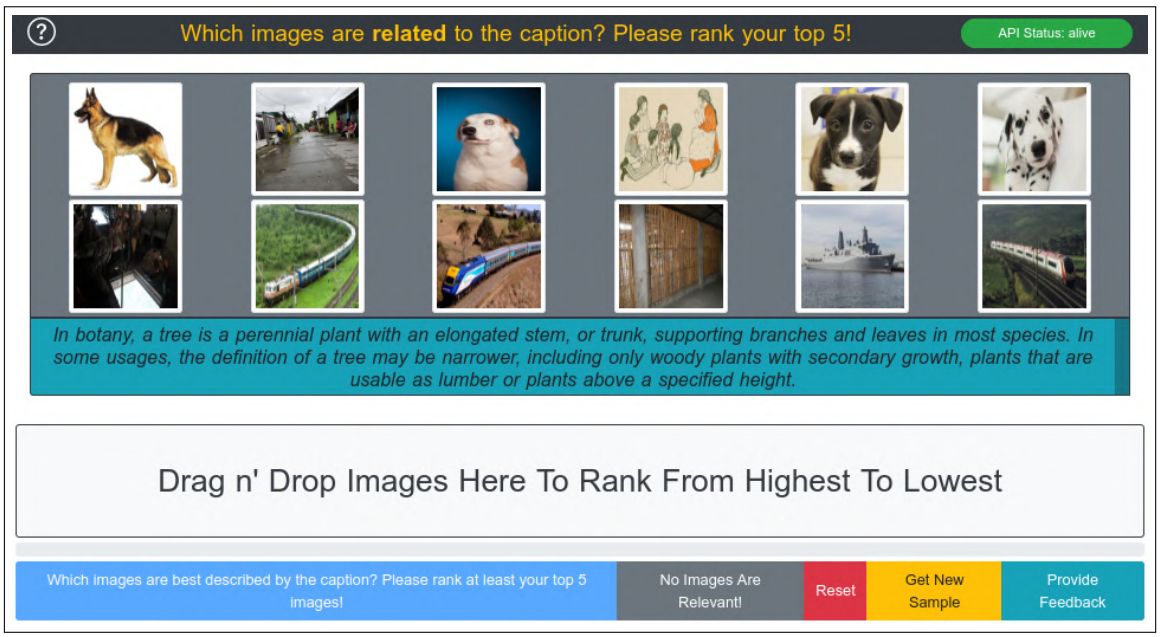


Figure 8.4: Screenshot of the IRST application showing the "tree" control question of the model evaluation pilot study. Here, since not a single image is related to trees, a worker is expected to use the gray "No Images Are Relevant!" button.

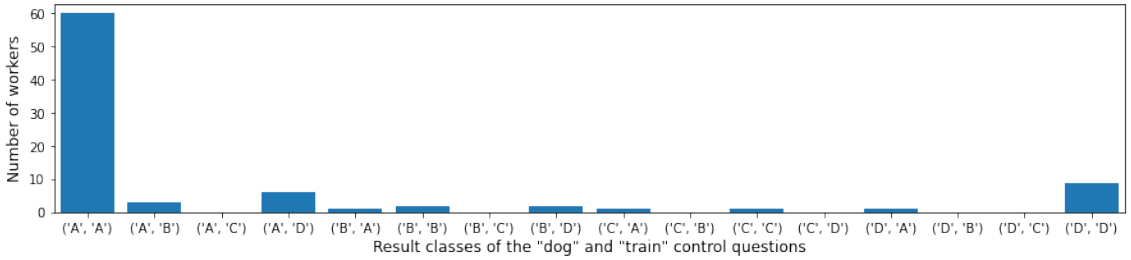
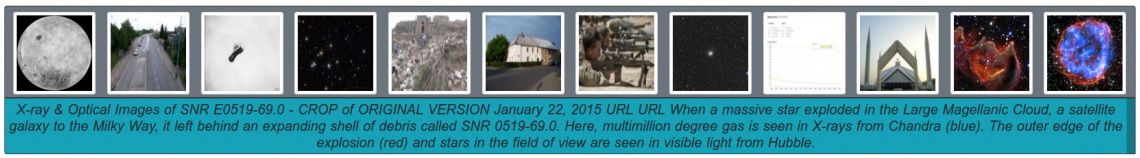


Figure 8.5: Bar chart showing how many workers achieved which result classes on the two control questions of the model evaluation pilot study. On the x-axis, the first letter beneath each bar is the result class of the "dog" control question, and the second letter indicates the result class of the "train" control question.

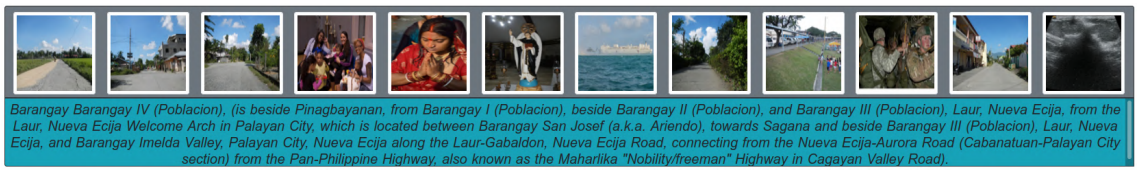
8.2.3 Main Tasks

The workers in the main study had the same task as in the pilot study, i.e., they had to choose and rank their top-5 matching images regarding the provided caption from a pool of 12 images. From those 12 images, 5 images are the top-5 ranked images from a TERAN model trained on WISMIR v2 regarding the caption of the respective sample from the test-set. The remaining 7 images are randomly selected images from the WISMIR v2 test-set. For the study, 50 samples were chosen at random and published as HITs on the MTurk marketplace utilizing IRST in MTurk mode with the ranking study method (see Chapter 7). For each HIT, three assignments were requested, resulting in a total number of 150 HITs. With the reward set to 0.15$ and the MTurk fee of 20%, the study cost 27$. The 64 selected workers from the pilot study were informed that they were elected to participate and that the HITs got published. A custom MTurk qualification requirement was created and assigned to each of the workers so that only those, instead of all workers, could accept and work on the HITs.

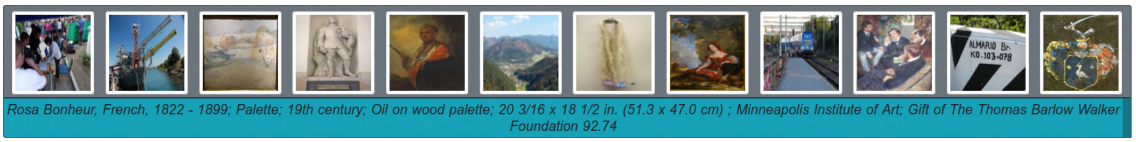
To get an impression of the tasks, 10% of the tasks are shown in Figure 8.6.



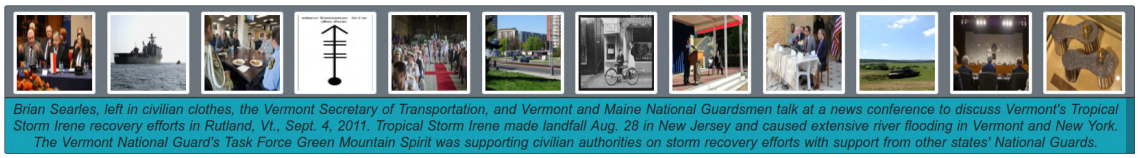
(a)



(b)



(c)



(d)

(e)

Figure 8.6: 5 of the 50 tasks of the model evaluation main study. Note that only the images and the captions instead of the full IRST user interface (see e.g. Figure 8.2) are depicted. Best viewed in color and digital form with a zoom of 200% or more.

8.2.4 Main Results

The purpose of this study was to assess the quality of text-image retrieval results of a TERAN model trained and evaluated on WISMIR v2 by humans. Therefore, the study results were evaluated according to different metrics.Note that in this section, only the results are presented without further discussion. An interpretation of the results and the outcomes of the study are elucidated in the summary Section 8.2.5.

In the following, a single ranking predicted by the model, consisting of the top-5 ranked images and the corresponding caption text, is referred to as a sample. For each sample, there are three assignments from three different workers.

Percentage Agreements

Since there are several ways to compute and interpret an agreement, first, it must be defined what "agreement" means specifically. Two different meanings for the percentage agreement between workers and the model were defined and reported below.

In the succeeding equations that formally define the different agreement measures, W is the set of workers ($|\mathrm{W}| = 3$); δ is the Kronecker-Delta; $\mathrm{I}_{i,k}^{(w)}$ is the k-th ranked image by worker w in sample i; and $\mathrm{M}_{i,k}$ is the k-th ranked image by the model in sample i.

Agreement Definition 1: First, the agreement of the top-5 ranked images of the model and the unordered set of the top-5 ranked images of the workers is considered. This agreement is defined as the proportion of workers that ranked the k-th image of the model as any of their top-5 images. Or, in other words, the proportion of workers who agreed that the k-th ranked image by the model is among the top-5 images according to their corresponding captions. This concept of agreement is visualized in Figure 8.7. In Equation 8.4, the measure is defined formally for the k-th image of sample i ranked by the model. Equation 8.5 specifies the average of the agreement for all top-5 ranks of sample i.

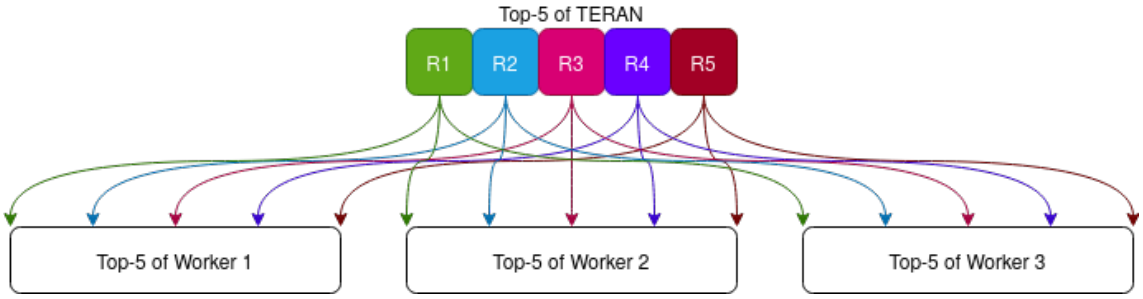


Figure 8.7: Visualization of the concept of agreement between three workers and the TERAN for a sample i, where the ordering of the top-5 rankings of the workers is ignored. Each image of the model's top-5 rankings can be contained in one, two, three, or none of the workers' top-5 images. Hence, in total there are $3+1$ levels of agreement per rank of a sample (see Equation 8.4), resulting in a total of $15+1$ levels of agreement per sample across all ranks (see Equation 8.5). The additional $+1$ level comes from including 0.

$$agr_i^{(1)}(k) = \frac{1}{3} \sum_{w \in \mathrm{W}} \sum_{j=1}^{5} \delta \left(\mathrm{I}_{i,j}^{(w)}, \mathrm{M}_{i,k} \right) \tag{8.4}$$

$$agr_i^{(2)} = \frac{1}{5} \sum_{j=1}^{5} agr_i^{(1)} \tag{8.5}$$

The resulting average agreement per rank across all samples between the TERAN model and the workers according to Equation 8.4 is shown in Figure 8.8.

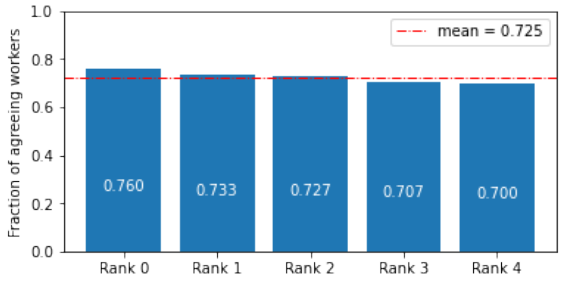


Figure 8.8: The average across all samples of the proportion of workers who agreed that the k-th image in the model's ranking is among the top-5 relevant images according to their corresponding captions. This agreement is calculated based on the definition in Equation 8.4 and illustrated by Figure 8.7.

Figure 8.9 depicts the average agreement between the model and the workers per sample for all ranks according to Equation 8.5.

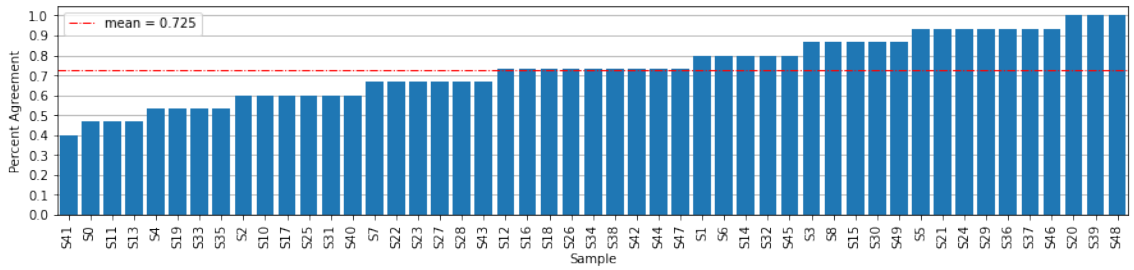


Figure 8.9: The average agreement per sample between the TERAN model and the workers according to Equation 8.5 and illustrated by Figure 8.7.

Agreement Definition 2: The second agreement is defined analog to the first definition, but instead of ignoring the ordering of the workers' top-5 images, their ordering is considered. That is, the model's and workers' exact rankings are compared. Hence, the agreement on the k-th image in the model's ranking for sample i is defined as the proportion of workers that ranked the same image on place k. This concept of agreement is visualized in Figure 8.10. This agreement is formally described by Equation 8.6. Equation 8.7 specifies

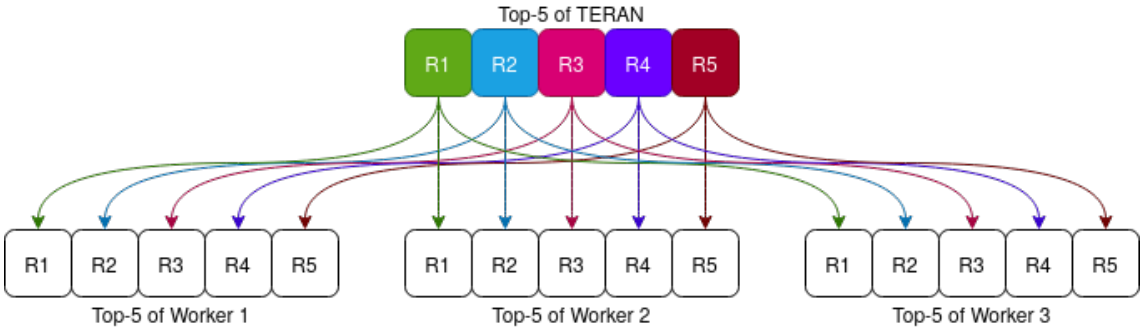


Figure 8.10: Visualization of the concept of agreement between three workers and the TERAN for a sample i, where the ordering of the top-5 rankings of the workers is taken into account. Each k-th ranked image of the model can be equal to the k-th ranked image of one, two, three, or none of the workers. Hence, in total there are $3 + 1$ levels of agreement per rank of a sample (see Equation 8.6), resulting in a total of $15 + 1$ levels of agreement per sample across all ranks (see Equation 8.7). The additional $+1$ level comes from including 0.

the average of the agreement for all top-5 ranks of a sample i.

$$agr_i^{(3)}(k) = \frac{1}{3} \sum_{w \in \mathrm{W}} \delta \left(\mathrm{I}_{i,k}^{(w)}, \mathrm{M}_{i,k} \right) \tag{8.6}$$

$$agr_i^{(4)} = \frac{1}{5} \sum_{j=1}^{5} agr_i^{(3)} \tag{8.7}$$

Figure 8.8 shows the average agreement per rank for all samples between the TERAN model trained on WISMIR v2 and the workers according to Equation 8.4.

The average agreement per sample for all ranks between the model and the workers according to Equation 8.7 is depicted Figure 8.12.

Normalized Discounted Cummulative Gain Metric

To determine the NDCG (see Equation 8.3), which evaluates the model's performance on a single sample, the necessary relevance scores of the images are calculated based on the

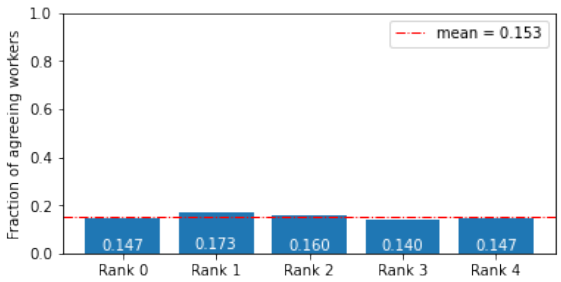


Figure 8.11: The average across all samples of the proportion of workers who agreed with the model on the k-th ranked image. The agreement is calculated based on the definition in Equation 8.6
and illustrated by Figure 8.10

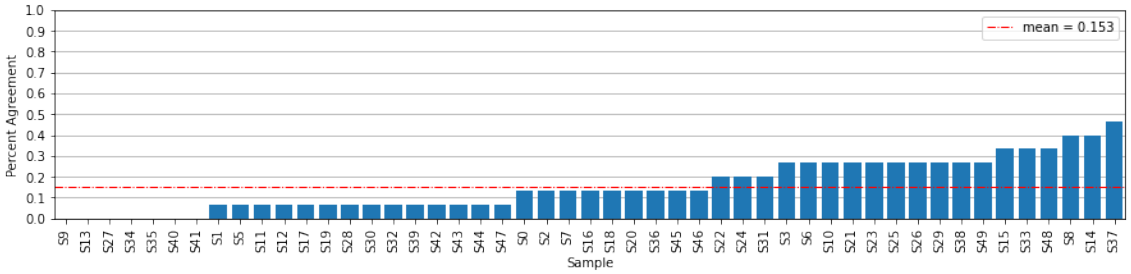


Figure 8.12: The average agreement per sample between the TERAN model and the workers according to Equation 8.7 and illustrated by Figure 8.10.

rankings of the workers. The overall performance of the TERAN model is determined by the average NDCG across all samples of the study.

Similar to the agreement definitions between the workers and the TERAN model, multiple ways to calculate the relevance scores for an image j of sample i are by the following formulas and get explained in Table 8.2.

$$rel_{i,j}^{(1)} = \begin{cases} 1 & \text{if } j \in \bigcap_{k \in \mathrm{W}} \mathrm{T5}_k \\ 0 & \text{otherwise} \end{cases} \tag{8.8}$$

$$rel_{i,j}^{(2)} = \begin{cases} 1 & \text{if } j \in \bigcup \{\mathrm{T5}_k \cap \mathrm{T5}_l \mid k, l \in \mathrm{W} \wedge k \neq l\} \\ 0 & \text{otherwise} \end{cases} \tag{8.9}$$

$$rel_{i,j}^{(3)} = \begin{cases} 1 & \text{if } j \in \bigcup_{k \in \mathrm{W}} \mathrm{T5}_k \\ 0 & \text{otherwise} \end{cases} \tag{8.10}$$

$$rel_{i,j}^{(4)} = \frac{1}{3} \sum_{k \in \mathrm{W}} \sum_{l \in \mathrm{T5}_k} \delta_{j,l} \tag{8.11}$$

$$rel_{i,j}^{(5)} = \frac{5 + 1 - \frac{\sum_{k \in \mathrm{W}} \mathrm{rank}(i, \mathrm{T5}_k)}{3}}{5} = \frac{1}{15} \left(18 - \sum_{k \in \mathrm{W}} \mathrm{rank}(i, \mathrm{T5}_k) \right) \tag{8.12}$$

where i and j are images from the model's top-5; W is the set of workers per sample ($|\mathrm{W}| = 3$); $\mathrm{T5}_k$ is the ordered set of the top-5 rankings of a worker k; δ is the Kronecker Delta; and $\mathrm{rank}(i, \mathrm{T5}_k)$ denotes the rank or position of an image i in the top-5 rankings of a worker k.

The binary relevance scores for an image j or sample j defined by Equation 8.10, Equation 8.9, and Equation 8.8, are illustrated in Figure 8.13.

To better understand the non-binary relevance scores for an image j of sample j defined by Equation 8.11 and Equation 8.12, Figure 8.7 and Figure 8.10 are supportive, when only

Abbreviation	Formula	Description
BinAgr3W	8.8	Binary relevance score for image j in sample i. $rel_{i,j}^{(1)} = 1$ if all three workers ranked i in their top-5.
BinAgr2W	8.9	Binary relevance score for image j in sample i. $rel_{i,j}^{(2)} = 1$ if two of the three workers ranked i in their top-5.
BinAgr1W	8.10	Binary relevance score for image j in sample i. $rel_{i,j}^{(3)} = 1$ if at least one of the three workers ranked i in their top-5.
NonBinIO	8.11	Non-binary relevance score for image j in sample i. $rel_{i,j}^{(4)} \in [0, 1]$ is the fraction of workers which ranked i in their top-5 ignoring the ordering.
NonBinCO	8.12	Non-binary relevance score for image j in sample i. $rel_{i,j}^{(5)} \in [0, 1]$ is the normalized average rank of i in the top-5 rankings of the workers (considering the order!).

Table 8.2: Description of different methods to compute the relevance score rel_i on an image i utilized for the NDCG metrics to evaluate the text-image retrieval performance of the TERAN model trained and evaluated on WISMIR v2.

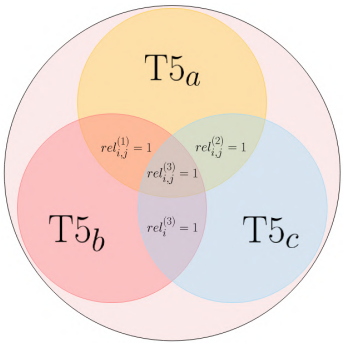


Figure 8.13: Venn-diagram showing the set of the top-5 rankings of the three different workers per sample j, and where the binary relevance scores for an image i (defined in Equation 8.10, Equation 8.9, and Equation 8.8) are equal to 1.

a single rank of the model is focussed.

In addition to the relevance scores, computing the NDCG requires knowing the IDCG or ideal DCG, which is usually the DCG of the descending sorted images according to their respective relevance. Besides the standard definition of the IDCG, the MIDCG or maximum IDCG is introduced in this work and defined in the following. Since the relevance scores per image described in Table 8.2 are normalized or binary so that $\forall(i, j)\ rel_{i,j} \in [0, 1]$, the maximum possible IDCG is:

$$\text{MIDCG@}k = \sum_{i=1}^{k} \frac{\max_{\forall j} rel_j}{ld(i+1)} = \sum_{i=1}^{k} \frac{1}{ld(i+1)} \tag{8.13}$$

The MIDCG is the IDCG that would result if the model predicted the perfect ranking for every sample, i.e., if all workers would always agree on every rank of every sample.

Figure 8.14 shows the mean NDCG@5 of all samples computed with different relevance scores and standard IDCG@5 or maximum IDCG@5.

Agreement Among The Workers

There exist multiple metrics to measure the agreement among the workers, also known as inter-rater agreement metrics. The Fleiss-Kappa metric measures the agreement between multiple raters, where agreement due to chance is factored out. To measure the workers' agreement with the Fleiss-Kappa metric, the study is interpreted so that the workers judge or classify each of the top-5 ranks with an image. Table 8.15 shows the inter-rater

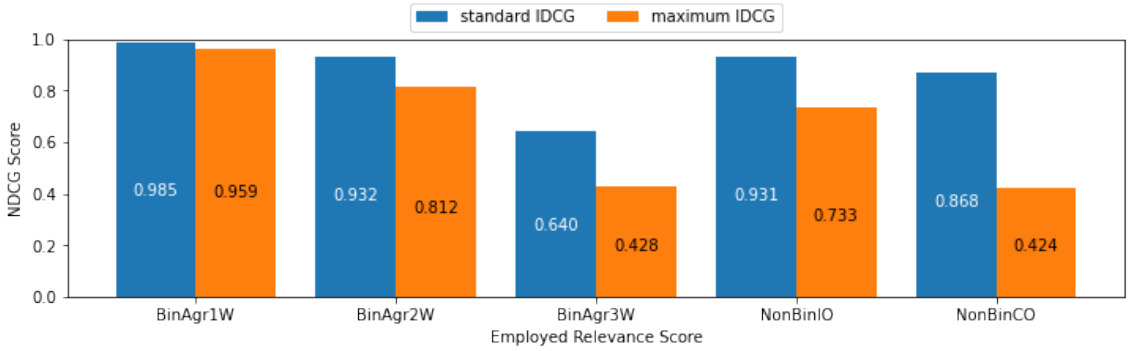


Figure 8.14: The mean NDCG@5 of all samples computed with different relevance scores (see Table 8.2) and standard IDCG@5 or maximum IDCG@5 (see Equation 8.13)

agreement of each of the top-5 ranks and additionally all ranks together according to the Fleiss-Kappa.

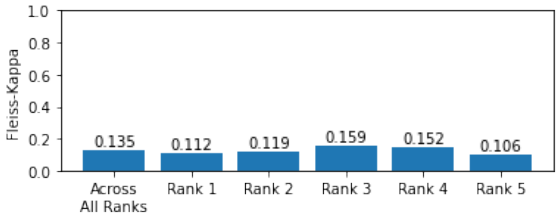


Figure 8.15: Inter-rater agreement according to Fleiss-Kappa among the workers between the top-5 ranks of each sample in the model evaluation Study.

In addition to the Fleiss-Kappa to measure the agreement among the raters per rank, the intraclass correlation coefficient (ICC) was computed to measure the workers' agreement per sample. From the ten different definitions of the ICC as reported in Koo and Li 2016, the ICC2k is of interest. This is because the raters are randomly selected, each sample is rated by three different raters also chosen randomly, and I am interested in the absolute agreement based on the mean of the three workers. To compute the ICC2k, a score per sample per worker that serves as the rating of the worker for the sample was calculated as follows:

$$score(i, w) = |\mathrm{T5}_m \cap \mathrm{T5}_w| \tag{8.14}$$

where i is the i-th sample, w is a worker, $\mathrm{T5}_x$ is the set of top-5 ranked images by either a worker w or the TERAN model m, and $score(i, w) \in \{0, 1, 2, 3, 4, 5\}$.

