universität wien

# MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Image Analysis for Extracting Facts from Images“

verfasst von / submitted by

Adrian Hofer, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

| | |
|---|---|
| Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet: | UA 066 935 |
| Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet: | Masterstudium Medieninformatik |
| Betreut von / Supervisor: | Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Klas |

# Acknowledgements

First of all, I want to thank my supervisor Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Klas for his supportive mentorship. He always provided valuable feedback during long and extended meetings, even during hectic times.

I am grateful to Dipl.-Ing. Dr. Peter Kalchgruber for inventing the FactCheck framework. His ideas laid the basis for this thesis.

I greatly appreciate the participation of all fellow students in the FactCheck workshops, which was helpful for understanding the bigger picture.

I want to thank Isabella Zborka BA BA MA for the second reading of the thesis, and Alexandra Wiesner MA, Pascal Gfäller BA BSc, and Sebastian Didusch MSc for their endurance in discussions and for helping me with different perspectives.

Last but not least, I want to thank my entire family for their support and encouragement throughout the process of writing this thesis.

# Abstract

False or misleading information on the web calls for specific approaches that allow to check facts that are conflicting. Fact-checking, as considered in this thesis, means the detection of conflicting or inconsistent data on the web. The role of images in fact-checking is central because images are a fundamental part of information on the web. However, it is still not known what the best way to extract facts from images is. Scene graphs, combined with entity linking, are a potential candidate technique to extract facts from images. The presented pipeline to extract facts from images is designed under the assumption that the system should be used on a large scale and can be used within the FactCheck framework. Considering these prerequisites, an image similarity measurement is analyzed to select images that can be matched. There are two prototypical implementations for the fact extraction pipeline. One implementation in Jupyter Notebooks showcases the execution of the pipeline and builds a potential environment for students in this field to work on. A second implementation showcases the use of the pipeline in a cloud environment. The quality of facts is determined by a learned link prediction model on Wikidata, which scores the triplets of the scene graph that have links to entities on Wikidata. The score of object detection and scene graph generation directly impacts the quality of triplets. The quality of produced triplets is under the best-scoring triplets in the model. This thesis provides a feasible approach for extracting facts, which are the property-value pairs in a resource description framework triplet, from images.

# Kurzfassung

Falsche oder irreführende Informationen im Internet erfordern spezifische Ansätze, die es ermöglichen, widersprüchliche Fakten zu überprüfen. Fact-Checking, wie es in dieser Arbeit betrachtet wird, bedeutet die Erkennung von widersprüchlichen oder inkonsistenten Daten im Web. Die Rolle von Bildern bei der Faktenüberprüfung ist von zentraler Bedeutung, da Bilder ein grundlegender Bestandteil von Informationen im Web sind. Es ist jedoch immer noch nicht bekannt, wie man am besten Fakten aus Bildern extrahiert. Szenegraphen, kombiniert mit Entity Linking, sind ein möglicher Kandidat, um Fakten aus Bildern zu extrahieren. Die hier vorgestellte Pipeline zur Extraktion von Fakten aus Bildern wurde unter der Annahme entwickelt, dass das System in großem Maßstab eingesetzt werden soll und im Rahmen von FactCheck verwendet werden kann. Unter Berücksichtigung dieser Voraussetzungen wird eine Bildähnlichkeitsmessung analysiert, um Bilder auszuwählen, die abgeglichen werden können. Es gibt zwei prototypische Implementierungen für die Faktenextraktionspipeline. Eine Implementierung in Jupyter Notebooks zeigt die Ausführung der Pipeline und bildet eine mögliche Arbeitsumgebung für Studenten in diesem Bereich. Eine zweite Implementierung zeigt die Verwendung der Pipeline in einer Cloud-Umgebung. Die Qualität der Fakten wird durch ein erlerntes Link-Vorhersagemodell auf Wikidata bestimmt, das die Triplets des Szenegraphen bewertet, die Links zu Entitäten auf Wikidata haben. Die Bewertung der Objekterkennung und der Erzeugung des Szenegraphen wirkt sich direkt auf die Qualität der Triplets aus. Die Qualität der erzeugten Triplets liegt unter den am besten bewerteten Triplets des Modells. Diese Arbeit bietet einen praktikablen Ansatz für die Extraktion von Fakten aus Bildern. Fakten sind dabei die Eigenschaft-Werte-Paare in einem Ressourcenbeschreibungs-Framework-Triplet.

# Contents

# List of Tables

# List of Figures

# Listings

# 1. Introduction

## 1.1. Motivation

Detecting facts in the news and on the web has grown large interest in today's society. Fake news can be a severe threat to democracy [1]. E.g., in 2017, photos of Donald Trump's inauguration speech were edited to make the crowd appear larger[1,2]. Such events show the urge for resolving conflicting data in images on the web.

In 2018, Kalchgruber [2] presented a framework for detecting and resolving conflicting data on the web [2]. This framework uses the structured data available on a website (e.g., data on a website available in JavaScript Object Notation for Linked Data (JSON-LD) format) to compare facts about websites. In their paper, the authors already mention that facts about images could also be compared.

This thesis aims to investigate how to extract facts from images. Facts are the property-value pairs in structured data [2], if the structured data is in a Resource Description Framework (RDF). RDF data is structured into subject, predicate, and object $\langle S, P, O \rangle$. For example, if a user opens the Wikipedia page of the movie *Flubber*, there is structured data available. The user can see this structured data in the info box of the Wikipedia page. The value for the structured data field "published date" is a fact about the movie *Flubber*. In RDF, the triple could look like this: $\langle Flubber, published, 1997 \rangle$. But what are the facts about images?

If we look at a real-world image, typically, we see objects interacting with each other. For example, we could see a woman walking with her dog or a fruit lying on a table. If we express this circumstance in RDF, the triplets could look like $\langle woman, walking, dog \rangle$ and $\langle fruit, on\ top\ of, table \rangle$. We could see such a triplet as fact about an image, but it will be referred to as structured data about an image in this section. Describing the content of an image as a triplet would make it possible for a machine to understand the content of an image. Suppose a machine understands the content of an image. In that case, the machine can analyze the content of an image, compare similar images (based on the content or technical features of the image) and find similar images based on their content. There are already numerous methods to generate triplets about the content of an image. These methods will be presented in this thesis and it must be discussed if there are other possible methods. It is still unknown how well such methods work if the structured data is used as facts. There is still uncertainty about how to analyze the content of an image as facts, as shown in Section 3.1 and 3.2.