The resulting ICC2k, computed with pingouin[1], is 0.43, the p-value is 0.009, and the 95% confidence interval is $C95 = [0.09, 0.66]$. The p-value indicates the statistical relevance of the computed inter-rater agreement. The wide confidence interval, which indicates the opposite, i.e., that the statistical significance is relatively low, can be explained by the small sample size. According to Koo and Li 2016, the ICC2k of 0.43 indicate poor study reliability and agreement among the workers. However, according to Cicchetti 1994, values below 0.4 and values between 0.4 and 0.5 indicate fair agreement among the raters.

8.2.5 Summary

Results Discussion

Percentage Agreement By looking at Figure 8.8 and Figure 8.11, which show the average

1. https://pingouin-stats.org

percentage agreement per rank across all samples between the workers and TERAN, it can be observed that the average agreement drops significantly from 72.5%, when the ordering of the model's top-5 ranked images is ignored, to 15.3%, when the ordering is considered. This severe decrease in the percentage agreement is also shown by Figure 8.9 and Figure 8.12, which depict the average agreement across all ranks per sample.

In Figure 8.16, Sample $S39$ and the workers' results are illustrated to highlight the resulting difference in the percentage agreement. Sample $S39$, serves as a good example to

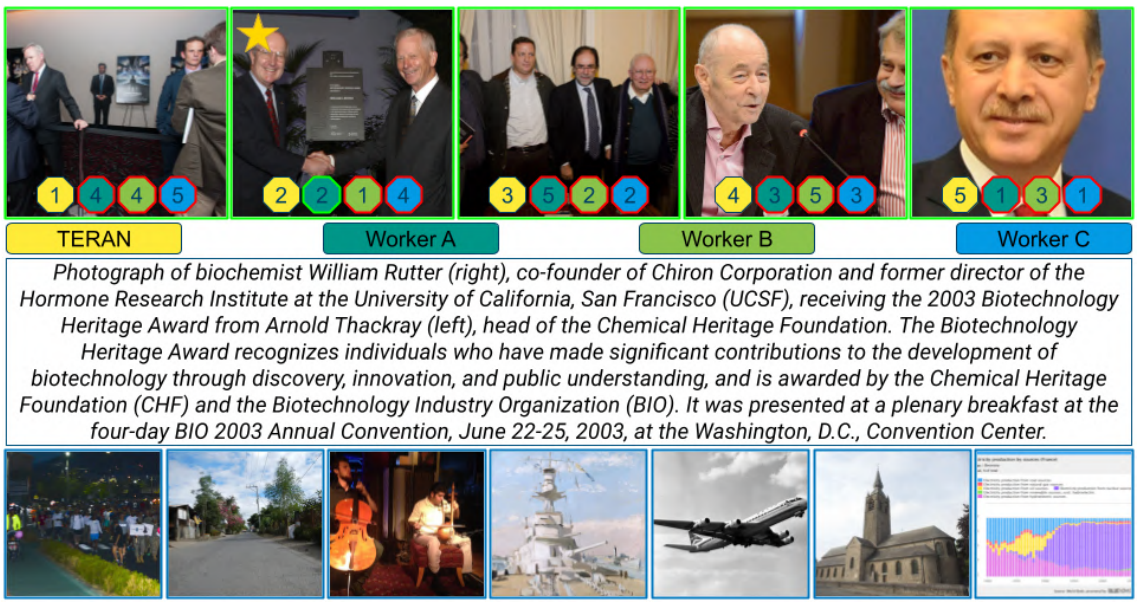


Figure 8.16: Visualization of Sample $S39$ of the model evaluation main study and corresponding ranking results from the TERAN model and the workers. At the top, the model's top-5 ranked images are shown, the 7 random images of the sample are shown at the bottom. The image with the star at the top-right is the gold label from the WISMIR test set. In the middle, the caption for which the model retrieved the top 5 images is shown. The green border around the top 5 images indicated that all the workers agreed on the image in any of their top-5 ranked images. The colored plates at the bottom of the images show how the workers ranked the respective image. If the plate has a green border, the worker and the model chose the respective image on the same rank. If the border is red, the worker and the model disagree on the rank of the respective image.

show the impact of ignoring or considering the exact ordering of the model's top-5 ranked images when measuring the percentage agreement. When ignoring the exact ordering, the percentage agreement score associated with this sample is 100% but drops to 6.67% when the workers have to agree on the images' exact ranks with the model (see Figure 8.9 and Figure 8.12).

What can be observed from Figure 8.8 is that there is only a slight decline in the average agreement across all samples from the rank 1 to rank 5. This finding is expected and further indicates that the model's top-ranked images are only marginally different.

NDCG Scores. The influence of considering or ignoring the ordering of the model's top-5 ranked images to evaluate the model's performance is also reflected in the NDCG@5 scores computed with the non-binary relevance scores from Equation 8.12 and Equation 8.11. While the differences in the corresponding NDCG@5 based on standard IDCG@5 are not as significant as in the percentage agreement measures, still, the consideration of ordering causes an absolute drop by 6.3%. In the NDCG@5 scores based on the MIDCG@5 (see Equation 8.13), however, the caused decrease of 30.9% in the scores is more significant.

This difference can be explained by the definition of the MIDCG, which would result if the model predicted the perfect ranks for all images of all samples according to every worker.

The NDCG@5 shown in Figure 8.14 and computed with the binary relevance scores from Equation 8.10, Equation 8.9, or Equation 8.8 indicate a certain disagreement on the top-5 ranked images among the workers. When the binary relevance of an image is 1 if only at least one worker has to vote for it, the NDCG@5 is close to perfect with 98.5%, and 95.9% for standard IDCG or MIDCG utilized, respectively. If, however, all three workers' vote is required for the binary relevance of an image becoming 1, the respective scores drop to 64.0% and 42.8%. The inter-rater agreement is measured and explicitly reported in the following.

Inter-Rater Agreement. One observation reflected in the reported percentage agreement measures and NDCG scores is that there is only little consistency between the workers' rankings, especially when contemplating the images on the single ranks. The low Fleiss-Kappa scores shown in Figure 8.15 as well as the reported ICC2k underline this finding. From the inter-rater agreement scores for the single ranks, it can further be observed that the highest agreement among the workers is for the third rank and not for the first rank. This finding, again, demonstrates that the model's top-5 images do not differ significantly from each other.

The reported low inter-rater agreements for the single ranks and across ranks per sample calculated with the Fleiss-Kappa and the ICC2k respectively indicate the difficulty and the subjectiveness of finding and ranking the best matching images for a given caption.

Limitations

Assignments Per Worker From the 64 workers, 13 participated in the main study, each of whom submitted a different number of results.

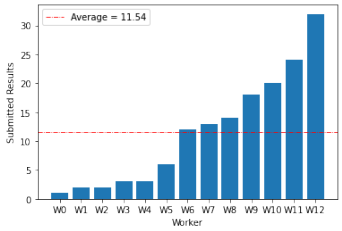


Figure 8.17: The number of workers participated in the study and the number of results they submitted.

As shown in Figure 8.17, the number of submitted results per worker varies a lot. This has the (unwanted) effect that the results are biased towards the opinions of the worker who submitted most of the results. In future studies, this should be fixed so that all workers submit the same number of results. If a study is conducted via MTurk, guaranteeing the same number of results per worker can only be accomplished by limiting the maximum submissions per worker.

Challenging Task From looking at the example shown in Figure 8.16 or the task in Figure 8.6 and the reported inter-rater agreement scores, it can be observed that finding and ranking the best matching images according to the respective caption is challenging even for humans. This is probably due to the averagely long and complex caption texts, which include many named entities and information unrelated to the image's content and require expert knowledge to grasp their meaning entirely. Another issue is that workers define and quantify the relatedness of an image individually, especially when comparing the relatedness of multiple images to a caption and ranking them accordingly. For some tasks like those shown in Figure 8.6a or Figure 8.6c, however, the top-5 matching images ranked by the model can be relatively clearly identified due to their coherent similarity, even though the caption text is hard to understand as a whole.

Study Design When the results of this study were collected, it was found that the study was not optimally designed for the following reasons: First, since the model's top-5

ranked images for a given query are usually very alike, it is easy for the workers to identify those between the other 7 random images only based on their inherent similarity. Second, the workers have to rank either five or more or zero images, i.e., they could not only choose, e.g., three images if only those were appropriate for the caption. Both reasons, especially combined, have the effect that the workers' top-5 are more likely to overlap with the top-5 of TERAN's ranked images, which results in a higher agreement per sample as if it might probably be. Further, to compare the results with the evaluation with the model computed as the R@k scores for all the 9380 samples of the WISMIR v2 test set, the sample size of this study is too small. It might likely be that the 50 selected samples do not represent the test set adequately.

Conclusion

Despite the limitations of this study, it can be concluded that the model does retrieve relevant top-5 images for a given caption text for the majority of the evaluated samples. This conclusion, however, is only valid if the unordered set of the top-5 ranked images by the model is considered, i.e., their exact ordering is not taken into account. Since the model's designated area of application is in our language learner scenario (see Section 1.1) and in an information retrieval system in general, where the exact ordering of the top-ranked images does not matter much, I conclude that the evaluated TERAN model is suitable for the use case.

Further, the average percentage agreement on the top-5 ranked images across all ranks and samples of 72.5 between the workers and the TERAN model and the reported NDCG@5 scores are more optimistic than the Recall@K evaluation metrics reported in Section 6.2.

8.3 L2 Language Learner Data Study

The purpose of this study is to evaluate the text-image retrieval performance of state-of-the-art multi-modal transformers on an unseen text-only language learner dataset. Further, the study seeks to gauge how the training data of the employed TERAN models and the image pool from which the images are retrieved affect the quality of the retrieved images concerning the relatedness to the corresponding textual samples. Since the employed dataset is purely textual, i.e., uni-modal, only human raters can assess the quality of the model's predicted top-ranked images.

8.3.1 L2 Language Learner Dataset

The uni-modal textual language learner (LL) dataset was collected manually by a student assistant and is based on InScript (Modi et al. 2016), which is a dataset containing scripts or stories around different topics and is originally thought to "study [...] the role of script knowledge in natural language processing". The student assistant hand-selected the samples according to their general depictability. That is, samples consisting of too many abstract and, therefore hardly depictable entities or concepts, are filtered out. In the following, two example texts are provided and statistics are reported in Figure 8.18 to get an impression on the employed dataset:

First Example:

> I decided to borrow some books from the library. I went to the library and started looking on the shelves. There were so many good books that it was difficult to pick out just a few, so I used the library 's computer book search program. Eventually I chose several modern plays, a biography, and a book on ancient Rome. I took these books up to the librarian 's desk and asked if I could check them out. She said yes, I could borrow the books, but I needed

to sign up for a library card. I filled out the form, and she took it. Then she made my library card and gave it to me. With this accomplished, I was able to check out my books. She put a card in each one with a date stamped on it. This was the date the books were due to be returned. I promised to return them on time and took the books home to read.

Second Example:

I travel to the local nursery to select a tree for planting, today I have decided to purchase a flowering maple tree. I ensure the one I pick is high quality with no rotting and looks healthy. Once in my yard I determine the best location based on the expected height it will grow and other plants in the yard. With my garden shovel I dig a hole at least twice the size of the roots of the tree. I ensure rocks and are removed from the hole. I put plenty of organic matter in the hole to ensure good growth. I gently place the tree in the hole and refill with the shoveled dirt. I place my garden hose at the base of the tree and turn the water to a very slow stream so it will absorb into the soil properly. I allow the water to run for about and hour ensuring it does n't get soggy. I step back and admire my new tree and envision what it will look like a year from now.

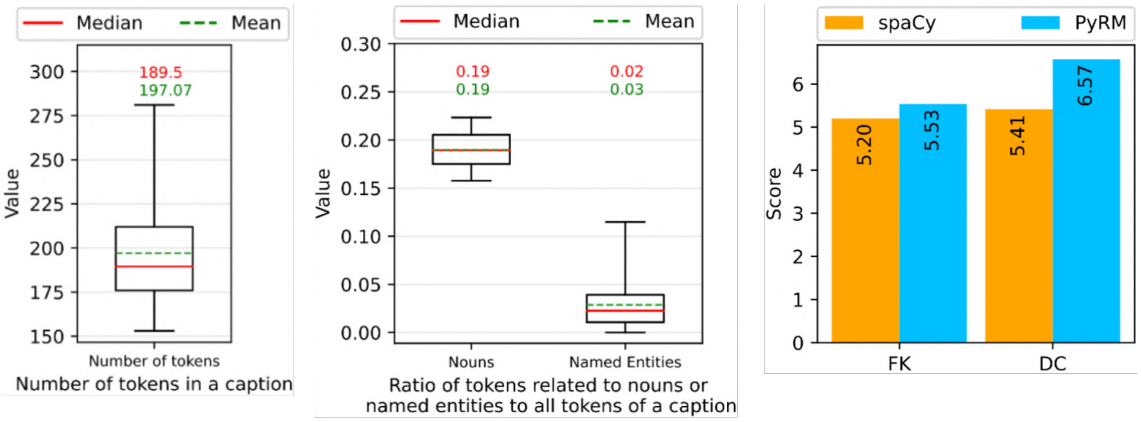


Figure 8.18: Box-plot diagrams for the number of tokens and the ratio of tokens tagged as nouns or named entities, and readability scores of the caption texts contained in our language learner dataset. The data represented by the box-plots were computed with spaCy. For the readability scores, FK stands for Flesch-Kincaid grade level, and DC stands for Dale-Chall and were computed by two different libraries, namely spaCy and py-readability-metrics.

From the statistics[2] shown in Figure 8.18, it can be observed that the LL dataset is more like COCO and Flickr30k. Although the average number of tokens per caption is much higher and even exceeds WISMIR v2 (see Section 5.3), there are almost no named entities in the LL dataset. Also, the average ratio of noun tokens in the LL dataset of 0.19 is closer to COCO (0.33) and Flickr30k (0.31) than to WISMIR v2 (0.46) (see Section 5.3.2).

Further, the overlap of the top-1000 frequent tokens shown in Table 8.3 between the datasets underlines the disparity of the LL dataset to WISMIR v2 and the similarity to COCO and Flickr30k.

The resemblance of the LL dataset to COCO and Flickr30k can be considered a bias in the data, so it is expected that the TERAN model trained on WISMIR v2 will perform worse than the TERAN models trained on COCO and Flickr30k.

2. The reported readability scores are popular metrics to measure the difficulty of a text in terms of the minimum U.S. grade level required to understand the text.

	WISMIR v2	Flickr30k	COCO	LL
WISMIR v2	100.0%	22.29%	18.93%	19.97%
Flickr30k		100.0%	64.95%	33.05%
COCO			100.0%	34.83%
LL				100.0%

Table 8.3: Overlap of the top-1000 frequent tokens of WISMIR v2, Flickr30k, COCO, and the language learner dataset (LL) used in this study.

If this expectation is reflected in the results of this study, it indicates that the textual part of the training data does affect the performance of the multi-modal text-image retrieval results.

8.3.2 Pilot Study

Similar to the model evaluation user study, a pilot study was conducted to find workers who correctly understood the task.

MTurk HIT Configuration

Using IRST, the three control questions were published as HITs on MTurk with 100 assignments each, resulting in a total of 300 HITs. To accept and work on one of the HITs, a user needs to have at least 1000 approved assignments with an approval rate of 90%. With the reward per HIT set to 0.10$ and the MTurk fee of 20%, the total cost of the pilot study was 36.00$.

Control Questions

The data of the three control questions from the previous pilot study described in Section 8.2.1 was re-used. Instead of using the ranking method of IRST, the control questions were generated using the rating method. The three control questions, referred to "train-all", "tree-none", and "dog-mixed" are shown in Figure 8.19, Figure 8.20, and Figure 8.21, respectively. The "train-all" control question consists of 5 images of a train and a caption about trains, and the workers are expected to rate all of the images high. The "tree-none" control question consists of 5 random images unrelated to trees and a caption about trees, and the workers are expected to rate all of the images low. The "dog-mixed" control question consists of 3 images of dogs, 2 images of trains, and a caption only about dogs. The workers are expected to rate the dog images high and the train images low.

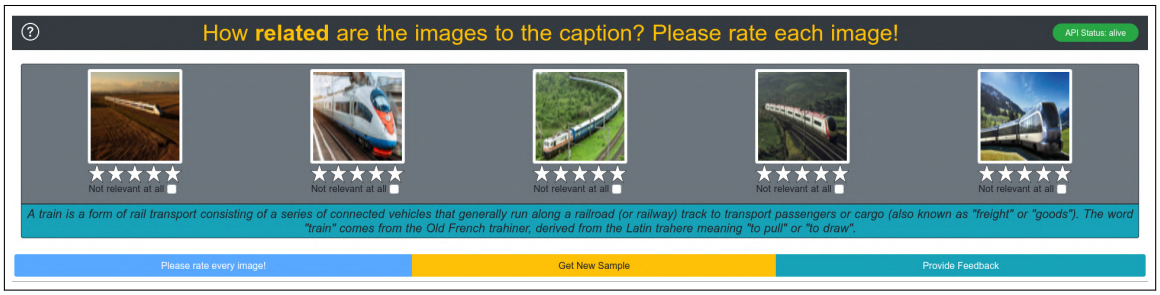


Figure 8.19: Cropped screenshot of the IRST application showing the "train-all" control question of the language learner data pilot study. Here, the workers are expected to rate all the images high.

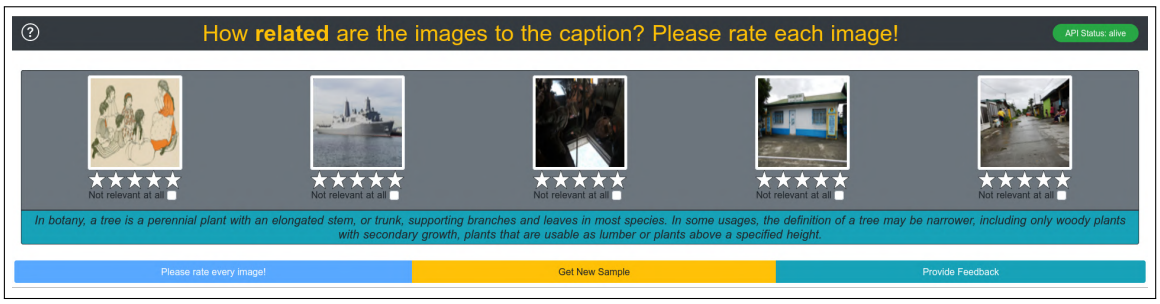


Figure 8.20: Cropped screenshot of the IRST application showing the "tree-none" control question of the language learner data pilot study. Here, the workers are expected to rate all the images low.

Figure 8.21: Cropped screenshot of the IRST application showing the "dog-mixed" control question of the model evaluation pilot study. Here, the workers are expected to rate the dog images high and the train images low.

Pilot Study Results

To identify suitable workers for the main study, three classes of results described in Table 8.4 were defined for each of the three control questions. For the main study, only

CQ	Class	Description
"train-all"	A	all images rated as 5
"train-all"	B	all images rated 4 or 4.5
"train-all"	C	all images rated lower than 4
"tree-none"	A	all images rated as 0
"tree-none"	B	all images rated 1 or 1.5
"tree-none"	C	all images rated higher than 1.5
"dog-mixed"	A	all dog images rated as 5 AND all train images rated as 0
"dog-mixed"	B	all dog images rated 4 or 4.5 AND all train images rated 1 or 0.5
"dog-mixed"	C	any dog image rated lower than 4 OR any train image rated 1.5 or higher

Table 8.4: Three classes of results for each of the three control questions (CQ) of the language learner pilot study.

workers who did not submit a class 'C' result for any of the control questions are accepted.

From 116 unique workers, who submitted results, 89 worked on both of the tasks. Figure 8.22 shows how those 89 workers performed on the three control questions. As is can be observed, the majority of workers, i.e., 76.4%, submitted a class 'C' result.The remaining 23.6% of the workers, who achieved exclusively class 'A' or 'B' results, were selected and notified to participate in the main study.

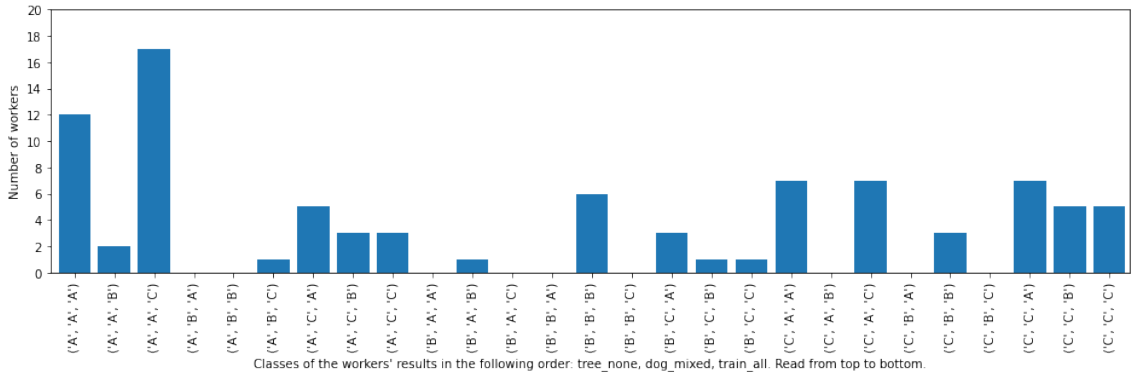


Figure 8.22: Bar chart showing how many workers achieved which result classes on the three control questions described in Table 8.4. On the x-axis, the first, second, and third letter beneath each bar represent the result classes of the "tree-none", "dog-mixed", and "train-all" control-questions, respectively.

8.3.3 Main Tasks

In the main study, the workers had the same task as in the pilot study. That is, they were shown the top-5 images retrieved by a particular TERAN model from a particular dataset and had to rate each of the images on a zero to five-star scale. The images were retrieved utilizing MMIRS and for texts from the language learner dataset described in Section 8.3.1, interpreted as queries.

Sample Subsets

To measure how the training data of the employed TERAN models and the image pools, i.e., datasets from which the models retrieve the top-k images, affect the quality of the retrieved images, several subsets of samples were evaluated. Each subset contains 100 samples consisting of a query or caption text, and the top 5 retrieved images by a TERAN model trained on WISMIR v2 (see Chapter 5), COCO (see Section 4.1.2), or Flickr30k (see Section 4.1.1) training set from the pool of unique images from either WISMIR v2, COCO, or Flickr30k.

The following naming convention applies to refer to one of the nine resulting subsets: "TERAN <TRAIN DATASET> from <IMAGE DATASET>". For example, "TERAN WISMIR2 from COCO" refers to the subset of samples containing the top 5 images retrieved by a TERAN model trained on the WISMIR v2 training set from the pool of unique images in the COCO dataset. To decrease the wordiness, when referring to one of the subsets or averages across subsets, the names are abbreviated, as shown in Table 8.5.

IDS \ **TDS**	Flickr30k	COCO	WISMIR	Any
Flickr30k	TFF	TCF	TWF	TAF
COCO	TFC	TCC	TWC	TAC
WISMIR v2	TFW	TCW	TWW	TAW
Any	TFA	TCA	TWA	TAA

Table 8.5: Abbreviations of sample subsets and averages across sample subsets from the language learner user study (see Section 8.3.3). The column and row specifiers "**TDS**" and "**IDS**" refer to the **t**raining **d**at**s**et of the TERAN model and the **i**mage **d**ata**s**et from which the considered TERAN model retrieved the images, respectively. The columns and rows showing "Any" are averages across the sample subsets of all the columns beside or rows beneath.

Utilizing IRST (see Chapter 7), the study tasks were published with three assignments per HIT, resulting in $3 * 900 = 2700$ HITs. With the reward per HIT set to 0.15$ and the MTurk fee of 20%, the total cost of the study was 486$. The HITs were published in batches, and the 21 elected workers from the pilot study were notified before every batch. To restrict access to the HITs, a custom MTurk qualification requirement was associated with these workers using IRST.

8.3.4 Main Task Results

In this section, the results of the main study tasks are reported. The discussion of the results and conclusion of the study are reported in the following Section 8.3.5.

For each of the nine sample subsets introduced in Section 8.3.3, the distribution of the rating stars and the average ratings per rank were collected. The collected data is also averaged across models or datasets to assess the quality of the retrieved images, independent of the retrieval models or datasets from which the images are retrieved.

Note, that the following figures Figure 8.23, Figure 8.24, and Figure 8.25 all are based on the same underlying data but provide different views.

Distribution Of Rating Stars

The black-bordered subplots depicted in Figure 8.23 show the rating star distributions across all ranks for each sample subset. Each bar chart of the black-bordered subplots shows data collected from the 300 results of the 100 samples contained in the respective subset indicated by the subplot title. The subplots outlined in red show the average across models or datasets from 900 results, with the upper left subplot depicting the overall average across all models and all datasets from all of the 2700 results.

Average Ratings Per Rank

Similarly to the plots from Figure 8.23, in Figure 8.24, the subplots outlined in black depict the average rating of the model's top-5 ranked images across all samples of the respective subset indicated by the subplot title. Each of these bar charts shows data averaged from the 300 ratings of the subset's samples. The subplots outlined in red show the averages across models or datasets from 900 results, with the upper left subplot containing the overall average from all of the 2700 results.

For a direct comparison of all averages shown in the subplots of Figure 8.24, the mean values have been plotted on a number line and are shown in Figure 8.25.

Inter-Rater Agreement

The agreement among the raters per rank and for all ranks was measured via the intraclass correlation coefficient (ICC) across all the 2700 rating results. As with the model evaluation study (see Section 8.2.4) and according to Koo and Li 2016, the ICC2k is of interest because:

1. the star rating data is ordinal
2. the raters are randomly selected
3. each sample is rated by three different raters also chosen randomly
4. the absolute agreement, based on the mean of the three workers, is considered

The resulting ICC2k agreement scores, together with their corresponding p-Value and 95% confidence intervals, are reported in Table 8.6.

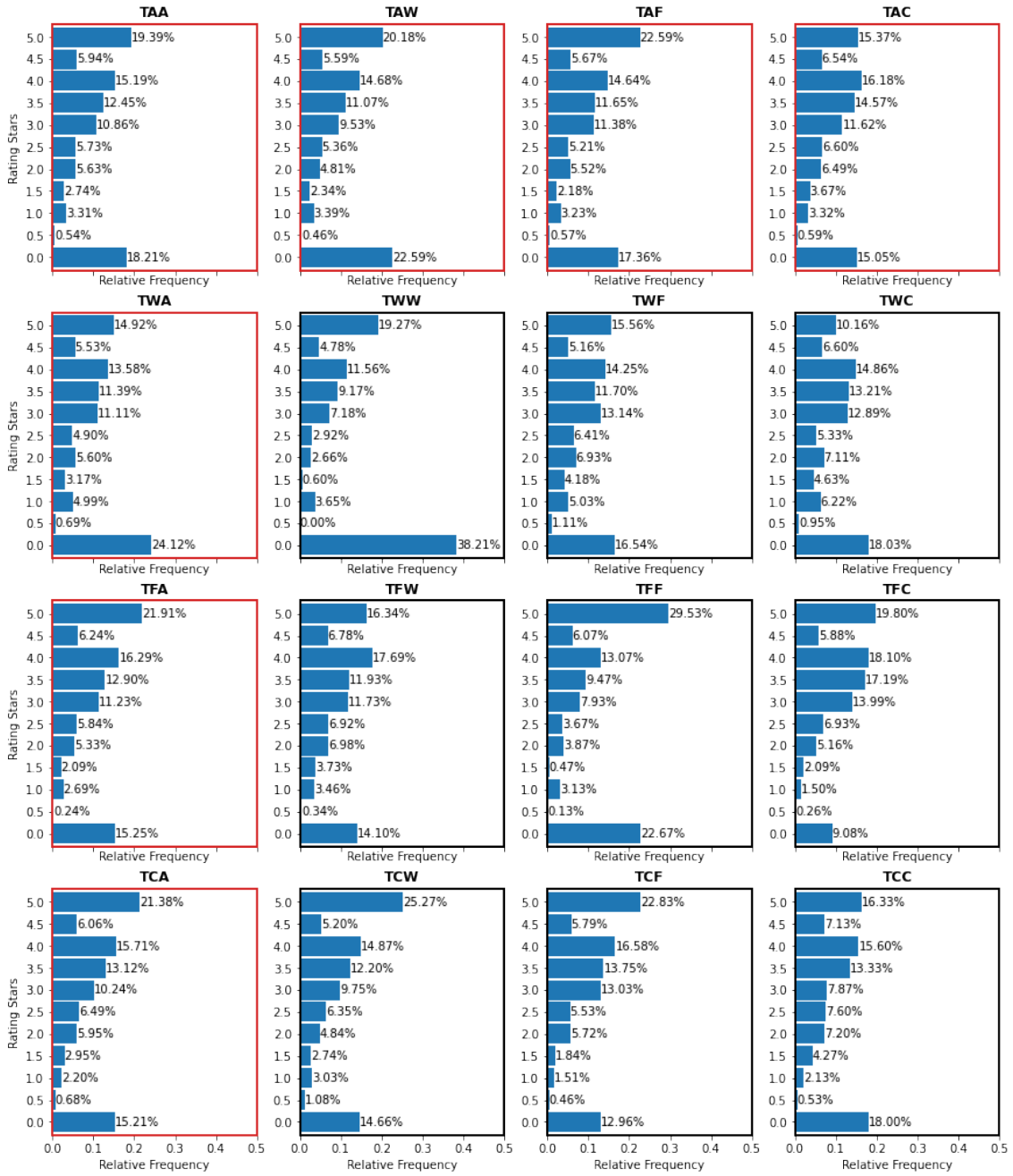


Figure 8.23: Relative frequency of stars rated across all image ranks for samples of different subsets. The sample subsets are indicated by the titles of the subplots and are described in Table 8.5. Plots with a red frame show the average across of the columns besides or rows below, respectively.

8.3.5 Summary

Results Discussion

Influence Of The Training Dataset And Image Dataset In the following, the previously reported results shown are discussed to evaluate how the training data of the employed TERAN models and image pool affect the quality of the retrieved images.

Therefore, different views of the data from the sample subsets (see Table 8.5) shown in the subplots of Figure 8.23, Figure 8.24, and Figure 8.25 are compared to each other.

From all of the mentioned figures, it can be observed that the training data and the image pool indeed affect the quality of the retrieved top-5 ranked images. The most significant

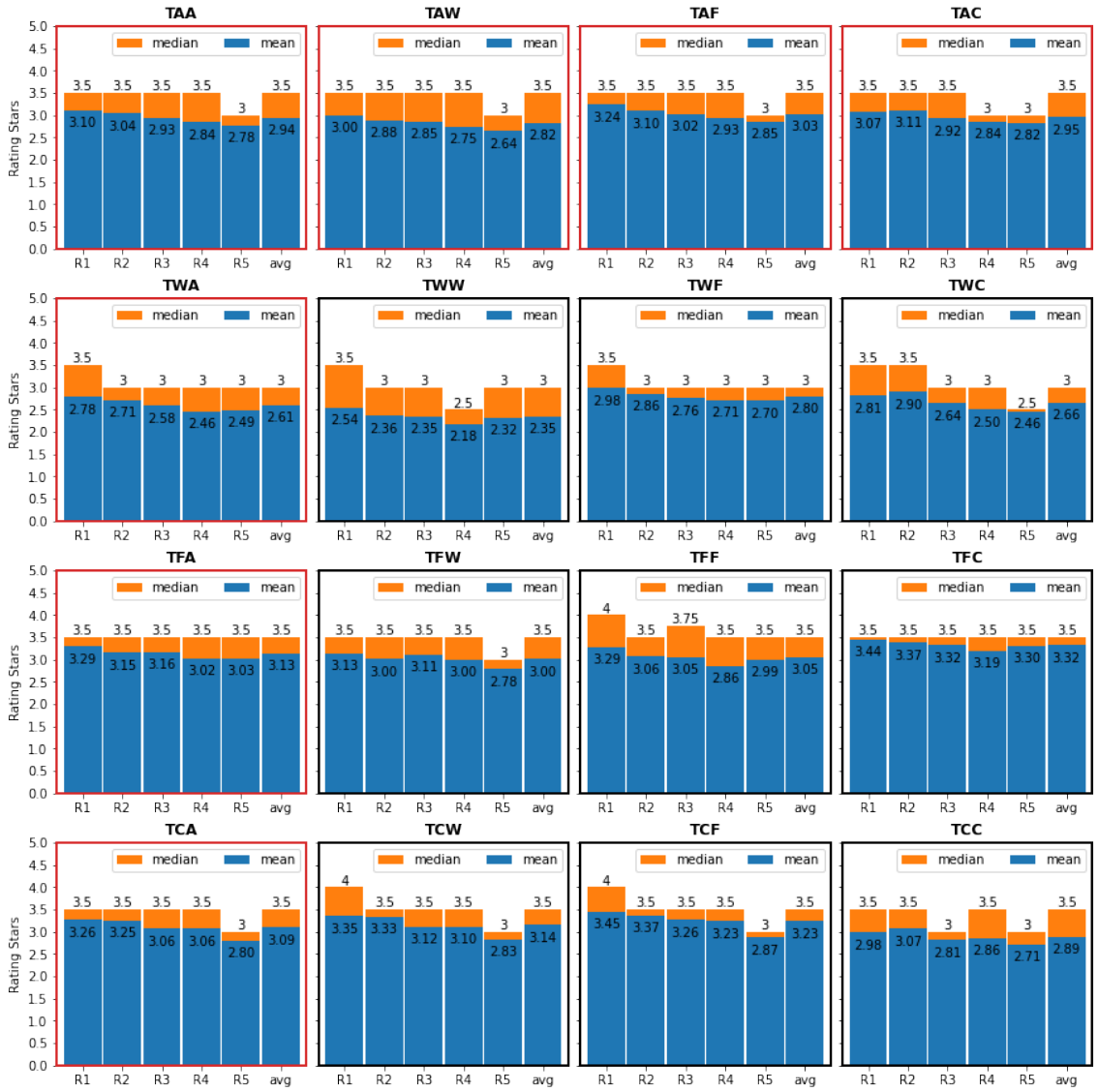


Figure 8.24: Median and mean of stars rated per image rank across all samples of different subsets. The sample subsets are indicated by the titles of the subplots and are described in Table 8.5. Plots with a red frame show the average across of the columns besides or rows below, respectively.

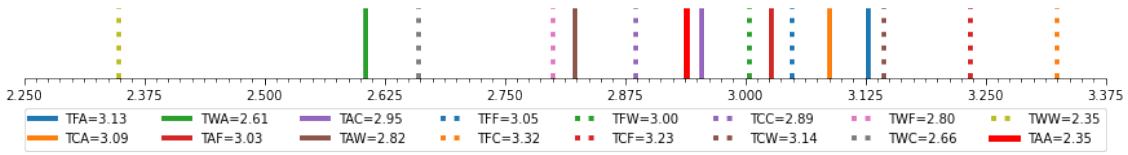


Figure 8.25: Mean values of stars rated across all samples of different subsets and averages of subsets. The sample subsets are indicated in the legend of this figure are described in Table 8.5. For a more detailed view on this data, see Figure 8.24 or Figure 8.23

.

difference in average ratings across all image ranks among the evaluated subsets is between TWW and TFC and with 0.97 almost one full rating star (see Figure 8.23). In other words, the top-5 ranked images of the TERAN model trained on WISMIR v2 and retrieved from the pool of WISMIR v2 images (TWW) are rated at average about one star worse than

	ICC2k	p-value	CI 95%
Rank 1	0.429569	$1.066214e-23$	$[0.36, 0.49]$
Rank 2	0.405765	$1.370899e-20$	$[0.34, 0.47]$
Rank 3	0.425319	$4.180247e-23$	$[0.36, 0.49]$
Rank 4	0.383614	$5.242521e-18$	$[0.31, 0.45]$
Rank 5	0.483691	$2.462311e-32$	$[0.42, 0.54]$
Overall	0.432027	$3.304165e-112$	$[0.4, 0.46]$

Table 8.6: Inter-rater agreement according to ICC2k with p-value and 95% confidence interval reported for the overall average off all 2700 results of the study. The agreement is reported for each of the top-5 ranked images and all for all ranks.

the top-5 images from TFC, which received an average of 3.32 stars. This finding is also supported in Figure 8.23, where it can be noticed that the great majority of 38.21% of the images from TWW received 0 stars. Whereas only 9.08% of the images from TFC received 0 stars, and 43.78% of the images received 4 or more stars.

From Figure 8.24 and Figure 8.25, it can be noticed that the TERAN model trained on WISMIR v2 performed worst with an average of 2.61 and a median of 3.0 stars across all ranks and datasets (TWA). Compared to TERAN trained on Flickr30k, which performed best with an average of 3.13 and a median of 3.5 stars across all ranks and datasets (TFA), this is almost 0.5 stars difference in mean and median. However, the difference between TFA and TCA is marginal, with only 0.04 stars at average across all ranks and datasets. The observation that the TERAN model trained on Flickr30k (TF) outperformed the TERAN model trained on COCO (TC) is surprising because of the difference in the size of the training sets. With approximately 160K text-image pairs in Flickr30k, the dataset is 3.85 times smaller than the COCO with roughly 616K text-image pairs. Moreover, TC outperforms TF by a large margin concerning R@K scores on the COCO evaluation set, whereas TF only slightly outperforms TC on the Flickr30k evaluation (see Section 6.2).

The most suitable images for the language learner dataset come from Flickr30k and receive an average of 3.03 stars across all models (see Figure AVG TAF). Images from COCO (TAC) are similarly relevant, with a slight difference of 0.08 average stars compared to TAF. These observations make sense because the stories from the language learner dataset handle situations of everyday life. Further, as shown earlier in Figure 8.18, the data is more akin to the captions from COCO and F30k. Therefore, it also makes sense that WISMIR v2 images (TAW) represent the language learner dataset worst. However, with an average of 2.82 stars received by the images, the 3.03 gap with Flickr30k images is not too large. When looking at the median number of 3.5 stars received by the images from the different datasets, there is no difference between the datasets.

The discussed findings are also supported by the relative frequencies of stars assigned to the images from the different subsets shown in Figure 8.23. For all subsets, where the TERAN model trained on WISMIR v2 retrieved images, i.e., TWF, TWC, TWW, and TWA, most images received zero stars. For all other subsets – despite TCC – and averages across subsets, the majority of images received 5 stars. Also interesting in Figure 8.23 is that a large proportion of images received between 3.0 and 4.0 stars, which is against common patterns occurring in product ratings.

Inter-Rater Agreement The small p-values and the relatively narrow confidence intervals underline the statistical significance of the ICC2k inter-rater agreement scores reported in Table 8.6. According to Cicchetti 1994, values below 0.4 and values between 0.4 and 0.5 indicate fair agreement among the raters. According to Koo and Li 2016, though, values below 0.5 are interpreted as a poor agreement among the raters. However, both sources investigated the ICC concerning clinical studies, where, e.g., the effectiveness of new drugs or therapies on participants is evaluated, which is essential for approval on the

market. Hence, the inter-rater agreement is of much more importance, so the ICC values are judged more harshly. The ICC2k scores reported in Table 8.6 are therefore considered acceptable but shine a light on the difficulties and limitations of the study, which are discussed in the following section.

Limitations

In the following, issues and limitations about the user study are discussed.

Assignments Per Worker 66.66% of the elected workers from the pilot study participated in the main study. As shown in Figure 8.26, the number of submitted results

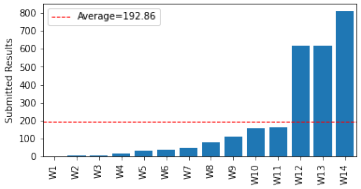


Figure 8.26: The number of workers participated in the study and the number of results they submitted.

per worker varies a lot. This has the effect that the collected data is biased towards the opinions of the three workers who submitted more than 600 results. In total, the three workers submitted 75.67% of the results.

Long Caption Texts The average length of a caption in the LL dataset is about 200 tokens (see Figure 8.18), which is problematic for two reasons: First, the users might not read the complete caption and could not adequately rate the images because they could miss a part of the text visualized in the images. Second, longer texts hold more concepts that could be visualized, which makes it hard to judge how well an image is representing the text. This could also explain the low inter-rater agreement reported in Table 8.6.

Missing Focus Support In our language learner scenario described in Section 1.1, a user's reading comprehension should be supported by providing visual cues for a word or an n-gram (focus term) she selects in a sentence or paragraph (context). However, the data utilized in this study or the study, in general, is missing these focus terms. Hence, the performance of the evaluated TERAN models concerning the suitability for the language learner use case is only partially assessed with this user study.

Missing Correlation Coefficients To precisely evaluate how much the training data of the employed TERAN models affects the quality of the top-5 retrieved images, it would be necessary to compute a correlation coefficients, e.g., the Pearson Correlation Coefficient, between the similarity of the textual training data to the utilized language learner data.

This, however, would require additional experiments with different subsets of the language learner data with varying similarities to the training datasets and is therefore out of the scope of this thesis.

Conclusion

This study showed that the training data of the evaluated TERAN models and the image pools, from which the models retrieved images, affect the quality of the models' top-5 ranked images considering the relatedness to texts from the utilized language learner (LL) dataset. However, due to the lack of a correlation coefficient between the similarities of the training datasets to the LL dataset and the average performance of the respective models, the effect cannot be measured precisely. There are also several other factors that influence the evaluated performance of the models by the users, which makes a quantitative estimation of the effect of the training data hard. Nevertheless, from the study results, it can be observed that the TERAN models trained on COCO or Flickr30k performed better on the task than the TERAN model trained on WISMIR v2, while the performance of the

two former models is only marginally different. This finding supports the expectation that models trained on data more akin to the utilized language learner dataset perform better.

Moreover, it was found that the image pool from which the models retrieve the top-k results influences the quality of the rankings. Although the Flickr30k dataset consists of relatively few unique images (about 30K) compared to COCO (about 123K) or WISMIR v2 (395K), the users rated the top-5 images from Flickr30k with the most stars at average independent of the TERAN models that retrieved the images. However, the mean number of stars is only marginally different, and the median number of stars is the same for all datasets.

With an average of 2.94 and a median of 3.5 stars received by the top-5 images across all employed models and datasets, the models' performances are considered acceptable. That is, the models and datasets are considered suitable for our language learner scenario.

9 MMIRS: Multi-Modal Image-Retrieval System

One goal of this master thesis is to create a system in which state-of-the-art multi-modal transformers can be leveraged for text-image retrieval in a language learner scenario (see Section 1.1). Therefore, MMIRS (**M**ulti-**M**odal Text-**I**mage-**R**etrieval **S**ystem) was developed. As the name of the system suggests, the primary objective of MMIRS is to retrieve the best matching images according to a user-provided textual query. To provide simple access to MMIRS text-image retrieval functionality, a browser plugin that serves as the user-interface was build (see Section 9.8). This browser plugin implements the practical use case of our language learner scenario as described in Section 1.1.1.

The uniqueness of MMIRS, as opposed to popular text-image retrieval systems such as Google Image Search[1], is that the textual query is not only a sentence or a sequence of words but a pair consisting of a context sequence and a focus sequence. This feature requirement comes from the language learner scenario, where a user can select a word or n-gram in an arbitrary text, e.g., from Wikipedia[2]. The n-gram the user selects is the focus sequence of the query, and the surrounding sentence or paragraph of the focus is the context of the query. MMIRS's main objective is to find contextualized images that match the focus and its context so that the images can serve as visual cues to support the users reading comprehension (see Section 9.3).

To reach this goal, two primary obstacles need to be overcome. One of these challenges is the performance and functionality of current multi-modal text-image retrieval models, lies outside MMIRS' direct responsibility. Current models are trained and evaluated on the retrieval of images for relatively short and simple queries, and above all without incorporating a focus term (see Chapter 3). The second obstacle MMIRS is designed to solve is the language learner scenario's "real-time" constraint, which is elucidated in the following section.

9.1 Problem Statement

In order to conceive the problem that MMIRS tries to solve, first, the general task of text-image retrieval is explained in the following.

Formally, in MMIRS, we define a query as

$$Q = (c, f) \tag{9.1}$$

where c is the context sentence and $f \subset c$ is the focus n-gram in the context. From a very abstract perspective, MMIRS implements a function $MMIRS : Q \to I$ that maps Q to a totally ordered set of images $I \subseteq P$ from a (large) pool of images P. The set of images I is sorted in descending order according to a similarity or distance function $\Phi(Q, I_k)$ that

1. https://images.google.com/
2. https://wikipedia.org

measures how similar an image I_k is to a query Q, so that

$$I = \{I_i \mid i \in [0, \ldots, |P|] \wedge \Phi(Q, I_i) \geq \Phi(Q, I_{i+1})\} \tag{9.2}$$

$$I_0 = \underset{I_k \in P}{\operatorname{argmax}}\, \Phi(Q, I_k) \tag{9.3}$$

$$I_n = \underset{I_k \in P}{\operatorname{argmin}}\, \Phi(Q, I_k) \tag{9.4}$$

In state-of-the-art multi-modal text-image retrieval models, Φ is a complex neural network (see Chapter 3), with high computational costs to compute a single similarity score for a text-image pair – even at inference time on a modern GPU-powered system. Hence, finding the best matching image from a large pool of images becomes infeasible for "real-time" applications on systems with limited hardware infrastructure.

9.2 Solution Approach

MMIRS is specially designed to overcome the obstacle defined in Section 9.1 by employing a two-stage retrieval process. In the first stage, the *Preselection Stage* (PSS) (see Section 9.5), the image pool P gets drastically reduced to a subset, referred to as *Image Search Space* (ISS), so that $|P| \gg |ISS|$. The *ISS* contains image candidates that either match the context c, the focus f, or both to some degree and is computed by efficient uni-modal retrieval strategies. In the second stage, the *Fine Selection Stage* (FSS) (see Section 9.6), a three-step process involving a state-of-the-art multi-modal text-image retrieval model, sorts the images of the *ISS* according to the context and the focus of a query to obtain the ordered set of images I (see Equation 9.2).

A schematic overview of this process is shown in Figure 9.1.

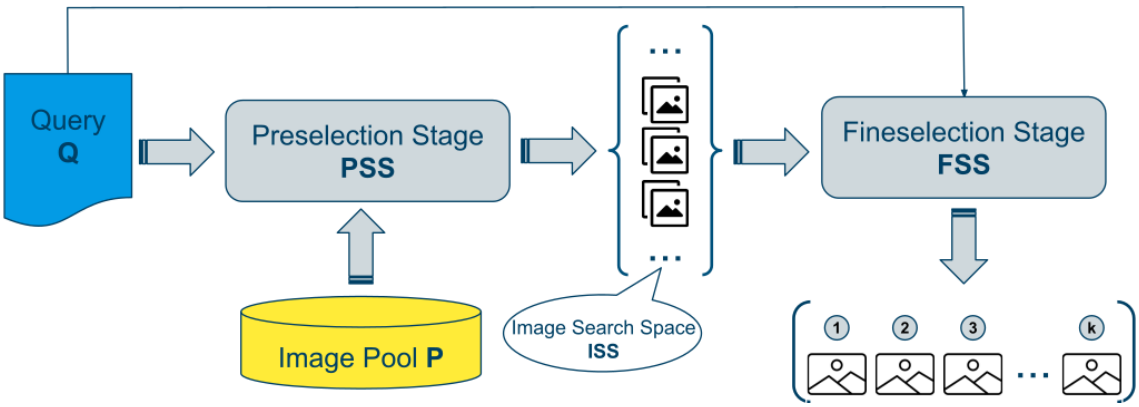


Figure 9.1: Schematic overview of the two-staged retrieval process of MMIRS. Note that the image pool P is much larger than the *Image Search Space* (ISS), i.e, $|P| \gg |ISS|$.

9.3 Features

In this section, the main features of MMIRS are summarized. For details, how these features are implemented, see Section 9.4.

Focus Term Support

The input to traditional text-image retrieval methods or state-of-the-art models used for this task is a sequence of words, e.g., a complete sentence often referred to as the query. As mentioned in the introduction of this chapter, one unique feature of the system is the

support of a focus n-gram as an extension to the query. In MMIRS, a query is a pair consisting of a context and a focus (see Equation 9.1). The system retrieves the best matching images according to the context, but paying particular attention to the focus. To exemplify this, think of a query consisting of the context "*Stingrays exhibit a wide range of colors and patterns on their dorsal surface to help them camouflage with the sandy bottom.*" with a focus on the word "*Stingrays*". MMIRS' goal is to find images that do not only match the context sentence but also, and essentially, the focus term. A visual example is illustrated in Figure 9.2. In this illustration, the context of the query is shown on the

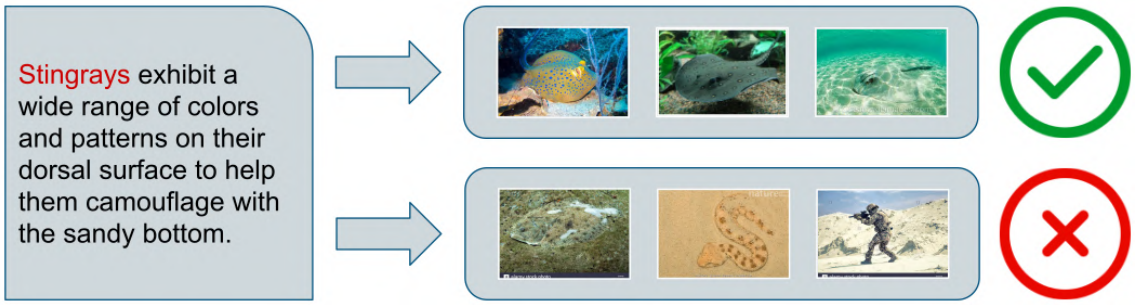


Figure 9.2: An illustration of the idea behind the support for focus terms in the query of MMIRS.

left with the focus term "Stringrays" highlighted in red. On the right, two example results containing three images are presented. The upper result is correct since it shows images that match both the context and the focus term by displaying images of stingrays that camouflage with the bottom of the sea. The bottom result is inaccurate because it contains images that only match the context, i.e., a flounder, a snake, and a soldier camouflaging with the background, but not the focus term "Stingrays".

Word-Region-Alignment Matrix Generation

MMIRS can visualize the word-region-alignment matrices generated by the text-image retrieval models and dynamically provide the generated figures via URL. The cells of a word-region-alignment matrix express the similarity of textual tokens of the query to the regions in the images, typically measured with cosine-similarity.

This feature is especially useful from a scientific perspective since it provides insights into the models' reasoning on the image retrieval process. For more information on this component, see Section 9.7.2

Focus Region Highlighting

Based on the information in the word-region-alignment matrices generated by the multi-modal models, MMIRS can highlight the region in an image that is most similar to the focus of the query. With this feature, MMIRS can provide specific visual cues to support the language learners' reading comprehension. For more information on this component, see Section 9.7.3

Different Retrieval Models and Datasets

There are several factors with varying degrees of influence on the resulting set of retrieved images by MMIRS. One might think that the text-image retrieval model has the most impact on finding the best matching images from a pool of images according to a user's query. Although the model plays a very prominent role in the system, the pool or dataset of images is at least as crucial for the system's response. MMIRS could theoretically employ the perfect text-image retrieval model but will not find any query-relevant images if the

pool does not contain these. Since the perfect text-image model does not exist, this logic also applies vice versa. If a model trained on a specific dataset is used to retrieve the best matching images from a pool with very different, it will not perform well. Therefore, in MMIRS, different retrieval models and datasets are available. The currently supported models and datasets are listed in Table 9.1 and Table 9.2, respectively. In the current version, a user chooses the model and datasets but in future work, it is planned that MMIRS automatically determines the best retrieval model and dataset from the users' queries.

Name	**Type**	**Training Datasets**
$TERAN_{W2}$	TERAN	WISMIR V2
$TERAN_F$	TERAN	Flickr30k
$TERAN_C$	TERAN	COCO

Table 9.1: Text-Image retrieval models available in the current MMIRS version. More information about the models and datasets can be found in Chapter 3 and Chapter 4, respectively.

Name	**Number of images**
WISMIR V2	395872
Flickr30k	31784
COCO	123287

Table 9.2: Available image datasets in MMIRS and their number of images. More details on the respective datasets can be found in Chapter 4.

System Configuration

In order to keep the system as flexible as possible, the system administrator can define most of the system's behavior in a configuration file. Also, the models and datasets are registered via the configuration file, so that adding new models or datasets requires no additional code (if the resources are of the default supported types).

9.4 System Overview

As mentioned in the introduction of this chapter, the multi-modal text-image retrieval system, MMIRS, is designed to function as a "real-time" application. The system's back-end is entirely written in Python and builds on multiple popular frameworks, which are mentioned in the following corresponding sections.

To provide access to the system's functionality to other applications, REST API endpoints (see Section 9.7.4) are made available. In the current version of MMIRS, a browser plugin for Google Chrome[3] that consumes the API serves as the user interface (see Figure 9.3).

The retrieved images are provided via URL to a static image server (see Section 9.7.1), which is also part of MMIRS.