[1] https://www.theguardian.com/world/2018/sep/06/donald-trump-inauguration-crowd-size-photos-edited

[2] https://edition.cnn.com/2018/09/07/politics/trump-inauguration-photos/index.html

Having obtained structured data from the image, it would be possible to link this data to a knowledge base. If the structured data of an image is linked to a knowledge base, it could be possible that the structured data becomes facts about an image. But how to link the detected objects to named entities in a knowledge base? It would be interesting to not only have structured data about the content of an image, but also gather facts about the visible and maybe invisible objects depicted in the image. Such knowledge is saved in knowledge bases, e.g., Wikipedia. The dog in the image of a woman walking with a dog could be linked to the Wikipedia page of dogs (https://en.wikipedia.org/wiki/Dog), where a fact about a dog could be the dog's species C. familiaris $\langle Dog, species, C.\ familiaris \rangle$. Other or multiple knowledge bases are possible and imaginable as well. If structured data of an image is linked with knowledge bases, it could be possible to analyze the generated facts about an image by quality standards.

A primary concern is which information is initially contained in an image that could be used as facts. For example, each image is initially shipped with meta information like the file format. This information could be used as facts and stored in an appropriate format.

Gathering and comparing facts about images is a major area of interest in the field of fact-checking, as will be shown in Section 3.1 and Section 3.2. Images are used everywhere on the internet, and it is of major interest to be able to compare if an image presented on a news page corresponds with the textual content of the news page and if the overall information in the image corresponds with other information available on the internet.

A key aspect of this thesis is to be able to discuss the integration of an image fact-analyzing system on a large scale. To achieve this, it is necessary to understand if an image has already been processed. If there are similar images in the database to which the image could be compared to, it would be possible to resolve conflicting data about the image. For example, suppose there is an image of a skyscraper with 100 windows and a second image of the same skyscraper from the same direction with only 50 windows. In that case, the system should be capable of knowing if the two skyscrapers are the same. Having knowledge about similar images in a database would not only enable the retrieval and comparison of facts about images. It would benefit the performance of the whole pipeline.

## 1.2. Research Question

This thesis examines the design and implementation of a pipeline for transforming an image into structured data containing links to knowledge graphs for usage in the FactCheck environment. This question can be split up into five sub-questions:

1. How can images be classified into originals, duplicates, and replicates? This step is necessary to prevent the pipeline from being executed for each request. When the answer to the question posed by the request is already known, it is unnecessary to execute the whole pipeline again. Furthermore, knowing if an image is original or replicated is essential for fact-checking. If a user seeks to gather facts about an image, it is crucial to know whether it is original or replicated. If the image is a replicate, the user should be capable of gathering facts about the image's original.

2. How can different objects in an image be detected? It is not enough to only detect the presence of an object in the image. It is necessary to detect the object's spatial position in the image.

3. How do these detected objects interact, and how can their interaction be detected? To be capable of making statements about the image's content, it is necessary to obtain a predicate. This predicate is the interaction. To make the statement processable by a machine, the statement needs to be in the form of a subject, predicate, and object.

4. How can the interaction and the object be linked to a knowledge graph? Objects or interactions in an image can be generalized to some higher concept. To gain knowledge about an image and make this knowledge machine-processable, the machine-processable statements about the image must be linked to a knowledge graph.

5. What should the format of the structured data look like? Information about the image is indispensable. Which information about the image could enrich the result?

This thesis focuses on images of humans. This limitation is selected because of the vast number of images available on the web and the novelty of the task of extracting knowledge from images. Hence, limiting the available data and restricting the approach to a subset of the available images seems necessary. For every other instance of an object, it is not in the scope of this thesis to define which instance of that object's general concept the object is showing. The general concept of the object is sufficient, except for humans. For humans, it is in the interest of this paper to propose an approach to enrich the knowledge about an image with the instance information about an object which is a human.

Another limitation is the retrieval of images. There will not be a bot to crawl the web for images automatically. The images will be the sole input of the framework, and the output will be facts about that image in a structured format. Considering the IdaFix implementation of FactCheck, the image will be obtained by the browser plugin and sent to the implementation of the framework for extracting facts from images.

## 1.3. Research Approach

The research approach is structured into five main phases. The first step in this process is to research the current state of the art of extracting facts from images. The techniques and tools found are compared with each other to identify potential candidates for a prototype in the second step. In a third step, the identified techniques and tools are combined to design an approach for extracting facts from images. The fourth step involves the prototypical implementation of the design. In the final step, an experimental analysis is performed using the prototypical implementation.

The results of this thesis are the following. There are two implementations for the design. Jupyter Notebooks showcase the execution of the pipeline and build a potential environment for students in this field to work on. A second implementation showcases

the use of the pipeline in a cloud environment. Both implementations are available on the GitLab of the university[3]. Several artifacts are created for these implementations: An entity linking document, presented in Listing A.1, contains links to Wikidata and ConceptNet for all annotated objects in Visual Genome. A set of 150 faces has been persisted in the Azure face recognition service. Each face is linked to Wikipedia in a linking document. Entity linking itself was implemented in a separate project together with another student who worked on entity linking in text. This separate project was built to isolate the entity-linking project for the FactCheck framework. This project is available on the GitLab of the university[4]. For experimental analysis, there are two approaches. The first approach analyses 20 images with 9 configurations each. This approach utilizes an annotated ground truth specifically created for this thesis, which is presented in Listing A.3. The second approach is a use case study. The use case study concludes an in-depth analysis of a single case with the same methods as in the first approach. With these experiments, it can be demonstrated that the accuracy of object detection and scene graph generation directly influences the quality of facts. The quality of facts is measured using various methods, including the scoring of triplets. The prediction model calculates scores for all potential triplets in a knowledge base. The generated triplets achieve high scores in the prediction model.

## 1.4. Outline

The remaining chapters of the thesis are organized as follows: Chapter 2 gives an overview of the tools and techniques used for this thesis. Chapter 3 provides an overview of approaches that could have been employed to address the research question. However, these approaches have not been used. Chapter 4 presents the requirements derived from the research questions and discusses the design of the approach taken. Chapter 5 is about the implementations of the prototype. Chapter 6 presents the results of the experimental analysis. Chapter 7 summarizes the approach taken and concludes the thesis. Lastly, Chapter 8 offers an outlook on emerging research questions.

[3] https://git01lab.cs.univie.ac.at/thesis/2021ss/1209967-adrian-hofer
[4] https://git01lab.cs.univie.ac.at/a1305573/entitylinking

# 2. Background

This chapter provides a brief summary of the literature relating to the concepts used in Chapter 4. In the first section, the FactCheck framework is introduced. Regarding object detection, the evolution of the most important object detection algorithms is illustrated. The evolution of the algorithms gives a more profound understanding of the particular concepts. For scene graphs, the first paper on scene graphs is introduced. Different implementations of scene graphs are explained. These implementations are used later on to execute experiments on the fact extraction pipeline. The face recognition section provides an overview of different face recognition algorithms and discusses the use of face recognition in Azure. The ontology section discusses why it is useful to utilize an ontology to tackle the problem of structured data in fact extraction from images.