9.4.1 Backend System

Since existing text-image retrieval models have a high computational cost to retrieve the best matching images according to a query, MMIRS employs a two-stage retrieval process.

3. https://www.google.com/chrome/

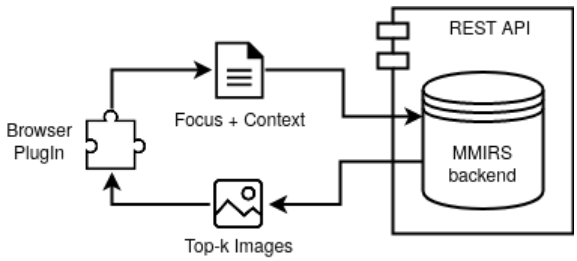


Figure 9.3: A simplified schematic overview of the browser plugin frontend that communicates with the backend of MMIRS via the REST API. Further the principal data flow is indicated with arrows between the components and artifacts.

A brief overview of the backend system and the data flow between its components is provided in the following. Architectural and algorithmic details on the single components and their constituents are described in later sections.

As distinct from Figure 9.1, in Figure 9.4, a schematic overview of MMIRS with focus on the *PSS* and *FSS* stages is illustrated.

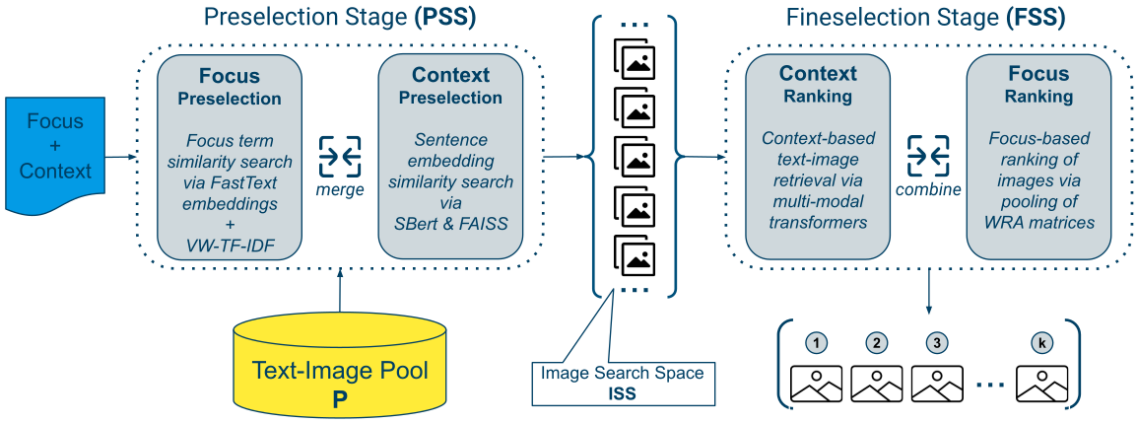


Figure 9.4: More detailed schematic overview of the two-staged retrieval process of MMIRS. Note that the image pool P is much larger than the *Image Search Space (ISS)*, i.e, $|P| \gg |ISS|$.

The *Preselection Stage (PSS)* is responsible for significantly reducing the number of images for the multi-modal transformer model applied in the second stage. The input to the *PSS* is the user's query consisting of the context and the focus term. In addition to the query, the dataset, referred to as *Text-Image Pool (P)* which the *PSS* filters, needs to be specified by the user. The word "text" in *Text-Image Pool* might be confusing but is actually accurate since the *PSS* also operates on the captions and the images of the datasets with uni-modal and multi-modal methods.

The *Preselection Stage* consists of two submodules. One is responsible for finding images related to the focus term (see Section 9.5.1), and the other is responsible for finding images related to the context (see Section 9.5.2). The output of the PSS is an unordered set of images called *Image Search Space (ISS)*, which is the result of merging the set of focus-related images and context-related images. The merging operation applied on the two sets can either be intersection or union.

The second stage, or *Fine Selection Stage (FSS)*, is responsible for retrieving the best matching images from the *ISS* according to the user query. This is accomplished by forwarding every image in the *ISS* through the text-image retrieval model selected by the user. The result of this process is a sorted list of the images in the *ISS* referred to as I in Equation 9.2.

9.5 Preselection Stage

To create the *ISS*, only the most relevant images according to the focus and the context of a user's query are selected from P. The *Focus Preselector (FPS)* described in Section 9.5.1 component finds focus-relevant images, and the *Context Preselector (CPS)* described in Section 9.5.2 finds context-relevant images. The resulting sets of images from the *FPS* and *CPS* components get merged to obtain the final *ISS*. The two sets get merged by intersection by default, but the union is used as a fallback if too few images are in the intersection set.

9.5.1 Focus-based Preselection

A schematic overview of the data flow and the process to create this subset of images relevant to the focus term of a user's query is shown in Figure 9.5.

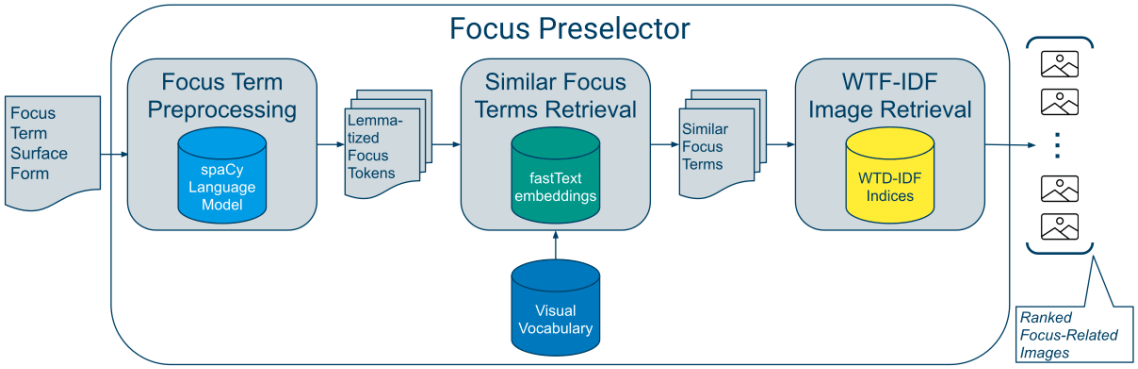


Figure 9.5: Schematic overview of the data flow and process of the *Focus Preselector* component. This component is part of MMIRS Preselection Stage and is responsible for finding a small subset of images relevant to the focus term of a user's query from a large pool of images.

To better understand the single steps, an example focus term is defined: "the small birds". The input to the component is the surface form of the focus term. In the first step, the focus term gets preprocessed. By default, this preprocessing consists of tokenization, stopword filtering, POS tag filtering, and finally, lemmatization. For this, the spaCy (Honnibal et al. 2020) framework is used with the language model "en_core_web_lg". After preprocessing the example focus, the output would be the list of lemmata "small" and "bird". To understand the following steps, the concept of "visually-weighted tf-idf" (VW-TF-IDF), which was developed within this master thesis, needs to be introduced.

Visually-Weighted TF-IDF (VW-TF-IDF)

The core of this idea is the well-known "tf-idf" score, a term weighting scheme that measures the importance of a term in a corpus of documents. Traditionally these documents are sentences, paragraphs, articles, or any other text. In this novel approach, referred to as VW-TF-IDF, this concept is applied to images, i.e., the images are interpreted as documents. The content of the image documents are the detected object labels, or, in other words, the output of an object detection and classification network. For this, a pre-trained Faster-R-CNN with ResNet-101 model (Anderson et al. 2018; Z. Yu et al. 2020) is utilized. The convolutional neural network is effective and very efficient in detecting prominent regions in an image in a bottom-up fashion. Additionally, the network classifies the detected regions by assigning object and attribute labels from a fixed-sized vocabulary, which is referred to as "visual vocabulary" in the following. The visual vocabulary used in this work

contains 1200 object labels and 400 attribute labels, which slightly overlap, resulting in about 1400 unique terms.

As an example, two different images where regions and their labels generated by the Faster-R-CNN are highlighted are depicted in Figure 9.6 and Figure 9.7.

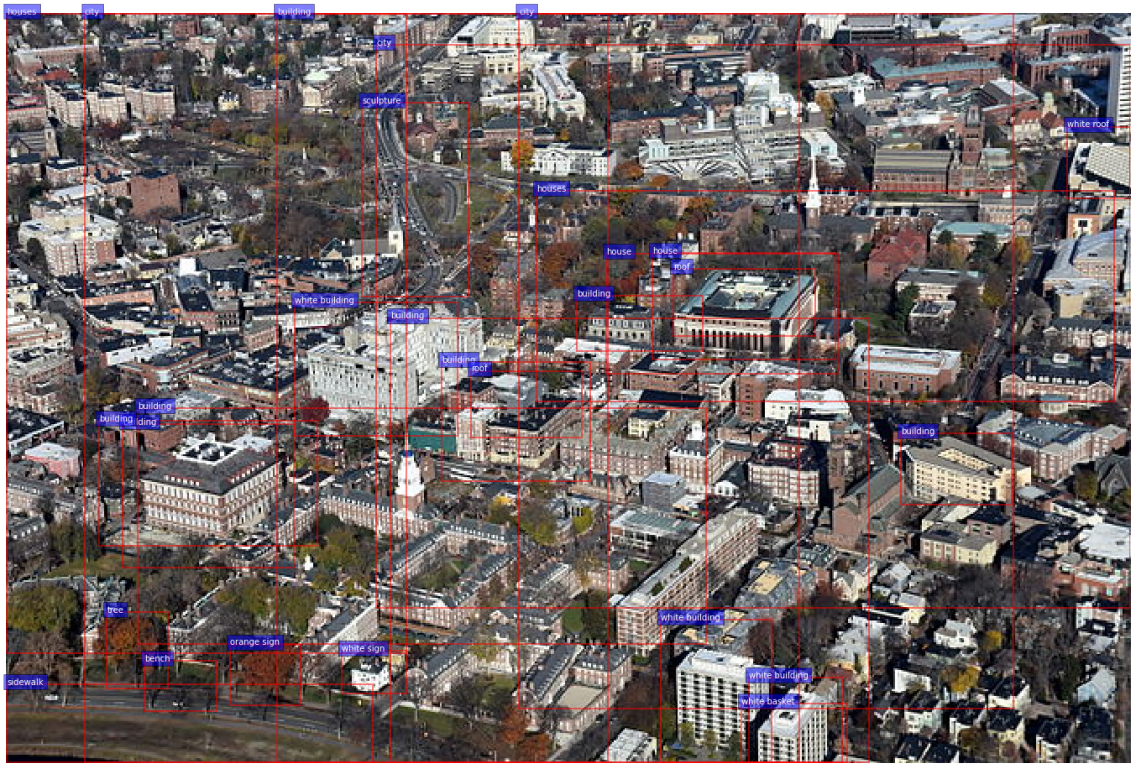


Figure 9.6: Detected and classified regions in an image showing buildings of a city from above, generated by the object detection and classification model Faster-R-CNN. The detected region of interests are surrounded by red borders with their respective classification labels shown in blue boxes in the top-left corner.

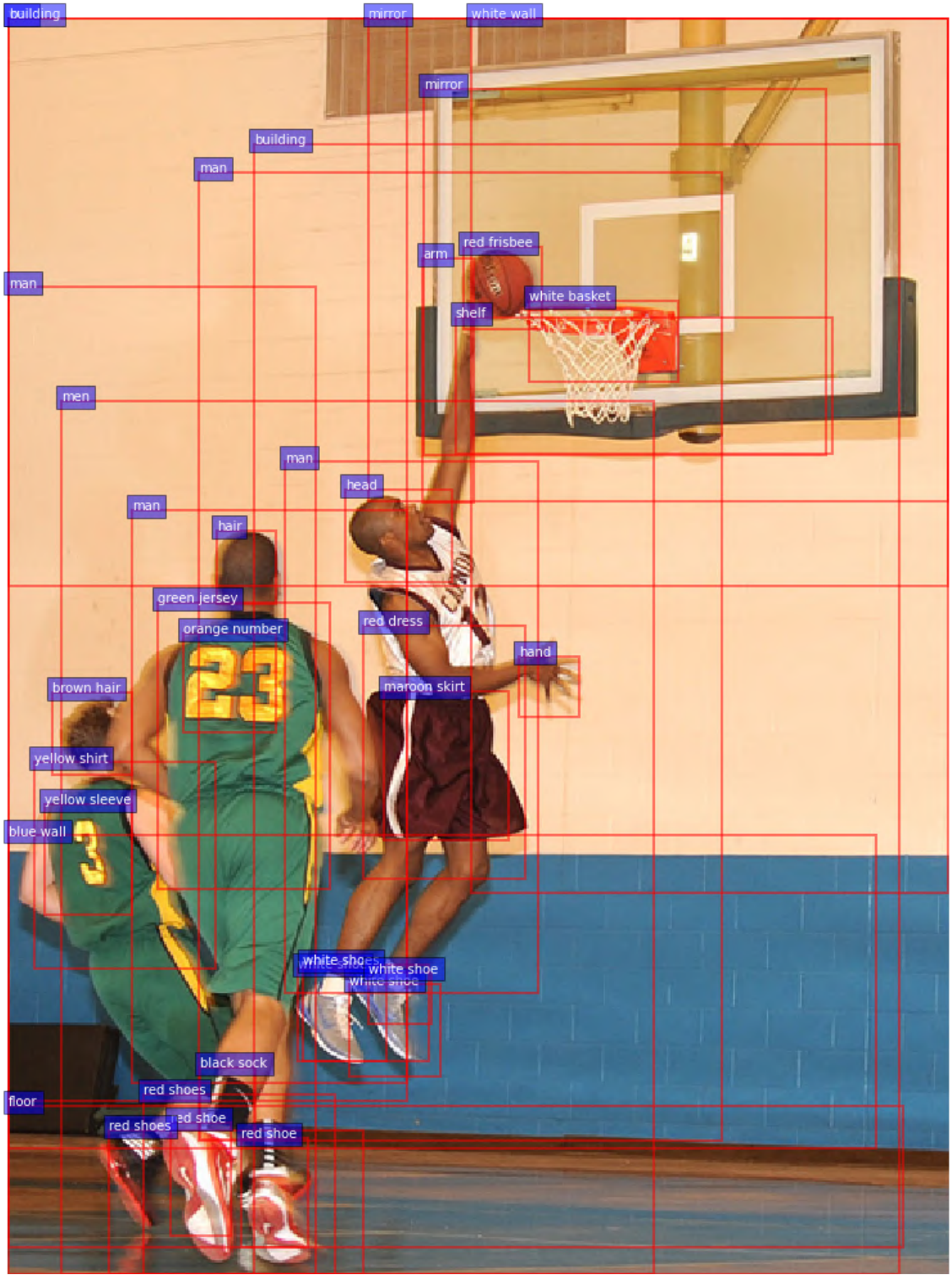


Figure 9.7: Detected and classified regions in an image showing basketball players, generated by the object detection and classification model Faster-R-CNN. The detected region of interests are surrounded by red borders with their respective classification labels shown in blue boxes in the top-left corner.

To compute the VW-TF-IDF score of a term and a document, i.e., of an object or attribute label and an image, the classical formula of tf-idf is extended by an additional weighting scheme. The motivation for this is that the score should be higher if the region with the respective label is prominent in the image, and the classification network was confident. Hence, the confidence scores of the classifications and the areas of the detected

regions are incorporated in addition to the counts of the terms from the traditional tf-idf formula.

Formally, the VW-TF-IDF of a term t and an image document d is defined as

$$\text{vw\_tf\_idf}(t, d) = \text{vw\_tf}(t, d) \cdot \log\left(\frac{\text{num}_{\text{docs}}}{\text{df}(t) + 1}\right) \tag{9.5}$$

where the logarithmic term is standard inverse document frequency (IDF) with simple additive Laplace-Smoothing for numerical stability.

The visually weighted term frequency (VW-TF) is defined as

$$\text{vw\_tf}(t, d) = \frac{\text{cnt}(t, d) \cdot \text{weight(t, d)}}{\text{num\_terms}(d)} \tag{9.6}$$

where $\text{cnt}(t, d)$ is the number of times term t appears in document d and $\text{num\_terms}(d)$ is the total number of terms in the document.

The weight of the term t in d is defined as

$$\text{weight(t, d)} = \alpha\ \text{conf}(t, d) + (1 - \alpha)\ \text{area}(t, d) \tag{9.7}$$

$$\text{conf}(t, d) = \frac{1}{\text{cnt}(t, d)} \sum_{t^{(i)} \in t} t^{(i)}_{conf} \tag{9.8}$$

$$\text{area}(t, d) = \frac{1}{d_{area}} \sum_{t^{(i)} \in t} t^{(i)}_{area} \tag{9.9}$$

where $t^{(i)}_{conf}$ is the confidence score and $t^{(i)}_{area}$ is the area of the region of term $t_{(i)} \in d$, and d_{area} is the area of the image document d. The parameter α is used to control the importance of the confidence or area of a term in the final weight of t.

Note that a term t with label l can occur multiple times in different regions with different confidence scores in a document d. Hence, a term t is the set of all terms with label l and is also identified by l.

There are two additional hyper-parameters, which are not reflected in the preceding formulas but are essential for computing meaningful scores. Both are threshold parameters that decide if a detected object or attribute in the image should be accounted for in the score or not. Objects or attributes classified with a confidence score below the respective threshold are ignored when computing the score defined in Equation 9.8 and Equation 9.9.

Now that VW-TF-IDF was introduced, the successive steps of the *Focus Preselector* component follow.

Since there are no constraints on the text, a query, and therefore also the focus, the set of possible focus terms is infinite. The set of different labels, i.e., the size of the visual vocabulary, is finite, with about 1400 distinctive terms. Hence, the probability for a focus terms to be out-of-visual-vocabulary is high. To overcome this issue, the top-k similar in-visual-vocabulary (IVV) tokens for each out-of-visual-vocabulary (OOVV) token in the output of the preprocessing step are computed. Cosine similarities between the OOVV token embeddings and all IVV token embeddings are computed to find these similar tokens. Since this would usually require a lot of computational power, the high-performance universal vector embedding library Magnitude (Patel et al. 2018) is utilized. Magnitude enables extremely fast and efficient similarity computation with dense vectors by using a sophisticated combination between the data format of precomputed embeddings and caching strategy. The framework supports several popular vector embeddings out-of-the-box, from which fastText (Bojanowski et al. 2017) embeddings were chosen. The reason for this decision is that fastText embeddings are subword embeddings, i.e., word or token embeddings are represented by the sum of multiple character n-gram embeddings. This

strategy makes fastText embeddings very stable against out-of-vocabulary issues since rare words, that the language model did not see during training are broken up into common character n-grams.

To retrieve the top-k related images for a given focus term, the VW-TF-IDF scores are computed and summed for each similar focus term and each image in the image pool. The resulting list of scores is sorted in descending order so that the top-k focus-relevant image documents are the first k elements of the list.

In order to meet the "real-time" requirements of the application, a precomputation and indexing step is mandatory. Before MMIRS runtime, the VW-TF-IDF scores for each term in each image document of a pool of images are precomputed and persisted in an efficient and accessible data format. For this, pandas[4] DataFrames are used since they support high-performance (multi-level-)indexing and aggregation functionality as well as efficient persistence formats.

9.5.2 Context-based Preselection

The input to this component is the context of a query, and a ranked list of context-related images is the output. A schematic overview of the component is depicted in Figure 9.8.

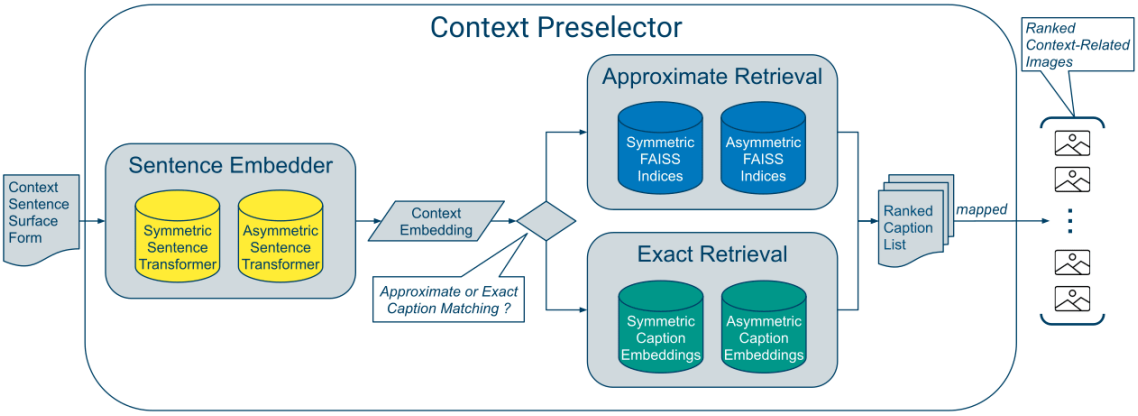


Figure 9.8: Schematic overview of the data flow and process of the *Context Preselector* component. This component is part of MMIRS Preselection Stage and is responsible for finding a small subset of images relevant to the context part of a user's query from a large pool of images.

Briefly summarized, the ranked list of context-related images is found by computing the best matching captions for the query via cosine similarity from a large pool of text-image pairs. In the first step, a sentence embedding of the context's surface form gets computed. Before continuing with the next steps, the concept of sentence embeddings is explained briefly in the following subsection.

Sentence Embeddings

In MMIRS, sentence embeddings are computed with pretrained SentenceTransformer models from the SentenceTransformers (Reimers and Gurevych 2019) framework, which are specially designed for computational efficiency and information retrieval tasks. These kinds of models compute fixed-sized embeddings for sentences with various lengths, which are suitable for tasks like semantic textual similarity (STS), automatic text summarization (ATS), or question-answering (QA). Several strategies and architectures exist to compute these fixed-sized sentence embeddings.

4. https://pandas.pydata.org/

The pretrained models of the SentenceTransformers framework use a similar approach like TERAN (see Section 3.3) and employ two separated stacks of traditional textual transformers like BERT (Devlin et al. 2019) or RoBERTa (Liu et al. 2019). Two input sentences are forwarded through the separate transformers to calculate the corresponding token embeddings. After that, a pooling strategy on the output tokens is applied to project the output embeddings in the same n-dimensional vector space. SentenceTransformer models use mean-pooling per default, where the average of the output token embeddings is derived. Depending on the task, different objective functions are used to train the model in an end-to-end fashion. To compute a single sentence embedding from the context of a user's query, the context gets forwarded through one of the two stacks of the SentenceTransformer model. The framework has published multiple models, which perform differently depending on the specific task. What has a significant impact on the models' performance, is the lengths of the two sentences used during training. For example, in the question-answering task, the two input sentences consist of a question and a paragraph containing the answer. Or, in automatic text summarization, the input consists of a summary sentence and the text to be summarized. In both example tasks, the similarity between the two inputs needs to be computed and compared, but the two sentence lengths usually are very different. This scenario is called asymmetric, and the scenario where the two input sentences have about the same length, e.g., in STS, is called symmetric. The *Context Preselector* component finds the most similar captions of the context of a user's query. Because MMIRS is designed to support multiple datasets like Flickr30k or WISMIR (see Chapter 4), where the average caption length is different, and the query context length can also be very different, one symmetric and one asymmetric model is employed. In the current version of MMIRS, the symmetric embeddings are chosen per default. In later versions, MMIRS should decide this dynamically, depending on the dataset chosen by the user.

Symmetric and asymmetric embeddings of all supported datasets are precomputed and persisted before MMIRS runtime. When MMIRS is booting, all embeddings are loaded into memory to meet the "real-time" requirements.

Now that the concept of sentence embeddings and models to produce these embeddings has been explained, the steps of the *Context Preselector* component are described in the following. If the input context length is different from the average caption length of the chosen dataset (see Section 9.4.1), asymmetric models and asymmetric embeddings are chosen, and symmetric models and symmetric embeddings otherwise. Per default, "paraphrase-distilroberta-base-v1"[5] as the symmetric and "msmarco-distilbert-base-v2"[5] as the asymmetric model are chosen. In the following (and also Figure 9.8), this decision is not elucidated or highlighted anymore but thought to be implicit.

The first step of the *Context Preselector* component is to compute the embedding of the context of a user's query. After that, the resulting context embedding is compared against the embeddings of the captions of the chosen dataset to find the most similar captions. Because MMIRS is designed to support different datasets with a different number of elements (see Section 9.3), retrieving the top similar captions can be computationally intensive. Therefore, in the system two strategies, namely, *exact retrieval* or *approximate retrieval*, are implemented to compute the best matching captions. In the current version of MMIRS, *approximate retrieval* is the default, but a user can also select the strategy she wants to use for a request.

With the *exact retrieval* strategy, the best matching captions are found by computing the cosine similarity between the context embedding and all embeddings of the captions of the dataset. This is done with optimized methods from the SentenceTransformers framework, but finding the best matching captions can take long for large datasets even with these methods.

5. https://www.sbert.net/docs/pretrained_models.html

For the *approximate retrieval* strategy, FAISS (Johnson et al. 2019), a popular library for high-performance similarity search and dense vector clustering is utilized. Despite several sophisticated methods to perform approximate similarity search on clusters of embedding vectors, an Approximate Nearest Neighbor (ANN) strategy on Voronoi Cell (VC) clusters of all sentence embeddings was chosen. Instead of naively searching through the clusters, an inverted file index is created by FAISS to drastically increase the similarity search efficiency[6]. FAISS also provides convenient methods to create and persist these indices for the sentence embeddings created with the SentenceTransformer models, which is done for every supported dataset before MMIRS runtime.

The approximate approach with FAISS is much faster than the exact retrieval strategy with SentenceTransformers at the cost of an eventual accuracy loss. That is, the list of retrieved context-relevant captions can be (slightly) different. Since the *Context Preselector* is only a preprocessing step in the overall retrieval process of MMIRS, this trade-off between accuracy and computational efficiency is deliberately accepted.

In the final step of the component, the ranked list of context-relevant captions gets mapped back to their corresponding images to obtain the ranked list of context-related images, i.e., the output of the *Context Preselector*.

9.6 Fine Selection Stage

In the *Fineselection Stage* (FSS), the primary multi-modal image retrieval happens. To the best of my knowledge, no current model or system used for text-image retrieval supports queries consisting of a context and a focus term. To support such two-fold queries in MMIRS anyway, the FSS retrieves the top-k images for a given query in three steps, described in the following.

9.6.1 Step 1 – Context-based Ranking

In the first step, the preselected set of images from the *Preselection Stage* together with the context of the query are forwarded through a state-of-the-art multi-modal transformer model to obtain context-based text-image similarity scores. The TERAN models employed in MMIRS compute this global text-image similarity score by aggregating a generated word-region-alignment matrix, in which cells are representing the similarity of a textual token to a visual region (see Section 3.3.1). The global similarities of each image to the context are computed, so that every image $i \in$ ISS has an associated "context-score", referred to as $s_i^{(c)}$. Further, the models were modified so that the raw WRA matrices of the images and the captions are returned in addition to the global similarity scores. For the following steps, the information contained in the WRA matrices, i.e., the fine-grained word-region similarities and the position of the focus in the context, is necessary.

9.6.2 Step 2 – Focus-based Ranking

In the second step, a "focus-score", $s_i^{(f)}$ is computed for every image $i \in$ ISS by pooling the WRA matrices of the images and the context generated by the multi-modal model in the first step. There are two pooling strategies, defined by Equation 9.10 and Equation 9.11, to determine the "focus-score".

$$\text{score}(f; \mathbf{A}) = \frac{1}{N * (f_e - f_s + 1)} \sum_{i=0}^{N} \sum_{j=f_s}^{f_e} \mathbf{A}_{ij} \tag{9.10}$$

6. https://github.com/facebookresearch/faiss/wiki/Faiss-indexes

$$\text{score}(f; \mathbf{A}) = \max_{0 \leq i < N} \frac{1}{f_e - f_s + 1} \sum_{j=f_s}^{f_e} \mathbf{A}_{ij} \tag{9.11}$$

where N is the number of regions per image (following previous work of Anderson et al. 2018; Nicola et al. 2020, this is fixed to 36); f_s and f_e are the starting and ending indices of the focus in the context, respectively; and $\mathbf{A} \in \mathbb{R}^N \times \mathbb{R}^{|c|}$ is the word-region-alignment matrix of an image $i \in$ ISS and the context c of the query (see Section 3.3.1). Equation 9.10 and Equation 9.11 are referred to as "average-pooling" and "max-average-pooling", respectively.

9.6.3 Step 3 – Re-ranking

In the third and final step, first, to combine the "context-score", $s_i^{(c)}$, and the "focus-score", $s_i^{(f)}$, the scores need to be normalized so that they are in the same range and on the same scale. To scale the scores, sklearn's MinMaxScaler[7] feature transformation is utilized, which distributes the scores so that the minimum score is mapped to 0 and the maximum score is mapped to 1. After scaling, the "context-scores" and the "focus-scores" of every image $i \in$ ISS are combined with a simple weighted average as defined in Equation 9.12.

$$s_i^{(q)} = \alpha \cdot s_i^{(c)} + (1 - \alpha) \cdot s_i^{(f)} \tag{9.12}$$

where $\alpha \in [0, 1]$ is the weight for the weighted average; $s_i^{(c)}$ and $s_i^{(f)}$ is the scaled "context-score" and the "focus-score", respectively; and $s_i^{(q)}$ is the final score of image i with respect to the query q. To obtain the final ranks, the ISS is sorted in descending order according to the final scores of the images. The top-k matching images according to a user's query q are the first k elements of the sorted ISS.

9.7 MMIRS – Auxiliary Components

In this section auxiliary components, that are not involved with the text-image retrieval process itself but are required to leverage MMIRS in an actual application like the browser plugin (see Section 9.8) are described.

9.7.1 Static Image Server

This component is responsible for serving retrieved and generated images via URL. Therefore, Python's built-in "http.server" module[8] is utilized.

Serving images from multiple datasets and dynamically generated images like WRA matrix plots (see Section 9.7.2) or images, with the maximum focus region highlighted (see Section 9.7.3), requires a different strategy than the default, i.e., statically storing all image files in or under the same root directory. Thus, the image server component dynamically creates hard links of the respective image files in the HTTP server's root directory when they need to be served to a user via the API. This has the advantage that the images of the available datasets, as well as the dynamically generated images, can be stored in different directories and that no additional space – besides the size of the Inodes – is occupied on the disk as opposed to copying the files. Further, creating hard links is faster than copying files, and the root directory of the image server can be cleared in regular intervals to prevent a cluttered directory. Note that symbolic links cannot be used since HTTP servers, in general, cannot traverse outside the specified root directory due to security reasons.

7. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
8. https://docs.python.org/3.7/library/http.server.html

9.7.2 WRA Matrix Plotter

The objective of this component is to dynamically generate plots of word-region-alignment matrices for an image and the context of a query. The necessary information, i.e., the WRA matrices themselves, the mapping from the textual token embeddings to their readable form, and the span of the focus in the context, is generated and collected from a multi-modal model employed in the Fineselection Stage. The Python library matplotlib[9] is utilized to generate a WRA matrix plot and persist the resulting figure on disk. After that, the component registers it at the static image server component to make it available via URL.

To meet the "real-time" constraints of MMIRS, the component generates and registers requested WRA plots for multiple images in parallel.

An example of a WRA plot is shown in Figure 9.9.

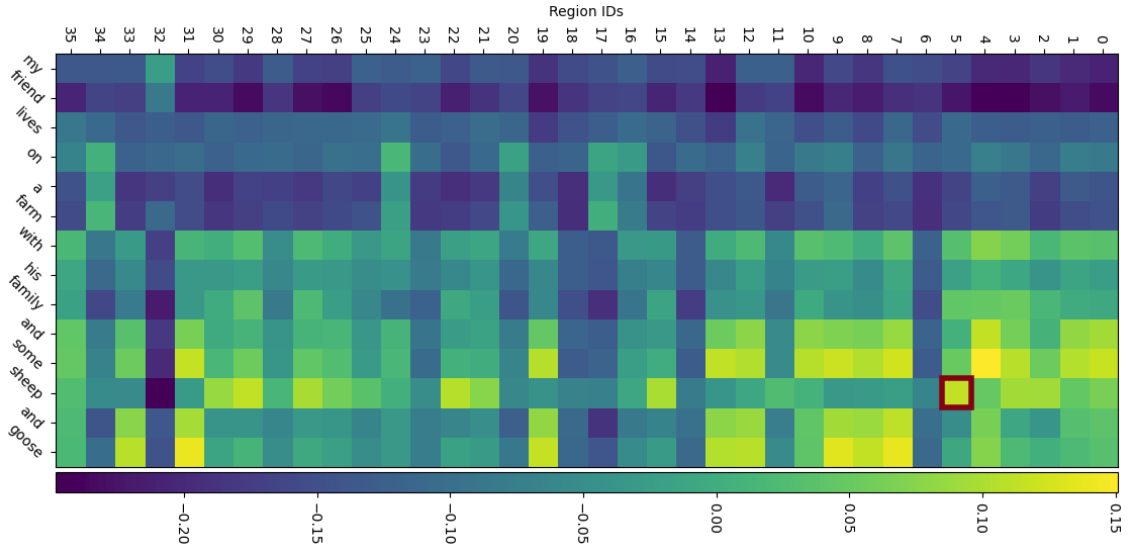


Figure 9.9: An example word-region-alignment (WRA) plot generated by the WRA Matrix Plotter auxiliary component of MMIRS. The highlighted cell indicates the region with the highest signal to the focus, which was *"sheep"* in this example query. The context of the query was *"My friend lives on a farm with his family and some sheep and goose"*. Note that the figure is rotated by 90 degrees.

9.7.3 Focus Region Highlighter

As the name suggests, this component highlights the region with the highest similarity to the focus in an image. Similar to the WRA Matrix Plotter component, the WRA matrix of an image and a context, the span of the focus in the context, and the mapping from the textual token embeddings to their readable form are required and obtained from a multi-modal model employed in the FSS. Additionally, the original image and the bounding boxes corresponding to the visual region embeddings are necessary and loaded into memory from specified locations on the disk.

The component combines all this information to annotate the target regions in the images with a red border and the readable form of the textual tokens. The annotated images are persisted on disk and registered at the image server component to make them available via URL. For multiple images, this is done in parallel to meet the "real-time" requirements of MMIRS.

9. https://matplotlib.org/

An example of a resulting image with the maximum focus region highlighted is shown in Figure 9.10.

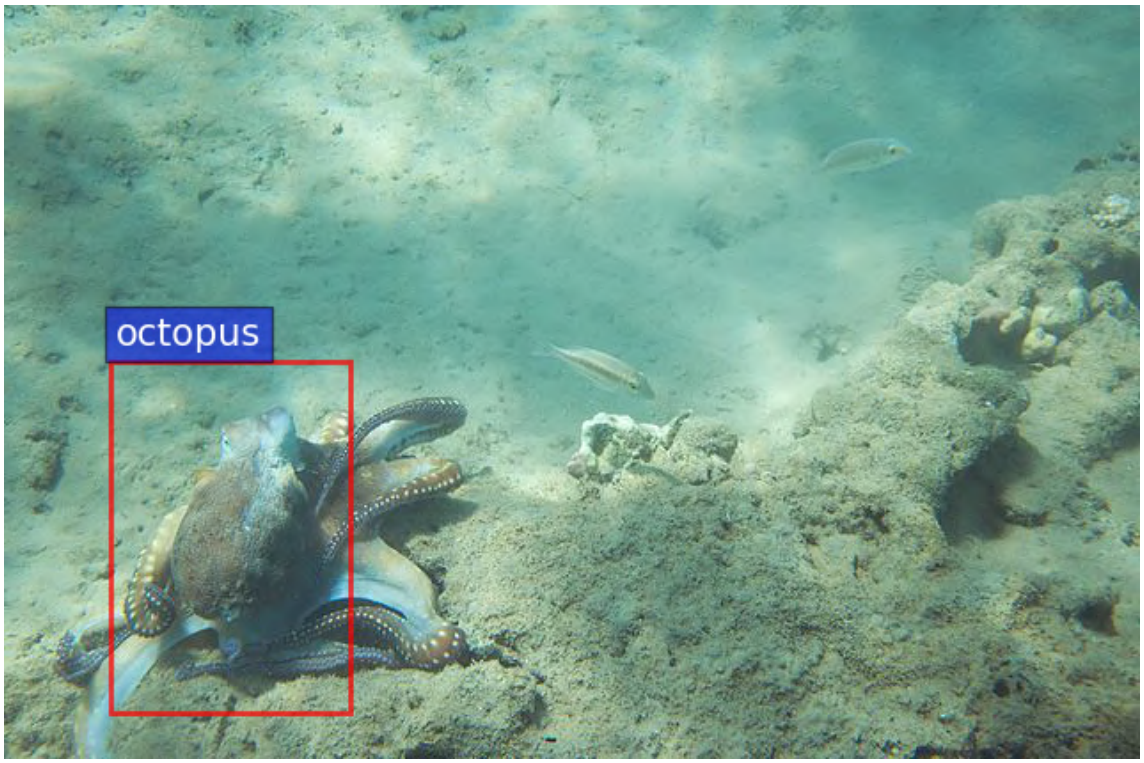


Figure 9.10: An example image with the maximum focus region highlighted, which was generated by the Focus Region Highlighter auxiliary component of MMIRS. The context of this example query was *"An octopus can adapt to a large variety of colors and patterns to camouflage on the sandy bottom."*, and the focus was *"octopus"*.

9.7.4 REST API

The REST API realized with the Python libraries FastAPI[10] and pydantic[11] provides access to MMIRS text-image retrieval functionalities via different endpoints, described in the following tables.

9.8 User Interface

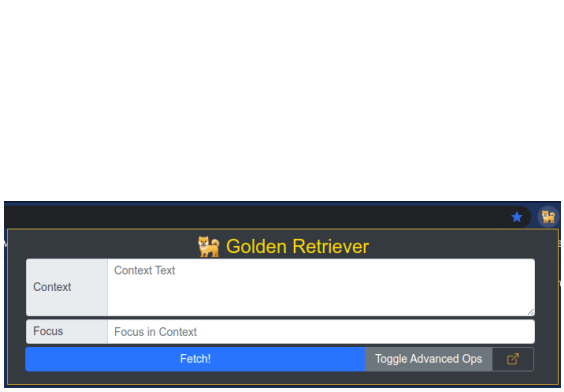
A simple browser plugin for Google Chrome called "Golden Retriever" was developed to provide access for users to MMIRS text-image retrieval functionalities. Basically, the "Golden Retriever" serves as a user-friendly client to consume MMIRS' API (see Section 9.7.4), and implements the practical use-case of the language learner scenario as described in Section 1.1.1. Once installed, the plugin can be opened by clicking on the respective icon (showing a golden dog) in the browser's extensions bar. With the right-most button on the bottom area of the user interface shown in the following figures, the plugin can be opened in a new tab for convenience.

When the plugin is opened, it provides a straightforward interface shown in Figure 9.11a to retrieve the most similar images for a context and a focus, i.e., a query, for non-technical

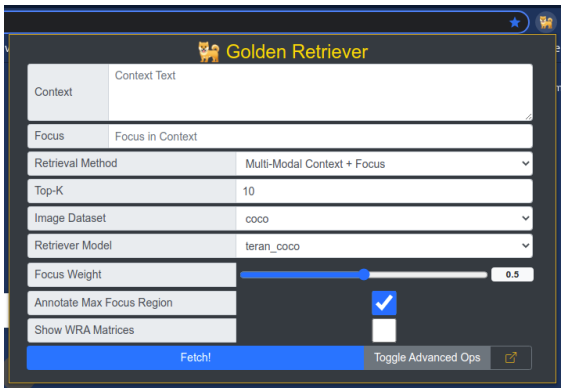
10. https://fastapi.tiangolo.com/
11. https://pydantic-docs.helpmanual.io/

Method	Endpoint	Parameters	Description
POST	/top_k_images	see Table 9.4	Fully multi-modal text-image retrieval based on context and focus via PSS and FSS (see Section 9.2).
POST	/pss/top_k_context	see Table 9.5	Context-only image retrieval via PSS (see Section 9.5.2)
POST	/pss/top_k_focus	see Table 9.6	Focus-only image retrieval via PSS (see Section 9.5.1)
GET	/available_datasets	none	Returns the available datasets
GET	/available_retrievers	none	Returns the available retrieval models for the FSS

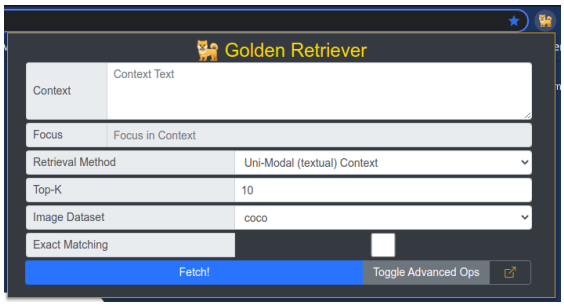
Table 9.3: Overview of the REST API endpoints to access MMIRS text-image retrieval functionalities.



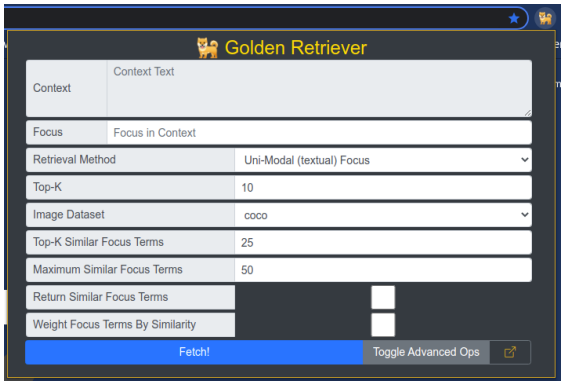
(a) Simple UI of the Golden Retriever browser plugin.



(b) Advanced UI of the Golden Retriever browser plugin.



(c) Advanced UI of the Golden Retriever browser plugin to trigger a context-based text-image retrieval (see Section 9.5.2).



(d) Advanced UI of the Golden Retriever browser plugin to trigger a focus-based text-image retrieval (see Section 9.5.1).

Figure 9.11: Different views of the Golden Retriever browser plugin, that serves as the user interface of MMIRS.

users like language learners. With the right-most button on the bottom area of the user interface, the plugin can be opened in a new tab for convenience.

For research purpose or advanced users, the plugin also offers an interface shown in Figure 9.11b with advanced options which can be shown or hidden via the respective button. The available options and parameters of this advanced UI are described in Table 9.4.

To retrieve images solely by a context text via the Preselection Stage (PSS) of MMIRS, the user choose it via the dropdown described as “Retrieval Method”. The interface then

Name	Type	Description
context	string	Context part of a query
focus	string	Focus part of the query, which has to be a substring of the context
top_k	int	Number of images returned by MMIRS. This parameter is optional with 10 as default value.
retriever	string	Name of the multi-modal model that will be used to retrieve the top-k images
dataset	string	Name of the dataset in which MMIRS will search for the top-k images
focus_weight	float	The weight of the focus when computing the combined ranks. The context is not taken into account if the value is 1.0, and the focus is not taken into account if the value is 0.0 (see Section 9.6.3). This parameter is optional with 0.5 as default value.
annotate_max_focus_region	boolean	If true, MMIRS annotates the regions in the top-k images, with the highest similarity to the focus (see Section 9.7.3). This parameter is optional with "False" as default value.
return_wra_matrices	boolean	If true, MMIRS returns the WRA matrices for each of the top-k images (see Section 9.7.2). This parameter is optional with "False" as default value.
return_scores	boolean	If true, MMIRS returns the similarity scores for each of the top-k images. This parameter is optional with "False" as default value.

Table 9.4: Parameters of the REST API endpoint for full multi-modal text-image retrieval with MMIRS utilizing the PSS and the FSS (see Section 9.2).

updates to the one depicted in Figure 9.11c, and the now available parameters are described in Table 9.5.

Similarly, if a user wants to retrieve images only by focus term(s) via the PSS of MMIRS, she can choose to do so via the respective dropdown. The interface updates as shown in Figure 9.11d and the available parameters are described in Table 9.6.

Once the images are retrieved, they are presented to the user in an interactive slide show component, as shown in Figure 9.12.

By clicking on an image, the image in full resolution is opened in a new tab. This is especially useful for large WRA matrix plots, as they appear small due to their large width for long context sentences (see Figure 9.12b).

Returned results are stored until the user clicks the "Fetch again" button on the bottom area, shown in Figure 9.12.

9.9 System Analysis

Since a comprehensive system evaluation, e.g., via a user study similar to the ones described in Chapter 8, is out of the scope of this thesis, the following analysis is only done by myself. Following interpretations and assessments of results are therefore subjective and do not claim completeness or objectivity.

Name	Type	Description
context	string	The context for which the PSS searches the top-k similar images
top_k	int	Number of images returned by MMIRS. This parameter is optional with 10 as default value.
dataset	string	Name of the dataset in which the PSS will search for the images
exact	boolean	If true, the top-k similar captions are to the context are computed exactly as opposed to an approximate search (see Section 9.5.2). This parameter is optional with "False" as default value.

Table 9.5: Parameters of the REST API endpoint for context-based text-image retrieval with MMIRS's Preselection Stage (PSS). For more information see Section 9.5.2
.

9.9.1 "Real-Time" Capability

In the following, timings of MMIRS and its sub-components are reported to assess the system's suitability in a "real-time" application. Note that "real-time" in the context of MMIRS is always in parentheses because it must not be confused with "true" real-time systems as defined in the context of robotics or real-time operating systems like RTOS[12]. Unlike these real-time systems, which must guarantee a system response in a precisely defined time, MMIRS "real-time" requirements are not exactly specified but are merely subjective constraints by the means that users should not have to wait too long for a response of the system.

Multiple factors have varying influence MMIRS multi-modal text-image retrieval response time. To find how much these factors weigh, the "real-time" assessment test of MMIRS reported in this section was conducted as follows: The system's multi-modal text-image query API endpoint (see Table 9.7.4) was consumed with different parameter combinations. Each of three queries (see Table 9.7) parameters was combined with four different modes (see Table 9.8) for the COCO, Flickr30k, and WISMIR v2 datasets and TERAN models. This results in a set of $3 * 4 * 3 = 36$ different parameter combinations, for which the average system response time over 10 consecutive was measured. For the measurements, an MMIRS internal fine-grained timing component was utilized so that the latency depending on the users' bandwidth is not taken into account. As it can be observed from the results presented in Figure 9.13 and Figure 9.14, the length of the context part of the query and the generation of WRA matrix plots affect the system's response time the most. This is an expected result since the similarity of an image is based on pooling the WRA matrix, representing the fine-grained similarity of each textual and visual token. Hence, the longer the context, the larger the WRA matrix and the more time the retrieval model takes to generate and pool the matrix. This logic also applies to the generation of the WRA matrix plots by the respective component (see Section 9.7.2), which is also part of the FSS. Highlighting the maximum focus region (see Section 9.7.3), however, only has a marginal impact on the system's overall response time.

Further, the effect of the PSS can be noticed: The larger the dataset is, from which MMIRS retrieves the top-k images, the longer the PSS takes, whereas the average response time of the FSS remains almost across different datasets. Flickr30k has about 31K, COCO about 123K, and WISMIR v2 about 395K images, and the corresponding average PSS response times are 0.09s, 0.27s, and 0.52s, respectively. This increase of time of the PSS is almost linearly proportional to the number of unique images in datasets. These results also highlight the effectiveness of the two-stage retrieval approach of MMIRS as introduced in Section 9.2.

As mentioned in the introduction of this section, the interpretation of the resulting

12. https://www.freertos.org

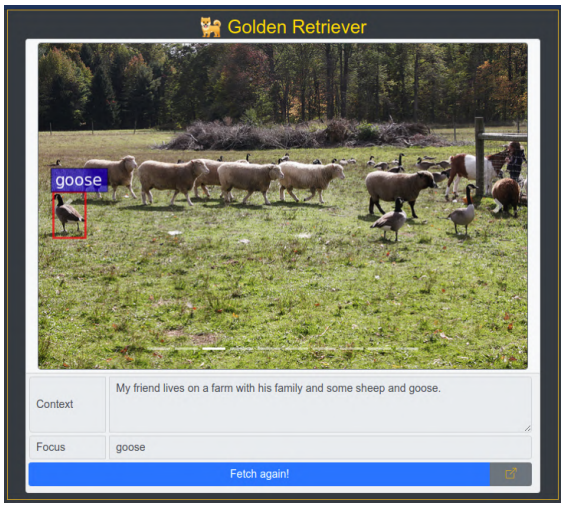
Name	Type	Description
focus	string	The focus term(s) for which the PSS searches the top-k similar images
top_k	int	Number of images returned by MMIRS. This parameter is optional with 10 as default value.
dataset	string	Name of the dataset in which the PSS will search for the images
top_k_similar_terms	int	Number of similar terms searched and taken into account for each term in the focus (see Section 9.5.1). This parameter is optional with 10 as default value.
max_similar_terms	int	Maximum number of similar taken into account for all terms in the focus (see Section 9.5.1). This parameter is optional with 25 as default value.
weight_by_sim	boolean	If true, the retrieved images per similar term are re-ranked based on the similarity to the respective focus term (see Section 9.5.1). This parameter is optional with "false" as default value.
return_similar_terms	boolean	If true, MMIRS returns the list of similar terms found and taken into account for the retrieval. This parameter is optional with "false" as default value.

Table 9.6: Parameters of the REST API endpoint for focus-based text-image retrieval with MMIRS's Preselection Stage (PSS). For more information see Section 9.5.1

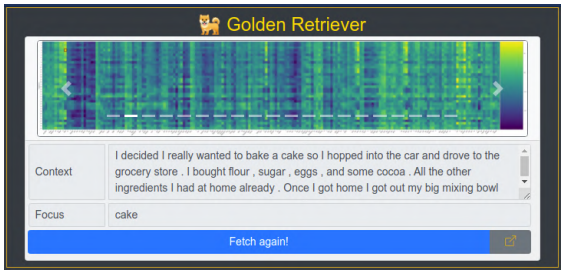
.

system response times is subjective due to the lack of an objective evaluation like a user study and the lack of a widely accepted definition of acceptable response times. However, there exists a loose definition of "near-real-time" systems, according to which there must not be "significant delays"[13]. As stated in the corresponding Wikipedia article, this "delay in near real-time is typically in a range of 1-10 seconds"[14]. As depicted in Figure 9.14, the overall average system response time across all datasets, queries, and modes evaluated in this "real-time" suitability test of MMIRS is 2.10s. Hence, in conclusion, it is considered as an acceptable result.

13. https://www.its.bldrdoc.gov/fs-1037/dir-024/_3492.htm – visited on 06.07.2021
14. https://en.wikipedia.org/wiki/Real-time_computing#Near_real-time – visited on 06.07.2021



(a) View of the Golden Retriever UI showing an example of a retrieved image via multi-modal text-image retrieval with MMIRS. The focus part ("goose") of the query highlighted in the image.



(b) View of the Golden Retriever UI showing an example WRA matrix of a retrieved image and a long context, generated by MMIRS during a multi-modal text-image retrieval.

(c) View of the Golden Retriever UI showing an example of a retrieved image via context-based text-image retrieval with the PSS of MMIRS (see Section 9.5.2). By clicking on the generated image, a high-resolution version opens in a new tab.

(d) View of the Golden Retriever UI showing an example of a retrieved image via focus-based text-image retrieval with the PSS of MMIRS (see Section 9.5.1). The similar terms to the focus "sheep", found by the PSS, are shown beneath the image.

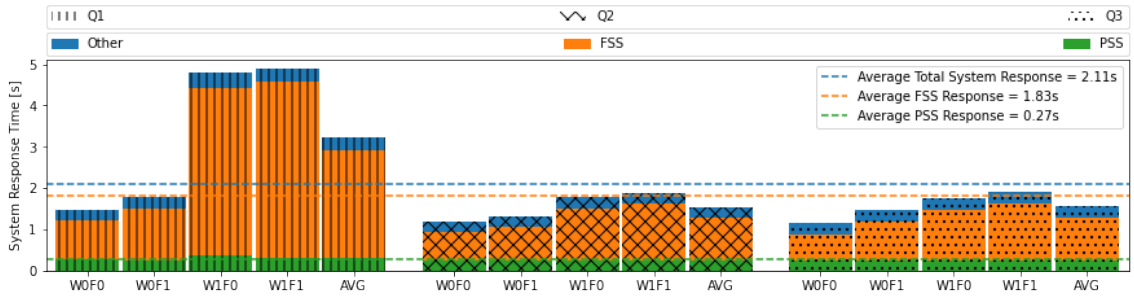
Figure 9.12: Different views of example text-image retrieval results in the Golden Retriever browser plugin, the user interface of MMIRS.