## 2.1. FactCheck

Nowadays, in web pages, structured information through semantically rich embedded data is contained more frequently. This structured data is used by search engines to provide improved results. The usage of structured data by search engines might motivate web page owners and maintainers to further integrate structured data into their web pages. However, structured data on the web is often conflicting, which is problematic considering its wide adoption. Conflicting data on the web can influence services and applications depending on structured data. E.g., in 2016, the loser of the second ballot of the Austrian presidential election was falsely identified as Austria's new president by Google Search[1]. Google Search uses data fusion to clean, aggregate, and integrate data of an entity available from various data providers into a single clean, consistent representation. However, these services can fail.

A potential solution is the FactCheck framework [2], which was proposed to detect and resolve conflicts in structured data on the web. The framework offers three benefits: (i) users can become aware of conflicting structured data, so the user can derive a potential problem; (ii) data owners, analytic software, web services, or crawlers can receive feedback on their data, leading to improvements in accuracy. This feedback contributes to the implicit enhancement of structured data on the web, allowing applications to rely on more consistent structured data; (iii) the framework does not require any complex data fusion algorithm. Initially, the FactCheck framework was applied to the textual domain. However, it is expected that the model can be extended to other domains like images.

[1] https://www.spiegel.de/netzwelt/web/oesterreich-google-legt-sich-versehentlich-schon-hofer-als-bundespraesident-fest-a-00000000-0003-0001-0000-000000582674

## 2.2. Object Detection

At first, when extracting facts from images, it is important to make a computer understand which objects are depicted in an image. In Computer Vision (CV), the field of object detection has grown large interest over the last few years. The application of CV can range from an image search on the web to unlocking your smartphone via the camera, to self-driving cars. When unlocking a smartphone, the task of the CV application is to classify the given image in classes of whether the given image shows the face of the smartphone's owner. Self-driving cars and other applications like this fact-checking, one needs to know what is shown in the image. For this purpose, different implementations of Convolutional Neural Network can be used. However, solely relying on CNNs is too slow because this would mean taking a different Region of Interest (ROI) and classifying the presence of an object in a certain ROI. This approach would take a huge number of ROIs and hence would take a huge amount of computational power [3]. In the following, Convolutional Neural Network(CNN), Regions with Convolutional Neural Network (R-CNN), Fast Regions with Convolutional Neural Network (Fast R-CNN), and Faster Regions with Convolutional Neural Network (Faster R-CNN) will be explained. At the conclusion of this chapter, it will be explained why R-CNN was taken over other approaches like You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD) and others.

### 2.2.1. Convolutional Neural Network

In this section, the CNN layers as implemented by Le Cun et al. [4] will be explained. CNNs consist of different layers to reduce the computation time and resources needed to classify an image, as shown in Figure 2.1. Without a CNN, an image's features would need to be computed for every pixel. The image's features are needed to feed it into the neural networks' perceptrons to be able to classify the image. However, with a CNN, the image is first convoluted with a kernel. This kernel can have different sizes, e.g., 3×3 or 5×5, and different objectives, e.g., edge detection. The result of the convolution is a new matrix representing the image, we call it the output image. This output image can be the same size as the input image or a smaller size depending on the padding used and the stride used. There are two padding strategies: Same Padding where the input image is padded with a neutral element and the output image after the convolution is the same size as the input image. E.g., an input image of the size 5×5 would be convoluted with a 3×3 kernel, and the input image would be padded with one pixel, resulting in an input image of the size 6×6. When convoluting this 6×6 input image with a 3×3 kernel, the output image is of size 5×5. The other padding strategy is Valid Padding. In Valid Padding, the input image is not padded with a neutral element and the output image is, depending on the stride and the kernel size, smaller than the input image. E.g., if the input image is of size 5×5 and convoluted with a 3×3 kernel, the output image is of size 3×3. The next layer is sub-sampling. The goal of sub-sampling is to reduce the computational time needed through dimensional reduction. Furthermore, pooling is useful to extract dominant features. There are two different types of pooling: Max-pooling and

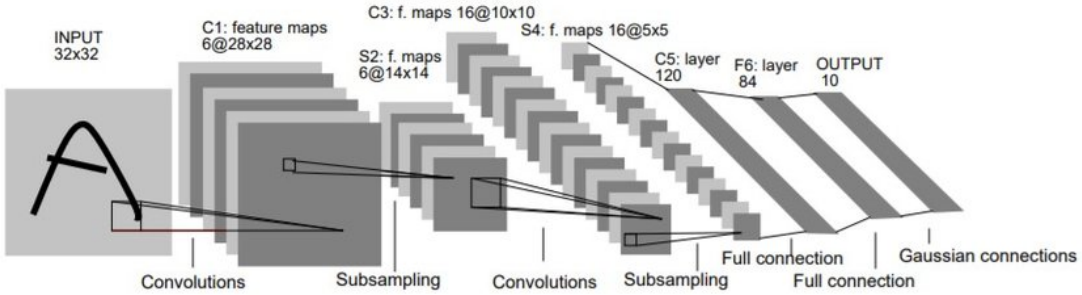



Figure 2.1.: CNN Overview [4]

Average-pooling. While Max-pooling, the largest pixel value in a searched area of the pixel is taken to represent this searched area of pixels. In Average-pooling, the average of the pixel values in the searched area is computed. The advantage of Max-pooling over Average-pooling is that Max-pooling performs a noise reduction when computing the dimensional reduction, while Average-pooling is only computing a dimensional reduction. Max-pooling can act as a noise reduction because the most present feature from the convolution layer is taken. The process of convolution and sub-sampling an input image can be applied many times to reduce the complexity of the input image. After this process of convolution and sub-sampling, the output image is flattened into a one-dimensional vector and sent to a neural network to be classified by perceptrons in a class.

### 2.2.2. Regions with Convolutional Neural Network

This section explains the idea of a RCNN as first proposed by Girshick et al. [3]. Their idea was to increase the performance of CNN by providing region proposals. Their design consists of three modules: region proposals, convolutional neural network, and class-specific linear Support Vector Machine (SVM) as shown in Figure 2.2.

The first module computes around 2000 region proposals for an image. These 2000 regions are proposed by a selective search algorithm [5]. This algorithm works the following way:

- The method of Felzenszwalb and Huttenlocher [6] is used to generate initial regions.
- A greedy algorithm is used to combine similar neighboring regions. Afterwards, the similarity of the newly created region and its neighbors is computed. This process continues until the whole image is one region.
- Use the generated regions in the step to produce the region proposals.