Query	Chars	Focus	Context
$Q1$	827	spoons	I decided I really wanted to bake a cake so I hopped into the car and drove to the grocery store. I bought flour, sugar, eggs, and some cocoa. All the other ingredients I had at home already. Once I got home I got out my big mixing bowl and measuring cups and spoons. I pre-heated the oven to 400 degrees and started pouring ingredients into the mixing bowl. After all the ingredients were mixed, I greased a cake pan and poured in the batter. I put the pan in the oven and started mixing the ingredients for the frosting in a new bowl. After baking for a while, I took the cake out and stuck in a toothpick to make sure the center was done. I let the cake cool for a bit and then put the frosting on it. After frosting went on, I cut it into pieces and put one on a plate for myself and another for my friend.
$Q2$	124	Stingrays	Stingrays exhibit a wide range of colors and patterns on their dorsal surface to help them camouflage with the sandy bottom.
$Q3$	67	goose	My friend lives on a farm with his family and some sheep and goose.

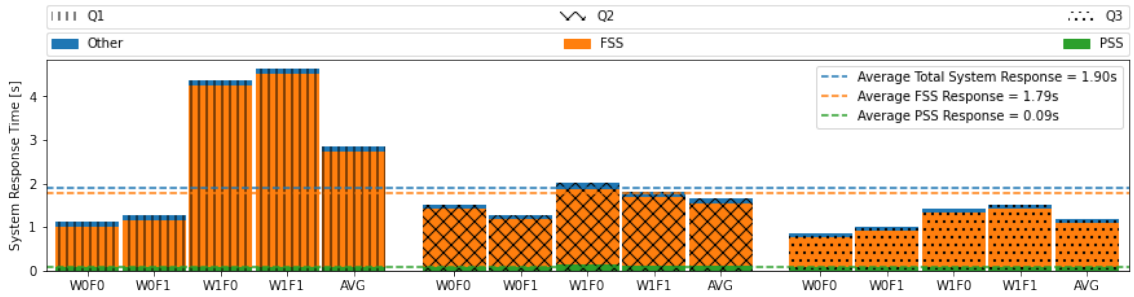
Table 9.7: Different queries used during the "real-time" assessment tests of MMIRS described in Section 9.9.1.

Mode	Generate WRA Matrix Plots	Annotate Max. Focus Regions
$W0F0$	False	False
$W0F1$	False	True
$W1F0$	True	False
$W1F1$	True	True

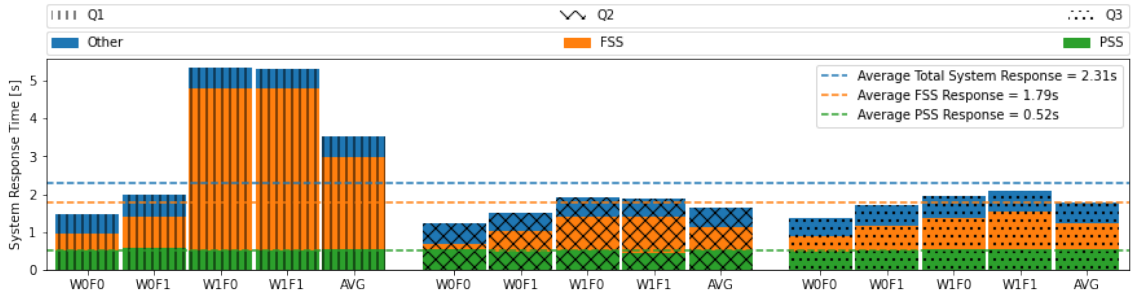
Table 9.8: Different text-image retrieval modes used during the "real-time" assessment tests of MMIRS described in Section 9.9.1. For more information about the generation of WRA matrix plots or the annotation of the maximum focus region, see Section 9.7.2 and Section 9.7.3, respectively.



(a) Averaged timing measurements of MMIRS' system response time on the COCO dataset.



(b) Averaged timing measurements of MMIRS' system response time on the Flickr30k dataset.



(c) Averaged timing measurements of MMIRS' system response time on the WISMIR v2 dataset.

Figure 9.13: Averaged timing measurements of MMIRS' system response time for multiple queries ($Q1$, $Q2$, $Q3$) and modes ($W0F0$, $W0F1$, $W1F0$, $W1F1$) on different datasets utilizing TERAN models. The queries and modes are described in Table 9.7 and Table 9.8, respectively. Each bar represents the total system response time, which comprises the response times of the Preselection Stage (PSS) in green, the Fineselection Stage (FSS) in orange, and additional overhead from subcomponents like the static image server in blue.

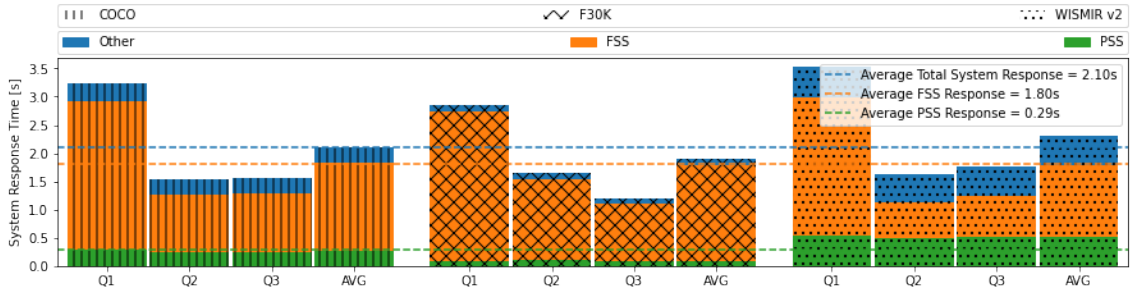


Figure 9.14: Averaged timing measurements of MMIRS' system response time for multiple queries $Q1$, $Q2$, and $Q3$ (see Table 9.7) on different datasets. Each bar represents the total system response time, which comprises the response times of the Preselection Stage (PSS) in green, the Fineselection Stage (FSS) in orange, and additive overhead in blue.

10 Conclusion and Future Work

In this thesis, the four research questions regarding the general applicability of current multi-modal methods for text-image retrieval within the scope of a language learner scenario to improve human reading were examined. In the following, first comes the general conclusion and afterwards the research questions of this thesis are answered with their solution approaches summarized.

10.1 General Conclusion

This thesis investigated multiple important steps towards leveraging state-of-the-art multi-modal text-image retrieval methods to improve human reading within a language learner scenario.

A new dataset based on "in the wild" Wikipedia data was collected to evaluate the performance of current text-image retrieval models with long and complex textual queries. Multiple experiments showed that the models perform poorly on this dataset compared to how they perform in the task when using only short and simple queries.

Nevertheless, two user studies found that the investigated models are still generally capable of retrieving relevant images for textual queries as it is required in the scenario. However, there is still much space for improvement considering the performance and computational efficiency of these models.

Further, a technique was devised to incorporate textual queries, which comprise a context sentence and a focus term contained therein, for text-image retrieval methods, as required by the language learner scenario. For the final usage in the scenario, the algorithm's parameters yet are to be optimized and thoroughly tested.

Finally, a "real-time" capable multi-modal text-image retrieval system powered by the investigated state-of-the-art multi-modal transformer model was developed. This system can handle arbitrary textual queries comprised of a context and a focus term. With the system's user interface, which is realized through a browser plugin, a first proof-of-concept solution for the practical use-case of the language learner scenario is presented. However, the system still has to be evaluated to assess the suitability for a real-world language learner scenario, especially whether it can be leveraged to effectively improve the language learners' reading comprehension.

To conclude, this work answered and solved initial research questions and engineering challenges towards leveraging current multi-modal text-image retrieval methods for the practical use-case of our language learner scenario. However, effectively improving human reading within a real-world language learner scenario is a comprehensive and challenging task, which requires much future work as suggested in Section 10.6.

10.2 Research Question 1 (RQ1)

To answer the first research question (RQ1), "How do state-of-the-art multi-modal transformers perform in text-image retrieval with complex and lengthy textual queries?" multiple steps were required.

First, available multi-modal transformers employed for text-image retrieval were investigated and compared with respect to the suitability for our language learner scenario.

Here, the performance in terms of evaluation metrics on the task, the computational efficiency – crucial for a practical use-case – and the availability of pre-trained models was important. Finally, TERAN, a state-of-the-art multi-modal model designed explicitly for efficient text-image retrieval, was chosen because it represents the best trade-off between computational efficiency and performance on the task.

Second, a new dataset called WISMIR was created. This was required since existing pretrained models utilized for the text-image retrieval task are trained and evaluated on datasets that comprise short, simple, and descriptive captions for the corresponding images. WISMIR is a subset of the WikiCaps, a large-scale multi-modal dataset containing "in the wild" text-image pairs crawled randomly from Wikipedia and Wikimedia. To ensure longer and more complex captions, an ETL pipeline tool was developed to filter the WikiCaps dataset accordingly and create the WISMIR dataset. Further, an in-depth analysis was conducted to prove the textual disparity of WISMIR to COCO and Flickr30k, the two most popular datasets for text-image retrieval.

In the third and final step, several TERAN models trained on different versions of WISMIR, COCO, or Flickr30k were evaluated on the datasets' test splits. Moreover, their performances in terms of the binary evaluation metric Recall@K were compared and discussed. This comparison showed that the TERAN models trained and evaluated on WISMIR performed much worse than TERAN models trained on COCO or Flickr30k – independent of the test set used for evaluation. Further, it was observed that training on COCO or Flickr30k resulted in inferior performance when evaluating on WISMIR. Very similar but not so severely deficient results can be observed vice versa.

To conclude and answer the first research question: it was found that current text-image retrieval models perform poorly in retrieving images for lengthy and complex textual queries, although they perform much better on short and simple queries.

However, the performance was assessed via binary evaluation metrics, which ignore relevant but not exact matches. For example, if the top-5 retrieved images are very similar and all are relevant for the textual query, but the ground truth image from the dataset is not in these top-5, binary evaluation metrics will assess 0% performance for this sample. This means that the evaluated models are not necessarily unsuitable for the language learner scenario, where not only the exact best image is relevant, but all retrieved images, which are similar to the exact match. This issue is addressed with the second research question as described in the following.

10.3 Research Question 2 (RQ2)

Two user studies were conducted to find an answer to the second research question (RQ2), "Are state-of-the-art multi-modal transformers applied on text-image retrieval suitable for our language learner scenario?". Both user studies, including their pilot studies to find genuine, were conducted on MTurk utilizing the tool developed for this purpose within the scope of this thesis.

In the first (smaller) user study, the text-image retrieval performance of a TERAN model trained and evaluated on WISMIR was assessed with the help of human raters. The raters' task was to rank their top-5 images according to a given caption text from a pool of 12 images comprising the top-5 images ranked by the model and 7 random images. The submitted results were analyzed, and the agreement between the raters and the model was examined. From these results, it can be said that the model's top-5 ranked images are relevant to the corresponding captions for the large majority of 72.5% of the evaluated samples. Further, based on the human assessments, the model's performance was measured by the non-binary NDCG metric common to evaluate information retrieval systems where not only the first ranked result is a relevant result. According to this metric, the model's

performance is very strong and much more optimistic than the results according to the binary evaluation metrics used to answer the first research question. However, the sample size of the study was too small for a solid estimate of the model's performance.

For the second user study, three different text-image retrieval models ranked their top-5 images from three different image pools for captions from a text-only language learner dataset on a 5-star scale. The task of the human raters was to assess how related these images are to the respective caption on a five-star scale. The employed text-image retrieval models were three TERAN models trained on WISMIR, COCO, and Flickr30k, respectively. These models retrieved images from three image pools comprising all unique images from the training and test sets of COCO, Flickr30k, and WISMIR, respectively. The text-only language learner dataset from which the captions – i.e., the queries for which the models retrieved the top-5 images – originated was manually collected from a student assistant and contains multiple short stories about different topics. From analyzing the submitted results, it was found that there were differences in the average quality of the top-5 images from up to one star, depending on the employed model as well as the image pool from which the model retrieved the top-5. This finding supported the initial expectation that models trained on data more akin to the utilized language learner dataset perform better. However, the human rates gave about 3 or 3.5 stars on average or median across all models, image pools, and image ranks. While these results show that there is still much space for improvement, the models' performance can be considered acceptable.

To conclude and answer the second research question: the results of the two user studies suggest that the employed models are generally suitable for our language learner scenario. Nevertheless, as described in the future work section, more comprehensive research must be done to find a more precise answer.

10.4 Research Question 3 (RQ3)

The solution found and realized in this work to the third research question (RQ3), "How can textual queries consisting of a context and a focus contained therein be supported in multi-modal text-image retrieval methods so that the retrieved images correspond to both the context and the focus?" is as follows: Multi-modal text-image retrieval transformers usually generate a word-region-alignment (WRA) matrix to compute the global cross-modal similarity between a text (the context) and an image. The cells of a WRA matrix represent the fine-grained similarity or alignment between a textual token and a visual region. The TERAN model, primarily utilized throughout this thesis, pools this WRA matrix to compute the global similarity between a text and an image. Ideally, the best matching images already correspond to every token in the textual input (including the focus token), but that must not be the case, especially for extended contexts. Moreover, in our language learner scenario, the focus is more important than the context. To ensure that the top-ranked images show the focus token while also representing the context to the best extent, a ranking algorithm based on WRA matrices was developed. This algorithm first pools the WRA matrices to compute a score for the focus token and a score for the full context. Then it ranks all images based on a combined score which is a weighted average of the focus score and context score.

To solve the problem stated by the third research question, a ranking algorithm considering the focus as well as the context was developed on top of current text-image retrieval models. However, the algorithm has several hyperparameters for which the optimal set has yet to be found for a practical use case. Another promising approach to this research question is described in the future work section.

10.5 Research Question 4 (RQ4)

The fourth research question (RQ4), "How to leverage state-of-the-art multi-modal transformers in a practical application, i.e., a "real-time" text-image retrieval system with a large pool of images?", was solved through an engineering solution approach. More specifically, the multi-modal text-image retrieval system (MMIRS) powered by efficient state-of-the-art visio-linguistic transformers (TERAN models) was developed. MMIRS solves the "real-time" requirement on the practical use-case of the language learner scenario with a two-stage process. In the first stage of the system, the large image pool from which the best images are retrieved gets drastically reduced by efficient uni-modal filtering methods. On systems with modern hardware, this process usually takes only a few hundred milliseconds. The resulting unordered set comprises several hundred to a few thousand images that roughly match the context or the focus part of the query. In the second stage, a TERAN model ranks the images in the reduced set by computing the cross-modal similarity between the query and each image. However, even computing the similarities for the reduced set of images would be impracticable for a "real-time" critical application with the standard approach. To solve this issue, the late-fusion architecture of TERAN is exploited. Since TERAN internally consists of two separate uni-modal transformers, it is possible to pre-compute the embeddings for all images in the large image pool. Further, the embedding for the query only has to be computed once per retrieval operation and can be re-used to compute the cross-modal similarity between an image. This saves an enormous amount of computation when computing similarity scores between a single query and lots of images. Using this approach, the average system response time is about 2 seconds, which is considered acceptable for a "real-time" application.

To solve the issue stated by the fourth research question, a multi-modal text-image retrieval system was developed, which first reduces the large image pool and finally utilizes an efficient multi-modal transformer to compute text-image similarities. The system retrieves the best matching images for a textual query within about 2 seconds on average, which fulfills the "real-time" requirement of the practical use-case for the language learner scenario. Nevertheless, there is still much space for optimizing the system, which is sketched next in the future work section.

10.6 Future Work

Since several broad topics were investigated in this work, and also the primary subject of the thesis is comprehensive, covering multiple different issues, lots of future work is possible and necessary to effectively apply current multi-modal text-image retrieval methods to improve human reading. In the following, ideas and suggestions for future work are separately described per topic.

10.6.1 WISMIR Dataset

The WISMIR is still a work-in-progress dataset with limitations and room for improvements, as described below:

Data Cleaning The dataset has to be cleaned further to remove samples with the following characteristics:

- Too small images
- Images that show abstract things like icons or plots of mathematical functions
- Captions containing incomplete or grammatically incorrect sentences
- Captions that are not entirely English
- Captions containing too many named entities

- Captions with too loose coupling with the image, i.e., captions where too many tokens are not represented in the image

Named Entity Augmentation Since the text-image pairs contained in WISMIR are crawled from Wikipedia, the captions contain many named entities. This might be problematic for multi-modal models when learning joint representations or word-region alignments, especially for named entities with only a few occurrences, because the model needs to learn what the named entity represents to find the corresponding region in an image. For example, humans are all depicted similarly independent of their names, and the same is valid for all other kinds of entities. One idea to solve this issue is to augment named entities by replacing them in the captions with the entities they represent. This could be done with the help of Knowledge Graphs like DBPedia[1] or other ontologies.

Dataset Size The samples removed by the cleaning process described above should be replaced with other samples so that the size of the dataset does not decrease. Or even better: since more data is always beneficial to train deep neural networks, more samples should be collected in general to increase the size of WISMIR.

Further, the test set of WISMIR has to be analyzed and revised so that it is not too similar to the training set and is well distributed.

10.6.2 TERAN Performance

To increase the TERAN's text-image retrieval performance, multiple approaches described below are promising and should be investigated.

Training Data The performance of TERAN, which is the primary focussed model of this work, can most probably be improved. First, the investigated pre-trained TERAN models could be trained on much more and diverse data. That is, instead of training it separately on COCO, Flickr30k, or WISMIR, the dataset can be combined to create a much larger and more heterogeneous training set. Also other popular multi-modal datasets like Visual Genome (Krishna et al. 2017), Conceptual Captions (Sharma et al. 2018), or SBU Captions (Kuznetsova et al. 2013) can be used additionally to increase the training set size further.

Another promising and very recent – and therefore unconsidered – dataset is WIT (Srinivasan et al. 2021). This very large-scale dataset created by Google researchers is based on Wikipedia data but might be "cleaner" than WikiCaps (Schamoni et al. 2018) or WISMIR. Increasing the heterogeneity of the training data could be especially beneficial when utilizing the dataset for the language learner scenario since there, the model has to perform well for arbitrary texts.

Hyperparameter Optimization In this work, the standard hyperparameter set from the TERAN authors is used to train the model. However, the authors did not mention any justification for these parameters, so tuning the hyperparameters, e.g., via grid search, will most probably increase the model's performance.

Features The features employed by the authors of TERAN and in this work might not be suitable when utilizing the model for multi-modal text-image retrieval, especially for Wikipedia-based data, for the following reasons: The textual features, which are the output of a pre-trained BERT-based tokenizer model (Devlin et al. 2019), are Byte Pair or sub-word embeddings. That is, words are broken up into smaller pieces if they are

1. https://www.dbpedia.org/

uncommon or to separate the stem and common morphological structures. When learning word-region alignments, the single sub-word embeddings are aligned with the regions, which is problematic. For example, the word "Stingrays" is split by the tokenizer model into three tokens: "Sting##", "##ray", and "##s". The multi-modal model now tries to align those tokens individually to regions, which is obviously problematic when retrieving images for queries including the word "Stingrays". Hence, other textual feature embeddings should be investigated and evaluated.

Also, the visual features might impose an issue for text-image retrieval with Wikipedia-based data. The employed Faster R-CNN with ResNet-101 (Anderson et al. 2018) model to extract the visual regions and features is pre-trained on ImageNet (Deng et al. 2009) and Visual Genome. However, the datasets comprise images of common objects or everyday life, which are probably different from images for Wikipedia articles, often showing particular objects. Therefore, when extracting features from Wikipedia images, the employed model might produce different low-quality features and regions. This issue should be investigated, and alternative models for visual feature extraction should be examined.

Another possible issue induced by the visual features employed to train TERAN models might be that the number of regions and features extracted per image is limited to 36. However, the previously mentioned visual feature extraction model can extract up to 100 regions and features per image. Having more features per image might be especially beneficial when training the model on extended captions so that a tighter coupling between the textual tokens and visual regions is possible. Hence, it should be investigated if TERAN's performance increases when utilizing more visual features per image.

10.6.3 MMIRS

Currently, the multi-modal text-image retrieval system, MMIRS, is a proof-of-concept application. To deploy it for a real-world language learner scenario effectively, the system requires much more elaborate testing and evaluation, as suggested below.

Parameter Tuning MMIRS is a complex system that comprises several components, each configurable by many parameters. In the current version, those parameters were found manually via trial-and-error so that the system's basic functionality operates acceptably. However, they are most probably by far not set optimally and further likely depend on the final use-case of MMIRS. So, to finally deploy MMIRS for the language learner scenario, the optimal set of parameters has yet to be found. Since the performance of MMIRS in such a scenario cannot be measured via evaluation metrics, the set of optimal parameters cannot be optimized computationally but has to be assessed by users of the application. This could either be accomplished via user studies or by A/B testing strategies.

Sub-Component Analysis As already mentioned, MMIRS is made up of several sub-components. Especially the components in the first stage of MMIRS that drastically reduce the large pool of images should be further investigated to assess their effectiveness and performance. Those components employ several pre-trained publicly available neural networks or other parametrized models to solve different tasks. However, there exists a large variety of possible models that solve the same or very similar tasks of which the optimal for the final use-case of MMIRS yet has to be determined. Support For Other Multi-Modal Text-Image Retrieval Models The system follows a modular and easily extensible software architecture to support different text-image retrieval models. However, in the current version of MMIRS, only TERAN models are fully supported. TERAN was chosen because it is computationally efficient while still performing well in text-image retrieval tasks. Nevertheless, it would be interesting to evaluate the performance of MMIRS powered by other models that are either mode performant or computationally efficient like, e.g., UNITER or non-transformer models like CAAN or SCAN.

Larger Image Pools One of the user studies showed that not only the employed model but also the pool of images affects the quality of the retrieved images in text-image retrieval. So employing a larger and therefore more diversified pool of images would most certainly improve the overall performance of MMRIS. In the current version of MMIRS, images from COCO, Flickr30k, and WISMIR are supported only separately. That is, the system retrieves images from one of the datasets but not from the combined pool of images. While it sounds trivial to combine the image pools to create a larger and more heterogeneous pool, it actually requires some effort due to the first stage of MMIRS, which filters those image pools via uni-modal methods that depend on pre-computed clusters and indices. Additionally, other popular multi-modal text-image datasets like those introduced in Section 4.1 should be considered to increase the image pool further.

However, employing a larger pool of images also increases the execution time of the first stage of the system that preselects promising image candidates. So increasing the image pool size might be an issue to the "real-time" constraint of MMIRS.

Hence, the optimal or maximum size of the employed image pool has to be assessed.

Assessing the Effectiveness in a Real-World Scenario While MMIRS is designed to suit the language learner scenario generally, the system's effectiveness in a real-world scenario and how it can be improved still has to be evaluated. That is, it has to be assessed if language learners can actually improve their reading comprehension with the aid of the system. This has to be assessed via user studies with actual language learners. For example, one could imagine a user study where the participants are divided into two groups: One group can utilize MMIRS while the other group can not. Then the participants are asked to solve reading comprehension tasks, and the final average performance of both groups is compared.

Bibliography

Albahiri, M. H. and Alhaj, A. A. M. 2020. "Role of visual element in spoken English discourse: implications for YouTube technology in EFL classrooms." *The Electronic Library* 38 (3): 531–544.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. 2018. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6077–6086. Salt Lake City, UT, USA.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. 2015. "VQA: Visual Question Answering." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* 2425–2433. Santiago, Chile.

Ba, J. L., Kiros, J. R., and Hinton, G. E. 2016. "Layer Normalization." *arXiv preprint arXiv:1607.06450.*

Bahdanau, D., Cho, K. H., and Bengio, Y. 2015. "Neural Machine Translation by Jointly Learning to Align and Translate." In *3rd International Conference on Learning Representations (ICLR).* San Diego, CA, USA.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. 2017. "Enriching Word Vectors with Subword Information." *Transactions of the Association for Computational Linguistics (TACL)* 5:135–146.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., et al. 2020. "Language Models are Few-Shot Learners." In *Advances in Neural Information Processing Systems (NIPS),* 33:1877–1901. Online.

Chall, J. S. and Dale, E. 1995. *Readability revisited: The new Dale-Chall readability formula.* Brookline Books.

Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J. 2020. "UNITER: UNiversal Image-TExt Representation Learning." In *European Conference on Computer Vision (ECCV),* 104–120. Online.

Cho, K., Merriënboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. 2014. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 1724–1734. Doha, Qatar.

Cicchetti, D. V. 1994. "Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology." *Psychological assessment* 6 (4): 284.

Dalton, B. and Grisham, D. L. 2011. "eVoc Strategies: 10 Ways to Use Technology to Build Vocabulary." *Reading Teacher* 64 (5): 306–317.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. 2009. "ImageNet: A large-scale hierarchical image database." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 248–255. Miami, FL, USA.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics,* 4171–4186. Minneapolis, MN, USA.

Ecalle, J., Magnan, A., Bouchafa, H., and Gombert, J. E. 2009. "Computer-Based Training with Ortho-Phonological Units in Dyslexic Children: New Investigations." *Dyslexia* 15 (3): 218–238.

Elliott, D., Frank, S., Sima'an, K., and Specia, L. 2016. "Multi30K: Multilingual English-German Image Descriptions." In *Proceedings of the 5th Workshop on Vision and Language,* 70–74. Berlin, Germany.

Farr, J. N., Jenkins, J. J., and Paterson, D. G. 1951. "Simplification of Flesch reading ease formula." *Journal of applied psychology* 35 (5): 333.

Francis, W. N. and Kucera, H. 1979. "Brown Corpus Manual." *Letters to the Editor* 5 (2): 7.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Reading.

Gerbier, E., Bailly, G., and Bosse, M. L. 2018. "Audio–visual synchronization in reading while listening to texts: Effects on visual behavior and verbal learning." *Computer Speech & Language* 47:74–92.

Girshick, R. 2015. "Fast R-CNN." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* 1440–1448. Santiago, Chile.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. 2014. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Columbus, OH, USA.

Güler, R. A., Neverova, N., and Kokkinos, I. 2018. "DensePose: Dense Human Pose Estimation In The Wild." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 7297–7306. Salt Lake City, UT, USA.

Hahn, N., Foxe, J. J., and Molholm, S. 2014. "Impairments of Multisensory Integration and Cross-Sensory Learning as Pathways to Dyslexia." *Neuroscience & Biobehavioral Reviews* 47:384–392.

He, K., Zhang, X., Ren, S., and Sun, J. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 770–778. Las Vegas, NV, USA.

Hochreiter, S. and Schmidhuber, J. 1997. "Long short-term memory." *Neural Computation* 9 (8): 1735–1780.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. 2020. *spaCy: Industrial-strength Natural Language Processing in Python.* https://spacy.io/.

Hudson, D. A. and Manning, C. D. 2019. "GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA).

Järvelin, K. and Kekäläinen, J. 2002. "Cumulated Gain-Based Evaluation of IR Techniques." *ACM Transactions on Information Systems* (New York, NY, USA) 20 (4): 422–446.

Johnson, J., Douze, M., and Jégou, H. 2019. "Billion-scale similarity search with GPUs." *IEEE Transactions on Big Data.*

Johnson, J., Karpathy, A., and Fei-Fei, L. 2016. "DenseCap: Fully Convolutional Localization Networks for Dense Captioning." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 4565–4574. Las Vegas, NV, USA.

Kabooha, R. and Elyas, T. 2018. "The Effects of YouTube in Multimedia Instruction for Vocabulary Learning: Perceptions of EFL Students and Teachers." *English Language Teaching* 11 (2): 72–81.

Kazemzadeh, S., Ordonez, V., Matten, M., and Berg, T. 2014. "ReferItGame: Referring to Objects in Photographs of Natural Scenes." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 787–798. Doha, Qatar.

Kirillov, A., Wu, Y., He, K., and Girshick, R. 2020. "PointRend: Image Segmentation as Rendering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 9799–9808. Online.

Koo, T. K. and Li, M. Y. 2016. "A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research." *Journal of Chiropractic Medicine* 15 (2): 155–163.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. 2017. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations." *International Journal of Computer Vision (IJCV)* 123 (1): 32–73.

Kuznetsova, P., Ordonez, V., Berg, A., Berg, T., and Choi, Y. 2013. "Generalizing Image Captions for Image-Text Parallel Corpus." In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* 790–796. Sofia, Bulgaria.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. 1998. "Gradient Based Learning Applied to Document Recognition." *Proceedings of the IEEE* 86 (11): 2278–2324.

Lee, K.-H., Chen, X., Hua, G., Hu, H., and He, X. 2018. "Stacked Cross Attention for Image-Text Matching." In *European Conference on Computer Vision (ECCV),* 201–216. Munich, Germany.

Li, G., Duan, N., Fang, Y., Gong, M., and Jiang, D. 2020. "Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 34:11336–11344. Online.

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., and Chang, K.-W. 2019. "VisualBERT: A Simple and Performant Baseline for Vision and Language." *arXiv e-prints,* arXiv–1908.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., et al. 2020. "Oscar: Object-Semantics Aligned Pre-training for Vision-and-Language Tasks." In *European Conference on Computer Vision (ECCV),* 121–137. Online.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. 2014. "Microsoft COCO: Common objects in context." In *European Conference on Computer Vision (ECCV),* 740–755. Zurich, Switzerland.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv preprint arXiv:1907.11692.*

Lu, J., Batra, D., Parikh, D., and Lee, S. 2019. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks." In *Advances in Neural Information Processing Systems (NIPS),* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, 32:13–23. Vancouver, Canada.

Luong, T., Pham, H., and Manning, C. D. 2015. "Effective Approaches to Attention-based Neural Machine Translation." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 1412–1421. Lisbon, Portugal.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. 1993. "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* (Cambridge, MA, USA) 19 (2): 313–330.

Messina, N., Falchi, F., Esuli, A., and Amato, G. 2021. "Transformer Reasoning Network for Image-Text Matching and Retrieval." In *25th International Conference on Pattern Recognition (ICPR),* 5222–5229. Online.

Modi, A., Anikina, T., Ostermann, S., and Pinkal, M. 2016. "InScript: Narrative texts annotated with script information." In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC),* 3485–3493. Portorož, Slovenia.

Nicola, M., Giuseppe, A., Andrea, E., Fabrizio, F., Claudio, G., and Stéphane, M.-M. 2020. "Fine-grained Visual Textual Alignment for Cross-Modal Retrieval using Transformer Encoders." *arXiv preprint arXiv:2008.05231.*

Nivre, J., Marneffe, M.-C. de, Ginter, F., Hajivc, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., and Zeman, D. 2020. "Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection." *arXiv preprint arXiv:2004.10643.*

Ordonez, V., Kulkarni, G., and Berg, T. 2011. "Im2Text: Describing Images Using 1 Million Captioned Photographs." In *Advances in Neural Information Processing Systems (NIPS),* 24:1143–1151. Granada, Spain.

Patel, A., Sands, A., Callison-Burch, C., and Apidianaki, M. 2018. "Magnitude: A Fast, Efficient Universal Vector Embedding Utility Package." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 120. Brussels, Belgium.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. 2015. "Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* 2641–2649. Santiago, Chile.

Qi, D., Su, L., Song, J., Cui, E., Bharti, T., and Sacheti, A. 2020. "ImageBERT: Cross-modal Pre-training with Large-scale Weak-supervised Image-Text Data." *arXiv e-prints,* arXiv–2001.

Reimers, N. and Gurevych, I. 2019. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 3973–3983. Hong Kong, China.

Ren, S., He, K., Girshick, R., and Sun, J. 2016. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39 (6): 1137–1149.

Robinson, J., Rosenzweig, C., Moss, A. J., and Litman, L. 2019. "Tapped out or barely tapped? Recommendations for how to harness the vast and largely unused potential of the Mechanical Turk participant pool." *PloS One* 14 (12): 1–29.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1985. *Learning Internal Representations by Error Propagation.* Technical report. California University San Diego La Jolla Institute for Cognitive Science.

Schamoni, S., Hitschler, J., and Riezler, S. 2018. "A Dataset and Reranking Method for Multimodal MT of User-Generated Image Captions." In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas,* 140–153. Boston, MA, USA.

Sharma, P., Ding, N., Goodman, S., and Soricut, R. 2018. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,* 2556–2565. Melbourne, Australia.

Shi, B., Ji, L., Lu, P., Niu, Z., and Duan, N. 2019. "Knowledge Aware Semantic Concept Expansion for Image-Text Matching." In *International Joint Conference on Artificial Intelligence (IJCAI),* 1:2. Macao, China.

Simonyan, K. and Zisserman, A. 2015. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In *3rd International Conference on Learning Representations (ICLR).* San Diego, CA, USA.

Srinivasan, K., Raman, K., Chen, J., Bendersky, M., and Najork, M. 2021. "WIT: Wikipedia-based Image Text Dataset for Multimodal Multilingual Machine Learning." In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR).* Online.

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., and Dai, J. 2020. "VL-BERT: Pre-training of Generic Visual-Linguistic Representations." In *International Conference on Learning Representations (ICLR).* Online.

Suhr, A., Lewis, M., Yeh, J., and Artzi, Y. 2017. "A Corpus of Natural Language for Visual Reasoning." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* 217–223. Vancouver, Canada.

Suhr, A., Zhou, S., Zhang, A., Zhang, I., Bai, H., and Artzi, Y. 2018. "A Corpus for Reasoning about Natural Language Grounded in Photographs." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 6418–6428. Florence, Italy.

Sutskever, I., Vinyals, O., and Le, Q. V. 2014. "Sequence to Sequence Learning with Neural Networks." In *Advances in Neural Information Processing Systems (NIPS),* 3104–3112. Montreal, Quebec, Canada.

Tan, H. and Bansal, M. 2019. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 5103–5114. Hong Kong, China.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. 2016. "YFCC100M: The New Data in Multimedia Research." *Communications of the ACM* 59 (2): 64–73.

Tsikrika, T., Popescu, A., and Kludas, J. 2011. "Overview of the Wikipedia Image Retrieval Task at ImageCLEF 2011." In *Conference on Multilingual and Multimodal Information Access Evaluation (CLEF),* 4:5. Amsterdam, Netherlands.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. 2017. "Attention is All you Need." *Advances in Neural Information Processing Systems (NIPS)* (Long Beach, CA, USA) 30:5998–6008.

Wang, Y., Yang, H., Qian, X., Ma, L., Lu, J., Li, B., and Fan, X. 2019. "Position Focused Attention Network for Image-Text Matching." In *International Joint Conference on Artificial Intelligence (IJCAI),* 3792–3798. Macao, China.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. 2004. "Image Quality Assessment: From Error Visibility to Structural Similarity." *IEEE Transactions on Image Processing* 13 (4): 600–612.

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., et al. 2013. "OntoNotes Release 5.0." *Linguistic Data Consortium* (Philadelphia, PA, USA) 23.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., et al. 2020. "Transformers: State-of-the-Art Natural Language Processing." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 38–45. Online.

Xie, H., Mayer, R. E., Wang, F., and Zhou, Z. 2019. "Coordinating Visual and Auditory Cueing in Multimedia Learning." *Journal of Educational Psychology* 111 (2): 235–255.

Xie, N., Lai, F., Doran, D., and Kadav, A. 2019. "Visual Entailment: A Novel Task for Fine-grained Image Understanding." *arXiv preprint arXiv:1901.06706.*

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. 2019. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." *Advances in Neural Information Processing Systems (NIPS)* (Vancouver, Canada) 32:5753–5763.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. 2014. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions." *Transactions of the Association for Computational Linguistics (TACL)* (Baltimore, Maryland, USA) 2:67–78.

Yu, L., Poirson, P., Yang, S., Berg, A. C., and Berg, T. L. 2016. "Modeling Context in Referring Expressions." In *European Conference on Computer Vision (ECCV),* 69–85. Amsterdam, Netherlands: Springer.

Yu, Z., Li, J., Luo, T., and Yu, J. 2020. *A PyTorch Implementation of Bottom-Up-Attention.* https://github.com/MILVLG/bottom-up-attention.pytorch.

Zeiler, M. D. and Fergus, R. 2014. "Visualizing and Understanding Convolutional Networks." In *European Conference on Computer Vision (ECCV),* 818–833. Zurich, Switzerland.

Zellers, R., Bisk, Y., Farhadi, A., and Choi, Y. 2019. "From Recognition to Cognition: Visual Commonsense Reasoning." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6720–6731. Long Beach, CA, USA.

Zhang, Q., Lei, Z., Zhang, Z., and Li, S. Z. 2020. "Context-Aware Attention Network for Image-Text Retrieval." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 3536–3545. Online.

Appendices

A.1 UNITER Pre-Training

The training dataset of UNITER is a combination of four popular text-image datasets, namely COCO, Visual Genome, Conceptual Captions, and SBU Captions. For information about these datasets, see Section 4.1.

One of the main contributions of UNITER and its advantages to other multi-modal transformers are the sophisticated self-supervised pre-training tasks and the examination of their most profitable combination. These tasks are briefly summarized in the following.

- Masked-Language Modeling (MLM)
- Masked-Region Modeling (MRM)
  - Masked-Region Classification (MRC)
  - Masked-Region Modeling (MRM)
  - Masked-Region Classification with Kullback-Leibler Divergence (MRC-KL)
- Masked-Region Feature Regression (MRFR)
- Image-Text-Matching (ITM)
- Word-Region-Alignment (WRA) with Optimal-Transport (OT)

The ablation study showed that the best combination of pre-training tasks is: MLM + ITM + MRC-KL + MRFR + WRA

Masked-Language Modeling (MLM)

The idea of Masked Language Modeling was initially proposed in (Vaswani et al. 2017) and (Devlin et al. 2019). In this task, a random proportion (15%) of tokens are replaced by a so-called MASK token, and the model's goal is to predict the original token that was masked-out. Since this goal can be interpreted as a classification problem, where the possible classes are the tokens in the vocabulary, negative Log-Likelihood is used as the loss function.

Image-Text-Matching (ITM)

For this task, a special CLS token is introduced to tell the model about fused multi-modal input. In addition to the CLS token, a pair of textual tokens and visual regions (w, v) are input to the transformer. During training, positive pairs and negative pairs get sampled. Pairs are called positive if the textual and visual tokens come from the same sample and negative otherwise. The task is the binary classification problem of predicting whether w and v is a positive pair or not. Therefore the transformer output of the CLS token is fed through a FC-Layer with sigmoid activation function to produce values between $[0, 1]$, and binary cross-entropy is used as the loss function.

Word-Region-Alignment (WRA) with Optimal-Transport (OT)

This task aims to align textual tokens with visual regions, so that feature vectors of tokens are close to region feature vectors according to a distance metric if they share some semantic meaning. The authors of UNITER use Optimal Transport technique by interpreting the textual inputs w and the visual inputs v as discrete probability distributions and cosine similarity as cost-function. Optimal Transport now seeks the most efficient way to align the two distributions in terms of the cost function.

Masked-Region Modeling (MRM)

Like the MLM task, MRM masks some proportion of the visual input, i.e., the visual regions, with the special MASK token. In MRM, the visual MASK token is an all-zero matrix with the same size as the masked region. Since visual features are the high-dimensional, continuous output of an object detector network (in this case Faster R-CNN with ResNet-101) instead of the token's discrete labels, it is not possible to classify the masked regions like in the MLM task. The authors introduced three variants of MRM to overcome this issue.

Masked-Region Feature Regression (MRFR)

In this task, the model learns to regress the masked-out regions by finding the minimum of the L2-norm of the transformer's CLS output and the target region.

Masked Region Classification (MRC)

With MRC, UNITER learns to predict the image regions' discrete labels from the object detection network's output by minimizing the cross-entropy loss.

Masked Region Classification with Kullback-Leibler-Divergence (MRC-KL)

The problem with classic MRC is that the regions' predicted labels from the object detection network can be wrong because there are ground truth labels. To overcome this problem, in MRC-KL, KLD is used to compute the loss between the continuous probability density function over all classes of the object detector and the transformer's output for the MASK token.

A.2 MMIRS – FSS Software Architecture

A.2.1 Design Goals

To support different datasets and retriever models in MMIRS (see Section 9.3) and to achieve "real-time" retrieval latencies, the main design goals of the *FSS* architecture are modularity, extensibility, and computational efficiency. These goals are realized by using software design patterns like Singletons, Facades, and Factories (Gamma et al. 1995), and by applying as many functions as possible in parallel. In the current version of MMIRS, only TERAN models are supported out of the box because extracting WRA matrices from UNITER is not straightforward and out of the scope of this thesis. In the future work Section 10.6.3, approaches to do so are proposed.

Due to the modular software architecture of the FSS, adding TERAN models, trained on a specific dataset, only requires specifying the model's location in the corresponding section in the configuration file. Adding datasets to MMIRS also only needs to be specified by its name, location, the type of model that can access the data, and some additional metadata in the configuration file. When booting MMIRS, the system parses the respective sections in the configuration file, automatically instantiates the specified retrieval models and datasets, and makes them available for the users. In the following, technical details of this functionality get elucidated.

A.2.2 Details

The primary challenge while designing MMIRS was to support state-of-the-art models like TERAN or UNITER, which are not supported yet by any framework as of the date of this thesis. Hence, the codebase of the models had to be extended to leverage them for custom text-image retrieval at inference time. Therefore, the original GitHub repositories[2,3] were forked[4,5] to implement customized access to the models. Still, it is not entirely possible to provide a universal interface for every type of model. This is because each model requires image features in a different format and processes them differently. To support different retrieval models on different datasets anyway, MMIRS is designed to wrap the model's retrieval functionality and the access to the image features of the datasets. Detailed UML class diagrams of MMIRS' *Fineselection Stage* are depicted in Figure A.1 and Figure A.2 and are explained briefly in the following. Note that the diagrams only show the most important methods, parameters, and member variables to keep the overview, and that UNITER is not (fully) supported in the current version of MMIRS.

The class diagram in Figure A.1 shows the software design to support different datasets. The central interface is the *ImageFeaturePool (IFP)*, which represents the complete pool of image features in a particular dataset for a particular retriever model. This interface has an abstract method to load the data, i.e., the features contained in the *IFP*, into the hosting server's main memory. The second abstract method is required to create an *ImageSearchSpace (ISS)*.

An *ISS* is an abstract subset of an *IFP* for a specific *RetrieverType* and is always stored in memory. The abstract method of this class is used by a *Retriever* model to directly process or forward the image features through the transformer stacks. As already mentioned, MMIRS supports TERAN and UNITER models out of the box, i.e., the respective *IFP* and *ISS* interfaces are already implemented. Instead of creating *IFP* instances manually, the *ImageFeaturePoolFactory* singleton parses the respective section in the configuration file and instantiates the specified *IFP*s automatically. An example *IFP* specification for COCO dataset and TERAN retriever is shown in Listing 1.

2. https://github.com/mesnico/TERAN
3. https://github.com/ChenRocks/UNITER
4. https://github.com/floschne/TERAN
5. https://github.com/floschne/UNITER

```yaml
fine_selection:
  feature_pools:
    coco:
      teran:
        data_root: /raid/datasets/coco/pre_computed_embeddings
        fn_prefix: COCO_000000
        num_workers: 32
        pre_fetch: False

    wismir_v2:
      teran:
        data_root: /raid/datasets/wicsmmir/v2/pre_computed_embeddings
        fn_prefix: ''
        num_workers: 32
        pre_fetch: False

    f30k:
      teran:
        data_root: /raid/datasets/f30k/pre_computed_embeddings
        fn_prefix: ''
        num_workers: 32
        pre_fetch: False
```

Listing 1: Example snippet of the configuration section which is parsed by the *ImageFeaturePoolFactory* to instantiate the specified *ImageFeaturePool*s (see Figure A.1). By parsing this specifications, *TeranFeaturePools* for COCO, WISMIR and Flickr30k would get instantiated. The most important parameters are the ”data_roots” in lines 5, 11, and 17. These tell the factory where the files containing the features are located. The other parameters specify what file prefix the features have, how many workers are used to load the files from disk into memory in parallel, and whether the respective *ImageFeaturePool* should be pre-fetched or loaded into RAM when booting MMIRS.

Figure A.2 shows the architecture to support different *Retriever* models. The *Retriever* interface plays the central role and is the universal interface to retrieve the top-k matching images according to a focus and context, i.e., a user's query. In addition to the query parameters, an *ImageSearchSpace* is required, having the image features, which get forwarded through the model, ready in memory. *TeranRetriever* and *UniterRetriever* implement the *Retriever* interface to support TERAN and UNITER model types. The *RetrieverFactory* singleton instantiates the specified retrievers by parsing the corresponding sections in the configuration file. Example *TeranRetriever* and *UniterRetriever* specifications are presented in Listing 2.

```yaml
fine_selection:
  retrievers:
    uniter_base:
      retriever_type: uniter
      device: cuda
      model: downloads/pretrained/uniter-base.pt
      model_config: config/uniter-base.json

    teran_coco:
      retriever_type: teran
      device: cuda
      model: pretrained_models/coco_MrSw.pth.tar
      model_config: configs/teran_coco_MrSw_IR_PreComp_API.yaml

    teran_wicsmmir:
      retriever_type: teran
      device: cuda
      model: pretrained_models/wicsmmir_v2_MrSw.pth.tar
      model_config: configs/teran_wicsmmir_v2_MrSw_IR.yaml

    teran_f30k:
      retriever_type: teran
      device: cuda
      model: pretrained_models/f30k_MrSw.pth.tar
      model_config: configs/teran_f30k_MrSw_IR.yaml
```

Listing 2: Example snippet of the configuration section which is parsed by the *RetrieverFactory* to instantiate the specified *Retrievers*s (see Figure A.2). By parsing this specifications, three *TeranRetrievers* and one *UniterRetriever* would get instantiated. The most important parameters are the ”model” and ”model_config”. These tell the factory where the pretrained model files and the corresponding configurations for the model are located. The parameter ”device” specifies on which device, i.e., GPU or CPU, the model is executed, and the parameter ”retriever_type” tells the factory whether to instantiate a *TeranRetriever* or *UniterRetriever*.

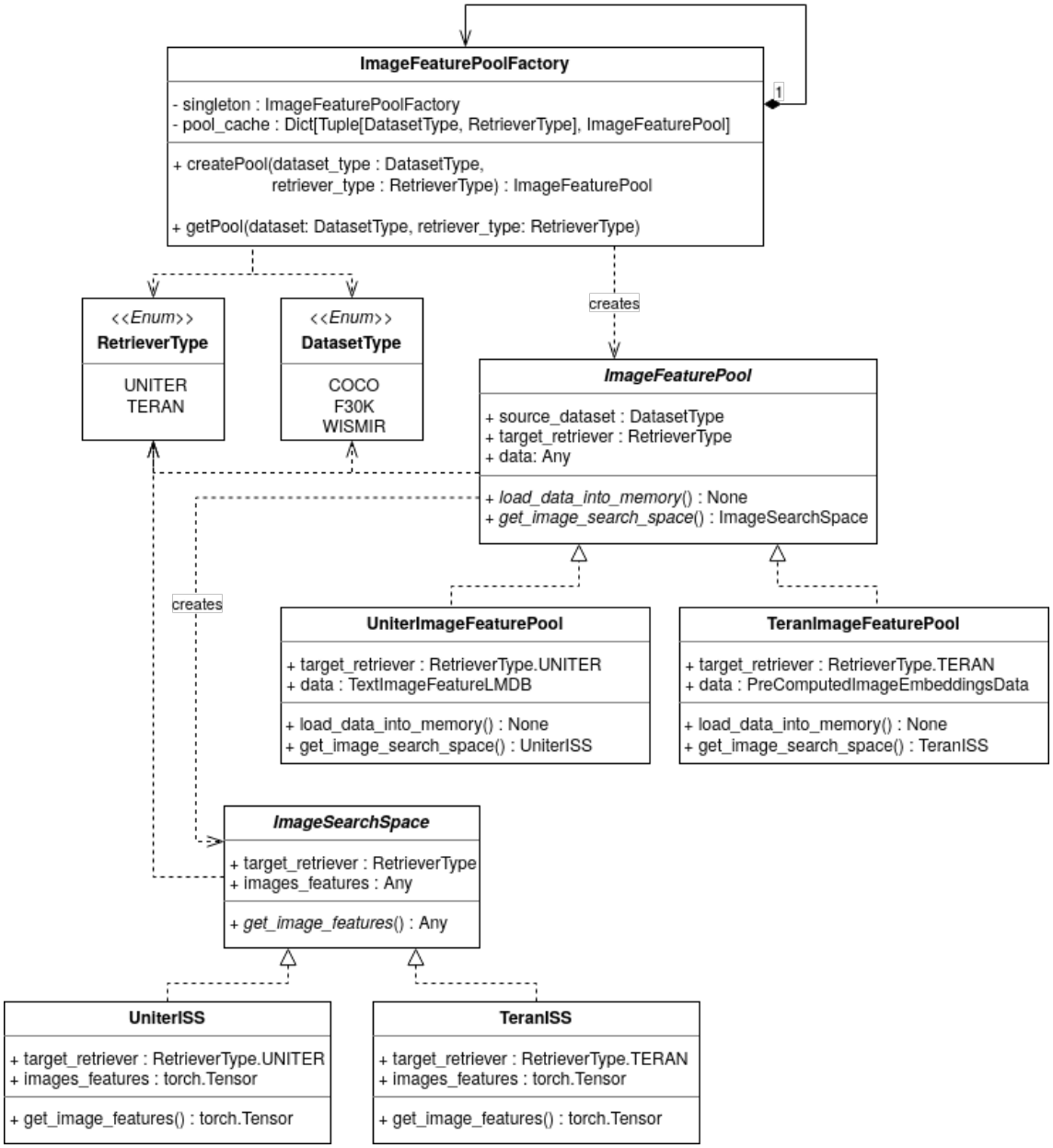


Figure A.1: UML class diagram that shows part the software architecture of the *Fineselection Stage* designed to support different dataset for different retriever models. Note that only the most important methods, parameters, and member variables are drawn to keep the overview.

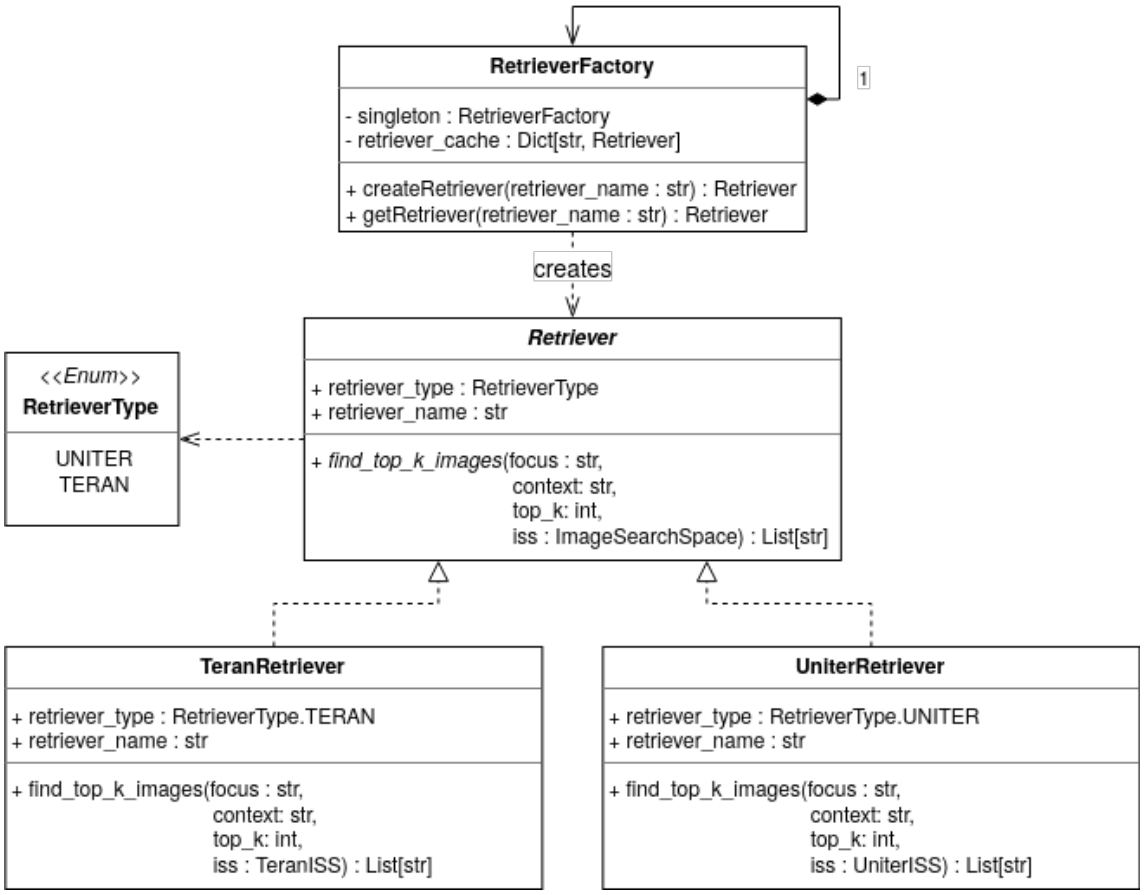


Figure A.2: UML class diagram that shows part the software architecture of the *Fines-election Stage* designed to support different different retriever models. Note that only the most important methods, parameters, and member variables are drawn to keep the overview.