Next, a CNN is used to generate a 4096-dimensional feature vector for each region proposal. In the last step, an SVM is used to classify the presence of a certain object in that region.

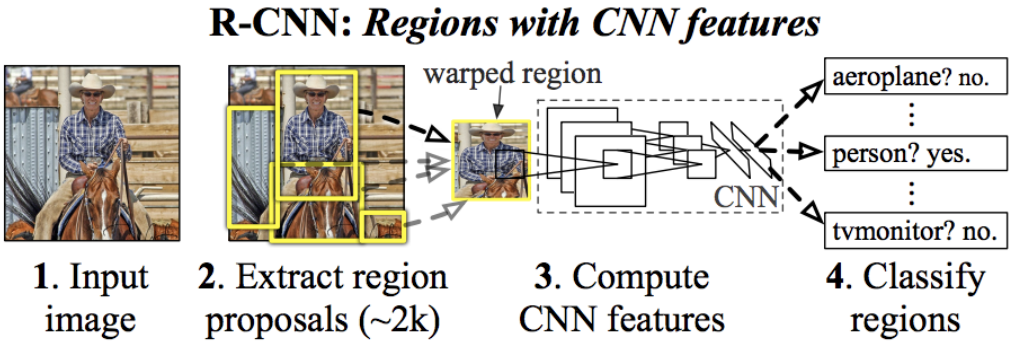



Figure 2.2.: RCNN Overview [3]

### 2.2.3. Fast Regions with Convolutional Neural Network

Fast R-CNN [7] is an optimization of the R-CNN algorithm developed by the R-CNN's original author Girshick et al. [3]. The approach of fast R-CNN is similar to R-CNN. The differences are that the CNN is calculated from the input image without region proposals. The region proposals are calculated from the output of the CNN. The region proposals are warped into squares by using an ROI pooling layer and reshaped to a fixed size to be able to feed them into a fully connected network. The ROI feature vector is used with a soft max pooling layer to be able to predict the class of the region. Fast R-CNN is faster than R-CNN because the expensive R-CNN operation only needs to be done once per image.

### 2.2.4. Faster Regions with Convolutional Neural Network

Faster R-CNN is an optimization of the fast R-CNN algorithm developed by Ren et al. [8] including the R-CNN's original author Girshick. Both of Girshick's algorithms, R-CNN and fast R-CNN, use the selective search algorithm by Uijlings et al. [5]. However, this algorithm is slow. Hence, Ren et al. [8] proposed a separate network to compute the region proposals instead of the selective search algorithm. This approach is remarkably faster than the fast R-CNN method. The CNN technology has come a long way since its inception in the 1990s when it was widely used, only to be surpassed by other technologies like SVM. However, in the 2010s, thanks to the innovations by Girshick et al. [3] and Ren et al. [8], CNN technology experienced a resurgence, significantly improving its performance. The results of [8] demonstrate that Faster R-CNN outperforms Fast R-CNN in terms of both speed and accuracy. Faster R-CNN achieves a higher mAP@.5 (mean average precision at an IoU threshold of 0.5) of 2.8% and a higher mAP@[.5, .95] (mean average precision averaged over IoU thresholds ranging from 0.5 to 0.95) of 2.2%.

## 2.3. Scene Graph

Regarding the translation of an image into structured data in an RDF, it could be useful to use a scene graph. Using a scene graph could produce a triplet data structure from an image. In 2015, Johnson et al. [9] proposed an approach to generate scene graphs from images. A scene graph is defined as follows [9]: "*A scene graph is a data structure that describes the contents of a scene. A scene graph encodes object instances, attributes of objects, and relationships between objects.*" An object could be a person, maybe some politician or an actor. An attribute describes color $\langle suit, is, blue \rangle$, shape $\langle pupil, is, round \rangle$, and pose $\langle leg, is, bent \rangle$. A relation between objects could be a geometry $\langle politician, besides, actor \rangle$, an action $\langle politician, walking\ on, street \rangle$, or object parts $\langle actor, has, hand \rangle$. A formal definition of a scene graph and a discussion about grounding image to scene graphs will follow in Section 4.2.2.

This work utilizes three distinct implementations for generating scene graphs. The forthcoming three sections will delve into the details of each implementation, explaining how they function.

### 2.3.1. Scene Graph Generation

In general, the generation of a scene graph involves a combination of object detection and a secondary model that calculates the relationships or relatedness among the detected objects. The goal of scene graph generation is to produce triplets in the form of $\langle object, relation, object \rangle$, abbreviated as $\langle o, r, o' \rangle$. See [10] for more information on different scene graphs. In this thesis, three different implementations of scene graph generation were utilized to conduct experiments, with the objective of comparing their performance in extracting facts. A fourth implementation will be presented in Section 2.3.4 to illustrate the formation between the approaches.

### 2.3.2. Iterative Message Passing

This approach [11] uses a joint inference framework, which enables information about the context to be sent through the scene graph by message passing. The challenge with this approach is that inference on a dense graph can be expensive, as was shown in [12] and [13]. To overcome this problem, a Gated Recurrent Unit (GRU) [14], which is a recurrent neural network [13], is used. A GRU is a gating mechanism in neural networks with a forget gate. The input of a GRU is its previous-hidden state and message. The GRU produces a new hidden state as output. For each node and edge in the scene graph, the model maintains an internal state and a corresponding GRU unit. The benefit of this approach is the possibility of message passing between GRU units.

During the experimental setup of the neural network, it was found that this architecture produces two sub-graphs of the scene graph. One edge set, consisting of the edge GRUs, and one node set, consisting of the node GRUs. These two graphs form a bipartite graph. No messages are passed inside these two sets. This observation can be used to define channels for iterative message passing between the two sets.

### 2.3.3. Multi-level Scene Description Network

Multi-level Scene Description Network (MSDN) [15] consists of five phases, as shown in Figure 2.3. 1) Object detection. Region proposals are used to generate ROIs. A Regional Proposal Network [8] is used to generate the object regional proposals. 2) Generate specialized features from ROIs for various semantic tasks. To generate specialized features, different branches are used. Different branches have different vision tasks. 3) Using semantic and spatial relationships of ROIs generates a graph model dynamically. While constructing, phrase proposals, connections between phrases and objects are built. A phrase proposal is connected to two object proposals, which form a triplet of $\langle object, relation, object' \rangle$. A graph is constructed to model the connections between objects and phrases. 4) Refine the specialized features by passing messages of different semantic levels in the graph. This process is parallelized, between object refining and phrase refining. 5) Finally, the refined features are used to predict classes for objects and predicates. The scene graph itself is generated from the predicted classes and their recognized relationships, which are the predicates. A matrix of i×j elements is built. The $\langle i, j \rangle$ element at the diagonal position is the ith object and the $\langle i, j \rangle$ element where $i! = j$ is the phrase for the relation between the objects i and j. An ith object can be an object class or "background". Likewise, the $\langle i.j \rangle$ phrase is a predefined predicate class or "irrelevant". The scene graph is generated based on this matrix.

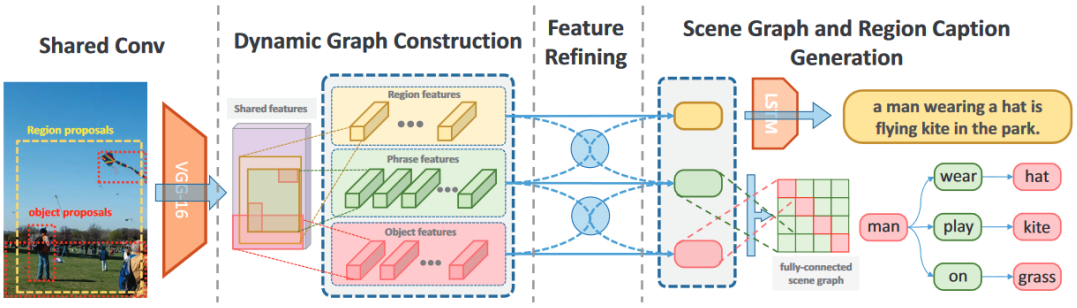



Figure 2.3.: MSDN Overview [15]

### 2.3.4. Motifnet

A Motif [16] is a mined node-edge-node or object-relation-object combination which can have different lengths. These object-relation-object combinations have a high degree of combined information, e.g., $\langle kid, playing, snow \rangle$ and $\langle kid, eating, snow \rangle$. Motifs were obtained by mining the Visual Genome [17] data set.

In Motifnet [18], the assumption is that strong independent local predictions about the graph's context perform less than global staged predictions (Motifs). Therefore, Motifnet builds upon staging box predictions, object classification and relationship prediction in a way that the predictions of previous stages establish a context for upcoming stages.

### 2.3.5. Graph-RCNN

This model proposed in [19] can be divided into three steps: 1) object node detection and extraction, 2) pruning of relationship edges between object nodes, and 3) integration of graph context to the scene graph. For step one, [19] claimed that a standard object detection pipeline can be used. Such standard object detection pipelines are described in Section 2.2. How the result of object detection could look is shown in Figure 2.4(a) and Figure 2.4(b). The novel approach of Graph-RCNN is introduced in steps two and three. In step two, they introduced a relation proposal network (RelPn). This network learns how to compute relatedness scores between object pairs. Those relatedness scores are used to prune connections between objects to create a sparse graph. In prior works of [19] connections were pruned randomly. An example of a sparse graph is shown in Figure 2.4(c). In step three, they introduced an attentional graph convolution network (aGCN) to the sparsely connected graph. The aGCN is used to propagate higher-order context through the graph. Meaning, the model updates the representation of each object and relation in the scene graph based on its neighbors, resulting in Figure 2.4(d).

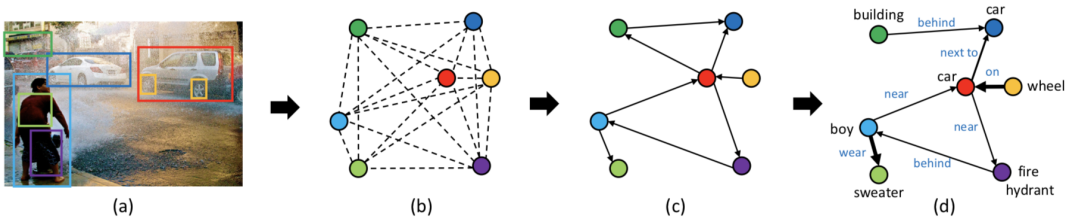



Figure 2.4.: Graph-RCNN Overview [19]

## 2.4. Face Recognition

Face recognition could be used to extract knowledge about a human from an image. Humans can recognize faces without effort [20], but for computers the task is challenging. In general, there are four steps to be performed to identify a human face [20]. First, face detection is used to identify the ROI and separate the face from the background. Second, the face is normalized to account for image size, face poses and photographic properties such as lighting and color. Third, facial features are computed. Fourth, the facial feature vectors are compared and matched against a database.

### 2.4.1. Face Recognition Algorithms

There is a wide range of face recognition algorithms. For reference of methods used in face detection and face recognition, compare [21] and [22]. However, today, deep face recognition is most widely used [23]. Most of these deep learning methods work similarly to the algorithm presented in Subsection Convolution Neural Network Cascade

of this section. In the following, a brief overlook of the most common face recognition algorithms [21] will be given.

### Viola Jones

Viola Jones is an algorithm [24] for face detection developed by Pail Viola and Michael Jones. The algorithm is capable of detecting faces in real time, the algorithm can process 15 frames per second.
The algorithm has four stages:

1. **Haar feature selection**. Common human face properties are matched using Haar-like features. Examples of common human face properties are that the eye region is darker than the nose bridge region, the location of the mouth, nose, eyes etc. The features are calculated and then searched in the image.

2. **Creating an integral image**. An integral image is a very fast feature calculation. An integral image at a location x, y contains the sum of all pixels above and left to x and y, including x and y.

3. **AdaBoost training**. This classifier discards features that are not necessarily part of the face.

4. **Cascading classifiers**. A total of 38 classifiers, each one slightly more complex than the one before. If a classifier rejects the feature, the image is discarded.

### Robust Face Detection Using the Hausdorff Distance

This is a model-based algorithm [25] that operates on grayscale images. The Hausdorff Distance is used to compute the similarity between the object or region in the image where the face needs to be detected and a sample face.
The algorithm works in two stages:

1. **Corase detection**. A ROI is defined which is re-sampled to a fixed size. The image is segmented using the Sobel operator. Using a face model and a binary representation of an image, a localization of the face in the image can be performed.

2. **Refinement**. This step is similar to the first one, but a more detailed model of the face is used.

### Convolution Neural Network Cascade

In opposition to the previous algorithm, this algorithm [26] is not using hand-crafted features but learns the features by itself using CNN. A more detailed description of CNN can be found in Section 2.2.1. For Convolution Neural Network Cascade, a cascade of 6 CNNs is used. The first three CNNs classify the faces into faces and non-faces, and the latter three calculate bounding boxes for the faces. Each second CNN is a calibration.
The six CNNs are the following:

1. **12-net**. Scan over the whole image at different scales to reject false detection windows.

2. **12-calibration-net**. Process the detection windows as 12×12 images to adjust size and location.

3. **24-net**. Apply Non-maximum Supression (NMS) to eliminate overlapped detection windows. The remaining windows are cropped out into 24×24 for further rejection.