A.3 IRST – Software Architecture

A.3.1 Central Technical Concept

IRST allows image rankings predicted by a text-image retrieval model to be evaluated by human raters using three different tasks. The three primary classes, representing image rankings, tasks, and the tasks' results, are briefly introduced next. For more details, see Section A.3.3 and Figure A.6.

A single image ranking, predicted by a model, consists of a text and a ranked list of images and is represented by an instance of the `MRanking` class. `MRanking` instances are generated from a DataFrame, where the raw image rankings, i.e., a model's output, are persisted. From `MRanking`s, `BaseSample` instances are generated, which represent tasks to be evaluated by human raters. Submitted evaluation results are stored and represented by `BaseResult`. This data flow is depicted in Figure A.3.

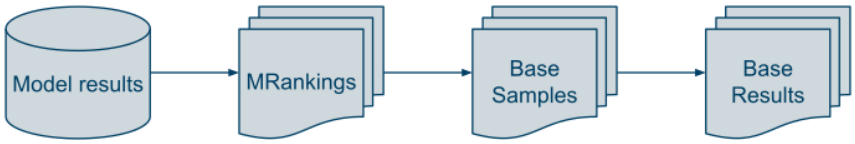


Figure A.3: Relationship of the main classes in the IRST application. Image rankings predicted by a text-image retrieval model are stored in the rows of a DataFrame. `MRanking` instances represent these predicted image rankings. From `MRanking` instances, `BaseSample` instances are generated, which describe a task to evaluate an image ranking. `BaseResult` instances represent the evaluation results submitted by raters.

Since IRST provides three different study methods (see Section 7.1), there are three different subclasses of `BaseSample`, which represent the tasks of the respective study method: `RankingSample`, `RatingSample`, or `LikertSample`. The results of the respective tasks are represented by the three different subclasses of `BaseResults`: `RankingResult`, `RatingResult`, or `LikertResults`.

A single image ranking consists of a text and a ranked list of images, i.e., their IDs, represented by an instance of the `MRanking` class. From the `MRanking`s, `BaseSample` instances are generated and evaluated by the raters. The submitted results of a `BaseSample` is contained in a `BaseResult`. This data flow is depicted in Figure A.3.

There are three different types of `BaseSample`: `RankingSample`, `RatingSample`, or `LikertSample`, and hold information for a single evaluation task (see Section 7.2). An evaluation result of a rater is represented by an instance of `BaseResult`, i.e., `RankingResult`, `RatingResult`, or `LikertResults`, respectively.

A.3.2 Overview

IRST is a client-server application and consists of a python backend, a NuxtJs[6] user interface, and an admin UI based on SwaggerUI[7]. It is also highly customizable while still easy to set up and deploy with or without Docker[8].

The application consists of four major components: the frontend or user interface, the backend, a Redis-server, and a static-image server. Typically, each component runs in a Docker container, and all containers are orchestrated via docker-compose (see Figure A.4). Note that while it is possible to deploy the application without Docker, it is not recommended.

6. https://nuxtjs.org/
7. https://swagger.io/tools/swagger-ui/
8. https://www.docker.com/

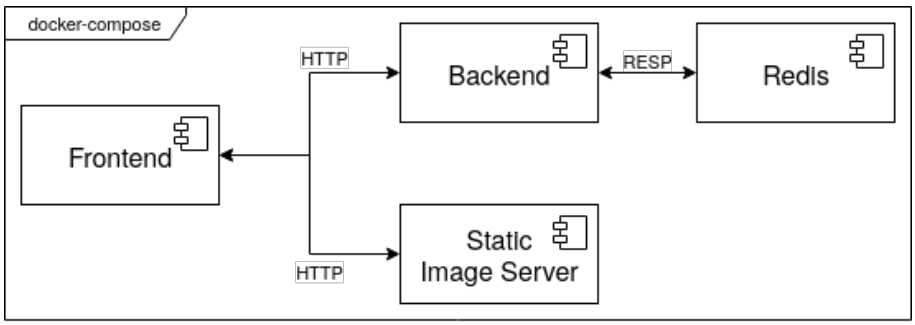


Figure A.4: High-level components or Docker containers of the IRST application, when deploying with docker-compose. Arrows are indicating the communication between the components.

The software design is roughly oriented to a layered architecture. A schematic overview of the different layers is depicted in Figure A.5. The data model contains the data or domain

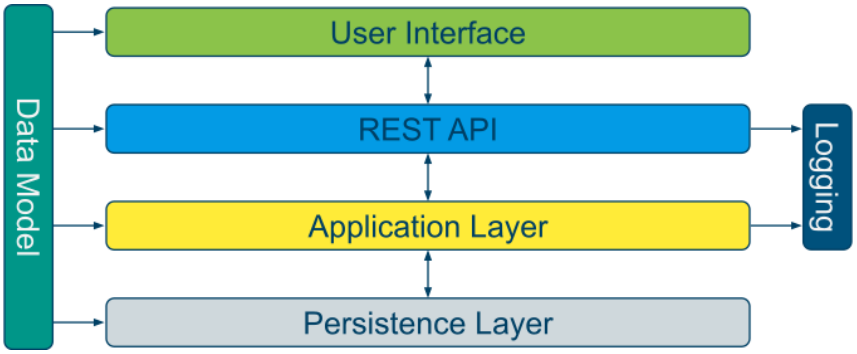


Figure A.5: Schematic overview of the different layers in the architecture of IRST.

classes, which are used throughout the whole application. Communication between the user interface or frontend (see Section A.3.4) and the backend happens via HTTP through REST API endpoints. The REST API layer (see Section A.3.5), which provides several endpoints to execute certain actions, is part of the backend and communicates the requests from the frontend to the Application Layer (see Section A.3.6). In this layer, the application's logic is contained and realized by several services responsible for particular actions. The lowest layer, the persistence layer, is responsible for serializing and deserializing data class instances to and from disk and is realized with Redis[9]. Redis was chosen as the database because, in IRST, there are no complex relationships between the domain classes. Further, Redis provides convenient JSON support, which is especially useful with the data classes that are based on the pydantic framework.

A.3.3 Data Model

A UML class diagram that represents the data or domain model of IRST is shown in Figure A.6. In this section, only a brief introduction is given because the classes can be described best in the following sections where their usage is explained.

The super-class of all domain classes is the pydantic[10] `BaseModel`. This class provides data validation, JSON serialization, and deserialization functionality out of the box. There-

9. https://redis.io/
10. https://pydantic-docs.helpmanual.io/

fore, it is commonly used in combination with FastAPI [11] and helpful for Redis-based[12] persistence (see Section A.3.5 and Section A.3.6).

The most important classes are `MRanking`, `BaseSample`, `BaseResult`, and their respective sub-classes. For an explanation, how the classes are related to each other, see Section A.3.1.

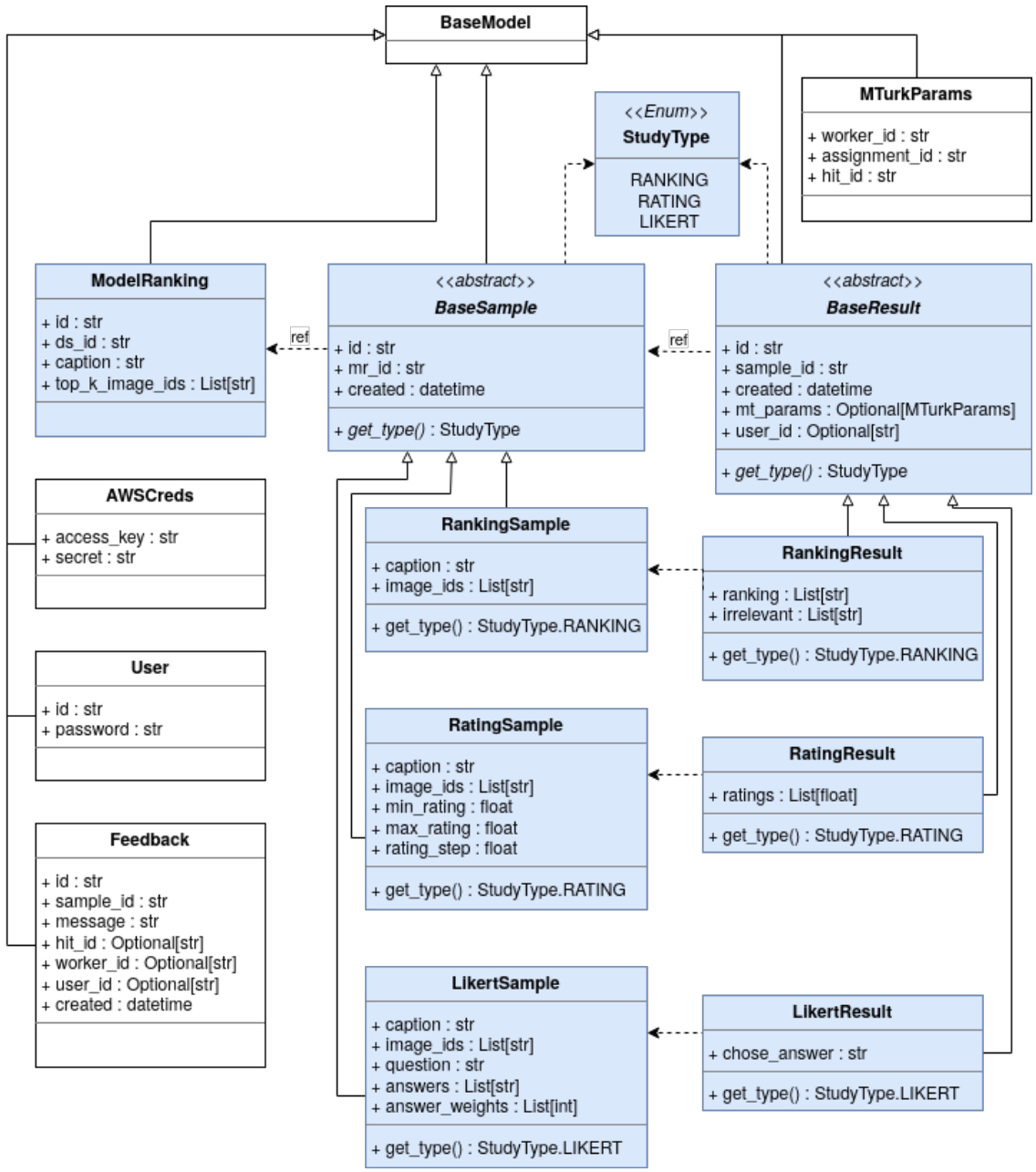


Figure A.6: UML class diagram that represents the data or domain model of IRST.

A.3.4 User Interface

The user interface of IRST is a NuxtJS single-page web application. In MTurk mode, it consists only of a single page, where a worker evaluates the provided sample. Examples for `RankingResults`, `RatingResults`, and `LikertResults` are depicted in Figures 7.1, 7.2, and 7.3, respectively.

11. https://fastapi.tiangolo.com/
12. https://redis.io/

The IRST UI also provides routes or pages for users to register, login, and logout in Standalone mode. A Vuex[13] store is used to share state information about a user between the required components.

In addition to the Vue[14] components that are the building blocks of the pages, several services that handle communication with the backend are implemented in plain JavaScript with axios[15]. To hide the communication with the backend from the clients and to avoid CORS issues, a reverse-proxy middleware is employed.

A.3.5 REST API

The API layer provides multiple endpoints to execute different methods in the application layer, realized with FastAPI[16] and pydantic[17] frameworks. Typically it accepts and returns instances of classes from the data model shown in Figure A.6, which are automatically marshaled in JSON format.

In total, there are currently 12 top-level endpoints, which are responsible for 54 different dedicated actions. These endpoints are shown in Figure A.7, and their scope of functions is briefly summarized in Table A.1. Most of the API resources are protected against unauthorized requests and require an HTTP Bearer authentication header with a valid JWT of an admin user sent with every request. This layer of security is crucial when the application is available from the public internet.

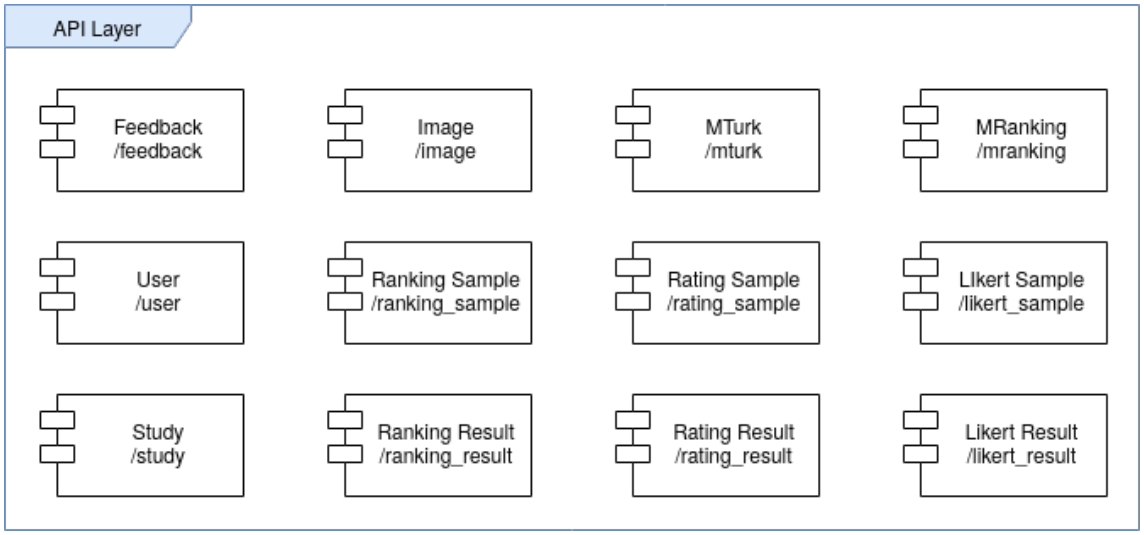


Figure A.7: Overview of the components, i.e., FastAPI router instances, in the API Layer of IRST.

A.3.6 Application Layer

This layer holds the business logic of IRST and consists of several singleton service classes responsible for different functionalities. The different services are shown in Figure A.8 and described in the following.

13. https://vuex.vuejs.org/
14. https://vuejs.org/
15. https://github.com/axios/axios
16. https://fastapi.tiangolo.com/
17. https://pydantic-docs.helpmanual.io/

Route	Purpose
/feedback	Collecting Feedback submissions and listing of all Feedback or specific Feedback.
/image	Converting image ids to URLs pointing to the static image server and vice versa.
/user	User registration and login.
/study	Returns progress and state of studies in standalone mode.
/mturk	Most complex endpoint with the highest number of functions. Responsible for: publishing, deleting, or listing of HITs; listing and approving Assignments; managing Qualifications; sending messages to Workers.
/mranking	Listing of all MRankings or loading of single MRankings.
/ranking_sample	Listing of all RankingSamples or loading of single RankingSamples.
/ranking_result	Collecting RankingResult submissions and listing of all RankingResults or a specific RankingResult.
/rating_sample	Listing of all RatingSamples or loading of single RatingSamples.
/rating_result	Collecting RatingResult submissions and listing of all RatingResults or a specific RatingResult.
/likert_sample	Listing of all LikertSamples or loading of single LikertSamples.
/likert_result	Collecting LikertResult submissions and listing of all LikertResults or a specific LikertResult.

Table A.1: Scope of functions summary of the top-level endpoints of the IRST REST API.

Study Services

The `RankingStudyService`, `RatingStudyService`, and `LikertStudyService` manage and coordinate studies of the respective type. That is, the services are responsible for creating the respective samples, i.e., `RankingSample`, `RatingSample`, or `LikertSample`, from the `MRanking`s, handing out the samples, and, finally, accepting the submitted `BaseResult` subclass (see Figure A.6).

In MTurk mode, the services need to provide the sample related to the respective HIT a worker has accepted. When the worker submits her result, the service has to store and reference it with the associated sample. The MTurk Marketplace completely handles the distribution of HITs.

In standalone mode, the distribution of tasks has to be managed by the services. To ensure that all samples of a specific study are evaluated, no sample is evaluated more often than specified, and no user receives the same sample twice, the following concept was developed.

For each study, the number of iterations is specified in the configuration file. One study iteration comprises the evaluation of every sample or task. Therefore, the services manage three lists, namely “todo”, “in progress”, and “done”, representing the state of a sample similar to a Kaban board. When initializing a study iteration, all samples are added to the “todo” list. If a user requests a sample, the service selects a sample the user has not evaluated in previous iterations and moves it from “todo” to “in progress” by an atomic action. A sample can be on the “in progress” list only for a specified amount of time. If a user does not submit her result in the specified time, the respective sample expires and is atomically moved from the “in progress” list back to the “todo” list. If a user submits her result on time, the respective sample is moved from “in progress” to “done” list also in an atomic fashion. This process is illustrated schematically in Figure A.9.

If all samples are on the “done” list, i.e., the other lists are empty, a new study iteration gets initialized until the maximum number of iterations has been reached.

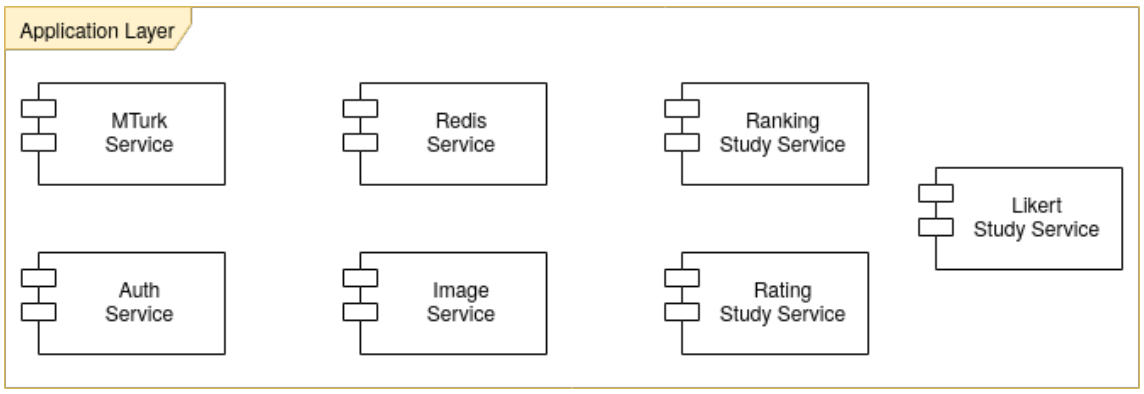


Figure A.8: Overview of the different services in the IRST application layer.

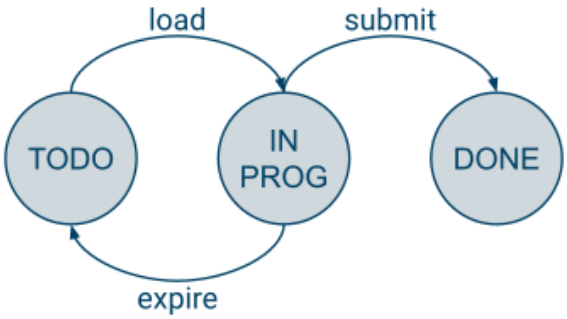


Figure A.9: Schematic overview of the different states, i.e., lists, in which instances of `BaseSample`. Transition arrows are indicating the actions of the study service with the respective `BaseSample` instance as parameter.

When all samples of all iterations of a study are evaluated, or a user cannot receive more samples because she already worked on every sample once, the services will throw an exception, which is reflected in the return value of the respective API endpoint.

Auth Service

Since IRST is thought to be available from the public internet, protecting the API endpoints from unauthorized requests is crucial. Especially in MTurk mode, it is mandatory that the application is reachable from all over the world. Therefore, almost all API endpoints require user authentication, and the administrative functions are only accessible by admin users. The Auth service handles the authentication and authorization logic, realized by combining JSON Web Tokens and PBKDF2. The registration, authentication, and authorization processes are schematically sketched in Figure A.10.

MTurk Service

As the name suggests, the service is responsible for all functions related to MTurk. The functions are accessible via the SwaggerUI of the REST API and require admin user authorization. To execute the methods of the MTurk API, the AWS SDK boto3[18] is used. Executing any method MTurk operation requires to provide AWS credentials (see Figure A.6) to associate the respective operation with the specified AWS account. These credentials can either be transmitted with every request or can be fixed in the configuration file. The service's primary purpose is to create and publish HITs based on `RankingSample`, `RatingSample`, or `LikertSample` in the MTurk Marketplace. Therefore the service generates `ExternalQuestion` HITs from the respective samples and section in the configuration

18. https://aws.amazon.com/de/sdk-for-python/

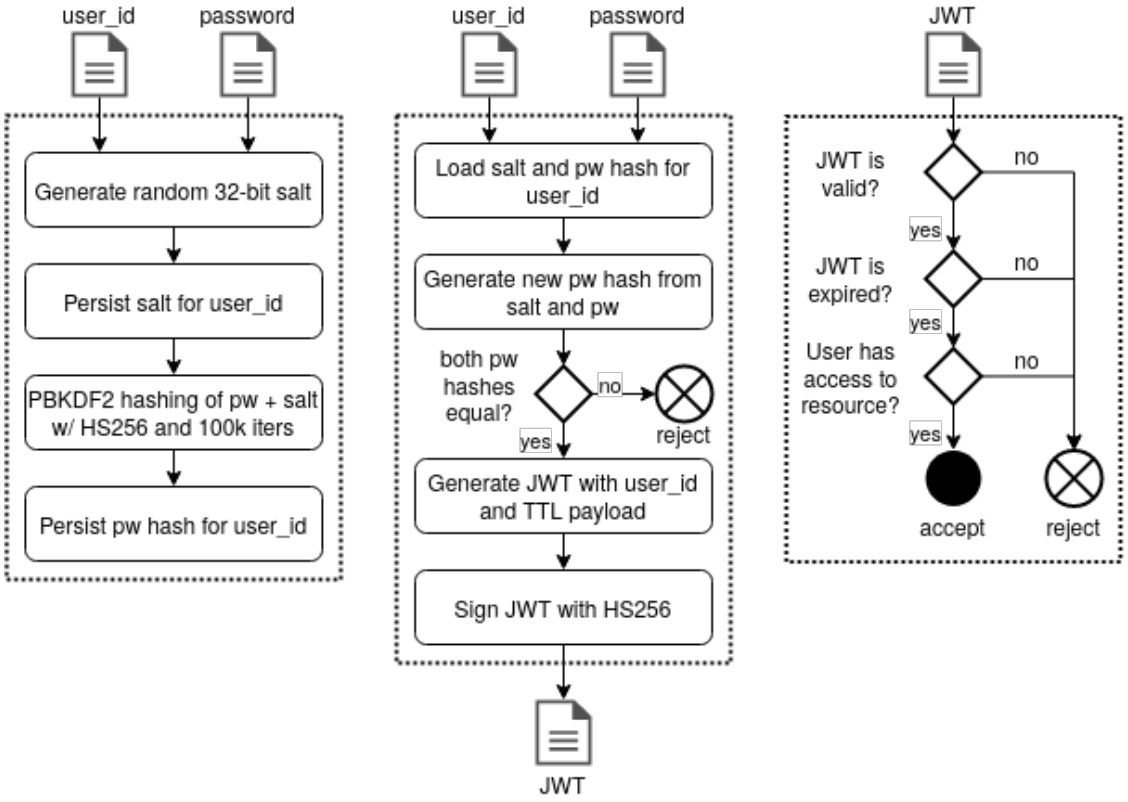


Figure A.10: Schematic overview of the registration process, the authentication process, and the authorization process (from left to right) of the Auth service in IRST.

```yaml
mturk:
  rating:
    hit_auto_approval_delay_in_seconds: 604800 # 1 week's seconds
    hit_assignment_duration_in_seconds: 300 # 5 min
    hit_reward: 0.15 # in dollars
    hit_title: Example HIT Title
    hit_description: Example HIT Description
    hit_keywords: some, example, hit, keywords
    hit_max_assignments: 3 # the number of times this HIT gets evaluated
    hit_lifetime: 604800 # 604800 = 1 week's seconds
    hit_custom_qualifications:
      3KHNGLCXQI6RANDOMQUALIFICATIONID:
        comparator: 'EqualTo'
        integerValues: [99]
        actionsGuarded: 'Accept'
    worker_quali_req_min_hits_approved: 1000 # min number of approved HITs
    worker_quali_req_min_percent_approved: 90 # assignment approval rate
    external_question_base_url: https://url.to.irst.app/rating
```

Listing 3: Sample configuration for HITs generated from `RatingSample`.

file. A sample configuration for HITs generated from `RatingSample` is shown in Listing 3. In addition to the HITs metadata, pre-defined and custom Qualification requirements can be configured, which are also parsed and automatically incorporated by the service. To save bandwidth because typically a large number of HITs are created, the service first

creates a HITType and reuses it when the HITs themselves are created.

Another essential function of the service is to associate or disassociate Qualifications with Workers. This can be done conveniently by providing a list of Worker IDs.

Additionally, the service can approve Assignments, send messages to workers, or get the balance of the respective AWS account.

Redis Service

This service is the connection to the persistence layer and is responsible for storing and loading instances of data classes (see Figure A.6). Since all data classes extend the pydantic `BaseModel`, which supports serializing and deserializing objects in JSON format out of the box, the service stores the id and the JSON serialization string of objects as key-value pairs. Further, the service stores the state of the study services and information about created MTurk HITs for caching reasons.

For the communication with the Redis server instance, the python redis-client[19] library is used.

Image Service

As hinted in Figure A.6, the images contained in an `MRanking` or any sample are only IDs of images. By default, a Lighttpd[20] instance is used as a static image server to host and display the images. The Image Service converts the respective image IDs to URLs, pointing to the image server as configured in the settings file.

One issue that IRST solves is the long page loading duration in the Client's UI, which is due to the images necessary to solve a task. To drastically save bandwidth and therefore reduce the page loading duration of the clients, the service offers functionality to generate thumbnails of the images and convert the images to WebP format. For example, the average size of all Flickr30k images is 139149 Bytes, the average size of all thumbnails generated from those images in WebP format is 15975 Bytes. In this case, using thumbnails in WebP format would save 88.5% of the bandwidth.

19. https://developer.redislabs.com/
20. https://www.lighttpd.net/

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Vorname Nachname

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum	Vorname Nachname