4. **24-calibration-net**. Process the detection windows as 24×24 images to adjust size and location.

5. **48-net**. This step takes 48×48 input images and is used for further elimination and evaluation. The elimination works like the following, apply NMS to eliminate detection windows with an Intersection-over-Union (IoU) ratio over a pre-defined threshold to the currently selected window with the highest confidence score.

6. **48-calibration-net** Process the detection windows as 24×24 images to adjust size and location to process Bounding Boxes for the faces.

#### Detecting Faces Using Eigenface

This algorithm [27] uses Eigenface [28] to detect faces. An image provided to a computer can potentially contain a lot of noise like backgrounds, lights, face poses, etc. However, an image of a face contains certain patterns which appear in every image of a face. E.g., some objects like eyes, noses, and mouths share a relative distance. Such patterns of features are called Eigenfaces or principal components. These features can be extracted using Principal Component Analysis (PCA). An image can be reconstructed from its principal components. To reconstruct an image from its principal components, each principal component is given some weight according to its presence in the image. By adding up all weighted principal components, the original image is reconstructed.
The algorithm itself [27] works as the following:

1. **Create Eigenfaces**. Eigenfaces are computed from the input image.

2. **Calculate weights**. Calculate the weights of the image's Eigenfaces.

3. **Compare weights**. Using the Euclidean distance, compare the weights of the image's Eigenfaces against all other weights saved in the database. The nearest match is considered the match.

### 2.4.2. Azure Face Recognition

In this thesis, the Azure face recognition service[2] was used. Though the service is proprietary, in the following a short overview is given by what is known from the

| detection_01 | detection_02 | detection_03 |
|---|---|---|
| The default choice for all face detection operations. | Released in May 2019 and available optionally in all face detection operations. | Released in February 2021 and available optionally in all face detection operations. |
| Not optimized for small, side-view, or blurry faces. | Improved accuracy on small, side-view, and blurry faces. | Further, improved accuracy, including on smaller faces (64×64 pixels) and rotated face orientations. |
| Returns main face attributes (head pose, age, emotion, and so on) if they are specified in the detect call. | Does not return face attributes. | Returns mask and head pose attributes if they are specified in the detect call. |
| Returns face landmarks if they are specified in the detect call. | Does not return to face landmarks. | Returns face landmarks if they are specified in the detect call. |

Table 2.1.: Azure Face Detection Models[4]

documentation[3] about the service. The pipeline is arranged in three steps, Detection, Identification, and Verification. There is an online Application Programming Interface (API) for the face recognition service as well as some Software Development Kits (SDK) for different programming languages including C# and Python. When querying a face for recognition on Azure, there are some prerequisites to be fulfilled. For Detection, there are three different models available, as can be seen in Table 2.1. A face can only be queried against an existing group of individuals. A face recognition model is trained using that group of individuals. For recognition, there are four recognition models available: recognition_01, recognition_02, recognition_03, and recognition_04. However, there is no exact documentation as to what the difference is between the four models. There are no pre-trained person groups available. The result of a query is a JavaScript Object Notation (JSON) document containing candidates with identifiers (ID) for individuals from the pre-trained person group and a confidence value for each candidate.

## 2.5. Ontologies

### 2.5.1. What Are Ontologies?

Regarding high-quality structured data, it can be useful to use ontologies. The concept of ”ontology“ has a long tradition in the philosophical sphere, where it poses the question if universal entities do exist or to encompass problems about the most general features

[2]https://azure.microsoft.com/en-us/products/cognitive-services/face/#overview
[3]https://learn.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-identity

and relations of the entities which do exist [29]. It is often confused with epistemology, which is the study of knowledge and justified belief [30].

Gruber et al. [31] give the following definition of ontology in the context of knowledge sharing: "*In the context of knowledge sharing, I use the term ontology to mean a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as a set-of-concept definition, but more general. And it is certainly a different sense of the word than its use in philosophy.*"

On the Semantic Web, vocabularies or ontologies are used to define the concepts and relationships (referred to as "terms") that are employed to describe and represent a specific area of concern [32]. Vocabularies or ontologies are utilized for the classification of terms within a particular application. They define relations and impose constraints on those relations.

There is a trend to use the term "ontology" for a more complex and potentially formal collection of terms, while the term "vocabulary" is employed when a strict formalism is not necessarily required, or only in a loose sense [32].

### 2.5.2. Why Do We Need Ontologies?

The reason for using an Ontology is to help data integration. Ontologies can be beneficial when there exist ambiguities in the terms used in different data sets, or when extra knowledge could be helpful to discover new relationships. In the field of extracting facts from images, it is important to define the structured data about an image in a way that can be compared by other applications using those facts. To be able to combine facts about images in the future, it is necessary to have a well-defined ontology on the structured data of the images. Another reason for using an ontology is that objects in an image can be ambiguous. E.g., there may be a bear in an image. However, this bear might be a teddy bear. Another example is the usage of the word riding. There is a difference between riding a bike and riding a horse. These ambiguities have to be resolved in the underlying ontology.

## 2.6. Summary

This chapter has reviewed the key technologies of extracting facts from images. These techniques will be used later on in Chapter 4 to build the fundament for the design of this thesis. It has been shown that Faster-RCNN is a feasible technique to detect objects. Considering scene graphs, it is shown that Graph-RCNN adds an interesting approach to generating scene graphs. The different settings for the Azure face recognition API are shown. For ontologies, it was discussed how they can be used to resolve ambiguities.

# 3. Related Work

This chapter summarizes the literature analysis relevant for this thesis. The most interesting findings will be presented. Those findings have not been used for solving the research question of this thesis. However, this chapter should give an overview of the literature searched to give a frame of why the techniques in Chapter 2 have been used. Furthermore, the chapter includes several interesting ideas of how extracting facts from images could be done or could be improved.

## 3.1. Studies on Fake News

There is a huge number of studies on fake news available. The most relevant to extracting facts from images are presented in the following. Spezzano et al. [33] showed in their study on the performance of humans and computers in detecting fake news that some features and facts are better to detect fake news than others.
They used:

1. ImageNet-VGG193, a state-of-the-art deep-learning-based technique to extract features from the images [34],

2. features describing facial emotions,

3. features referring to image quality such as noise and blur detection

To extract features describing facial emotions they used Microsoft Azure. They showed that they can achieve a fake news recognition of 78% ± 5% when combining these features with the title.

Gupta et al. [35] analyzed tweets and images shared during Hurricane Sandy and identified different features which can be used when detecting fake news on Twitter. He found Tweet features to be more effective compared to user features. Tweet features are features like tweet length concerning the tweet. User features are features like the number of followers concerning a user. They as well highlighted the role of Uniform Resource Locator (URL) shortening services like bit.ly. Chhabra et al. [36] showed that these services are not only used to shorten the URL and, hence, save space, but these URL shortening services are also used to hide phishing links since users are unaware of the destination of the shortened URL.

## 3.2. Fact-checking Frameworks

Fact-checking is a fast area with plenty of different approaches. Frameworks working with images are presented in this section. A different approach to Kalchgruber et al. [2] is the Kauwa-Kaate Fake News Detection System [37]. This approach indexes articles crawled on trusted news and fact-checking websites by their content. When an article is to be fact-checked, this approach searches the crawled indexes, intending to determine if the queried article is available on trusted news or fact-checking websites. The interesting aspect of this approach is its handling of images. To index images, they use exact and approximate features. These features are compared similarly to Jnoub et al. [38]. However, they are not using Speeded up Robust Features (SURF), but an approach based on Wong et al. [39]. An implementation of the used approach can be found at `https://github.com/ProvenanceLabs/image-match`. The problem with the Kauwa-Kaate Fake News Detection System [37] is that it fails if an article is queried respectively fact-checked, which is not present on a trusted news site or a fact-checking site. They argue that they are crawling other websites as well to determine if the queried article is present on those other websites. Furthermore, they are planning to build a system to introduce a bias towards authors and websites to be able to trust them.

Different other fact-checking frameworks were searched, but none of them implements image processing yet. Searched frameworks were FNED [40], YAGO2 [41], FACE-KEG [42] and others [43], [44], [45].

## 3.3. Feature-based Entity Linking

This section presents different methods which could be used to link objects in an image with entities in a knowledge base. In Section 3.10 another state-of-the-art approach for solving this problem is discussed in detail.

Csurka et al. [46] invented a method to classify images before neural networks. This method was heavily used for classifying images in the 2000s. Their approach is derived from the well-known Bag of Words approach and is called Bag of Visual Words (BoV). First, they detect and describe image patches, for this, e.g., Scale-invariant Feature Transform (SIFT) [47] can be used. Second, they assign a path to a cluster, e.g., with k-means. Third, they construct a bag of key points. This bag of key points counts the number of patches assigned to each cluster. With this approach, common key points in images can be detected and through quantization, these common key points can be generalized.

Viitaniemi et al. [48] proposed a method to enhance the Bag of visual Word approach by adding information on the spatial distribution of descriptors. These two BoV approaches by Csurka et al. [46] and Viitaniemi et al. [48] could be helpful for the generation of approximate features of images.

Jin et al. [49] proposed an approach to verify news by visual and statistical features extracted from images for microblogs. They proposed six visual image features to be computed out of an image against a collection of images. Furthermore, they mentioned

that it was pointed out by Sun et al. [50] that fake news is more likely to contain outdated images than recent ones. For that, they proposed a time span feature. Plenty of survey papers claim that these features by Jin et al. [49] are the newest visual features available. These features could be used in giving a user a first impression of how likely it is that an image is fake.

Cao et al. [51] show a method to extract semantic features out of images and text and make them comparable. This method should enable the retrieval of images via text and vice versa. Their approach to comparing the semantics of image and textual features could be helpful to evaluate a fact extraction method, or the method used to extract semantics out of images could be used to generate facts.

## 3.4. Knowledge-based Caption Generation

The integration of knowledge in knowledge bases with object detection could be an interesting approach to start improving scene graph results with information from knowledge bases. This information from knowledge bases could be added to the neural network of the scene graph generation. At least, they are worth mentioning on their own because these approaches could be used to extract facts from images on their own.

Weiland et al. [52] show how to obtain a description of an image thanks to Wikipedia and graph theory. They can describe the message encoded in an image. In other words, the description of an image that is not visible is extracted from the image by object recognition and projection of the extracted object onto nodes in the knowledge graph. Thanks to graph-theoretical approaches (path findings between the nodes) and adding nodes that are not visible in the image but exist in the knowledge graph between the selected nodes, they can construct knowledge from the extracted objects. The back leash of their approach was that object recognition was not good enough at this point, so they simulated perfect object recognition.

Chaudhary et al. [53] propose a method for image retrieval with complex queries. They can embed knowledge from a selected knowledge graph into an image and retrieve an image search with a complex query. Complex queries in their sense are queries where the message of the query is not obvious in the sentence at first when analyzing the query word by word. They use the example query “which animals are polar bear prey”. The result of this example query should not show polar bears, but seals or other polar bear prey. By enriching images with semantics, they claim to be able to answer such queries with their approach. They show results depending on the used knowledge base, ranging from 62.343% to 80.920% of relevant concepts found for images.

Battad et al. [54] aim to generate a possibility for human and computer chatting to make computers capable of talking about the content of images. They start by generating hypotheses about images and result in a knowledge graph about an image.

All these approaches could be the groundwork for fact extraction from images. At least, the approach to using object recognition on images and linking these objects to nodes on a knowledge graph is groundwork.

## 3.5. Image Caption

Image captions could be used to generate a caption for images, and further use entity linking to the text to extract facts from images. The three best-performing approaches identified in the image caption survey conducted by Hossain et al. [55] are presented in this section.

Wu et al. [56] show an approach to generate image captioning which seeks to process directly from image features. They propose a method that uses high-level features of CNNs and Recurrent Neural Networks (RNN) to generate captions for images.

Xu et al. [57] invented an attention base model, which can learn to describe an image's content in words. They train their model deterministically, utilizing standard backpropagation techniques and maximizing the lower bound. They validate their claims with three benchmark data sets, Flickr8k[1], Flickr30k[2] and MS COCO[3].

Rennie et al. [58] are using a policy-gradient method for reinforcement learning to train their deep end-to-end system. They show that they can optimize their system for generating image captions with this system in MS COCO.

Since an image caption is nothing else than a description of an image, these approaches can be seen as rudimentary knowledge about images. However, they are not as sophisticated as knowledge-based approaches because they only describe what can be seen in an image.

## 3.6. Tags

Tag-related methods could be an approach to extracting knowledge from images on social media. This should be seen as an extension of fact extraction from images for social media.

Maenishi et al. [59] show a method on how to distinguish tags that describe the content of images from tags that are describing the meta-data. Song et al. [60] give an approach to how to link user-generated tags to a knowledge base to generate knowledge out of the image's tags. Abdulraheem et al. [61] discuss an approach to generate tags for images automatically by analyzing the comments on the image in social media. However, this is only possible if an initial tag for the image is given. The whole method is used to enrich tags.

## 3.7. Optical Character Recognition

Optical Character Recognition (OCR) is the process of recognizing text on images. This could be helpful, especially for memes shared on social media.

Wang et al. [62] give a solution for reading and describing text on an image. Their architecture makes it possible to even give a spatial relation between object and text in

[1] https://hockenmaier.cs.illinois.edu/8k-pictures.html
[2] https://shannon.cs.illinois.edu/DenotationGraph/
[3] https://cocodataset.org/

images and therefore can extract knowledge out of images that contain text. Fedor et al. [63] give an architecture at a large scale for recognizing and detecting text in images. Their approach claims to be able to process the number of images uploaded daily on Facebook. Deshpande et al. [64] have developed a system to detect hateful memes and scale the hatefulness in an image by scalars. They use a mixed approach of textual features and image features.

## 3.8. Open Linked Data and Ontologies

If there are databases out there already having links of images to structured data, they could and should be used. However, the main problem is the timeliness or currentness of the data.

An interesting project is the IMGpedia: A Linked Dataset with Content-based Analysis of Wikimedia Images [65]. IMGpedia has built a database from all Wikimedia images available in 2017 and extracted three features Gray Histogram Descriptor, Histogram of Oriented Gradients Descriptor, and Color Layout Descriptor. They claim that these three feature descriptors are enough to find a similar image if the database is queried with a new image. Since they use Wikipedia's linked data set, they can gain knowledge on a queried image if it is found in the extracted Wikimedia database. This approach has two downsides: First, if an image is not on Wikimedia, they are unable to gain knowledge about this image. Second, the database is a snapshot of the 2017 Wikimedia. However, fake news is contemporary, meaning that in order for the database to be usable, it would need to be regularly updated

Ousmer et al. [66] invented an ontology for body-based gestures originally for movement detection in virtual reality, which can be used to describe body gestures in images. Bashar et al. [67] came up with an ontology for context-aware adaptive face recognition. They are focusing heavily on robustness against illumination variation. Tashiro et al. [68] show a method for pose recognition thanks to computer vision, followed by an ontology to describe these poses. Berthelon et al. [69] invented an ontology for emotions with embedded context awareness.

## 3.9. Cloud Service Analyzes

Since the FactCheck framework should be deployed in a cloud environment, the pros and cons of different cloud providers should be researched. For extracting facts from images, it is interesting to know which services are offered by different cloud providers.

Osamah et al. [70] give an overview of different Cloud Service Face Emotion Recognition Services. They are comparing Amazon Web Services (AWS), Google Cloud and Microsoft Azure. Following their research, Google has the highest accuracy in determining a person's emotion, however, their number of supported emotions is very limited. Microsoft Azure acquired the highest accuracy when a selected number of emotions were tested. This evaluation should be rated higher since Microsoft Azure has a broader palette of emotions and hence a lower accuracy if considering all emotions available on a platform.

Rawan et al. [71] give an overview of the most popular cloud computing offerings in the high-performance sector. They investigate solutions wherein another user would have deployed a real dedicated cluster. They are comparing Microsoft Azure, AWS, Google Cloud and Oracle Cloud. The criteria to compare the providers are: On-demand pricing; Elasticity; Resources control, security, maintenance, and management; Cost-effectiveness; Remote Direct Memory Access (RDMA) Support.

Yulei et al. [72] are comparing the performance of image and video services offered by different public cloud providers. The cloud providers include Google cloud, Microsoft Azure, AWS, Alibaba Cloud, Baidu Cloud and Tencent Cloud. The image cloud services analyses are about different tasks including facial recognition, image analysis and OCR. For these tasks, they used different data sets all of at least containing one million images, except for OCR. For OCR they used a data set of 3000 images. The results differ for the selected tasks.

Ugetta et al. [73] give a benchmark on the Old Baily data set for OCR performance of Cloud Services. The Old Baily[4] data set is a corpus of over 180,000-page images of court records printed from April 1674 to April 1913. This paper gives an overview of the performance of OCR of AWS', Microsoft Azure's Cognitive Services (Azure), and Google Cloud Platform's Vision.

## 3.10. CLIP

The paper "Learning Transferable Visual Models From Natural Language Supervision" [74] referred to as CLIP, presents a natural language supervised approach, which retrieves data about an image from the vast text available on the internet. The authors of the paper constructed image-text pairs. The text part of the image-text pairs include one of a set of 500,000 queries. The authors class balanced the model by including 20,000 image-text pairs per query. The queries in the CLIP approach consist of words that occur in Wikipedia more than 100 times, along with bigrams exhibiting high point-wise mutual information. Additionally, names of Wikipedia articles above a certain threshold and all words in WordNet that have not been included yet are included as queries. In total, there are 500,000 queries generated using this combination of criteria. They compute image features with a CNN and text features with a text transformer and jointly train the model. With N image-text pairs CLIP is trained to predict which of the NxN possible image-text pairs actually occur. They compute the cosine similarity between the image feature and the text feature and try to minimize the cosine similarity of the NxN-N pairs which do not occur, while maximizing the cosine similarity of the N pairs which actually occur. This means the way in which the image and text encoder calculates the features of the image and the text are learned.

[4] https://www.oldbaileyonline.org/

## 3.11. Summary

In summary, it has been shown from this review that there are no fact-checking frameworks extracting facts from images and comparing those facts. In general, there are several methods for annotating images. However, there are no methods to generate structured data from images. It has been shown that there exists open data and ontologies that can be used to generate and model structured data about images. In the future, CLIP could be a promising candidate for linking entities to knowledge bases.

# 4. Conceptual Foundation and Design

The following part of this thesis moves on to describe in greater detail the conceptual foundation and design used to answer the research question in Section 1.2. In the first Section 4.1 a list of functional and non-functional requirements will be presented. These requirements are the foundation for the concept and design. The sum of particular concepts described in the second section should fulfill all functional and non-functional requirements. In the second Section 4.2 all particular concepts needed to extract facts from images by the approach presented in this thesis will be described. There will be a discussion of why certain concepts are used. A design for the usage of the concepts will be given. In the third Section 4.3 a process flow of the pipeline will be offered. There will be a comprehensive presentation of the process flow, that will result in a detailed description of how the key concepts presented in this section work together.

## 4.1. Requirements

This section lists all functional and non-functional requirements for the design of the approach in this thesis. Section 4.1.1 lists functionalities the design of the approach must be able to fulfill. In Section 4.1.2 requirements are listed which can be used to judge the operation of the approach's design. The approach's design will be referred to as a pipeline, since the approach consists of several concepts which work together in a pipeline.

### 4.1.1. Functional Requirements

- **F1**: The pipeline must be executed exactly once for one image.
- **F2**: The pipeline must not be executed for copies of an image.
- **F3**: An image must be in one of the following categories: original, copy or tampered.
- **F4**: An image must be searched for known objects by object detection. An object in an image must have a confidence score. A link to a knowledge base from the object must exist. The spatial location of an object in the image must be persisted.
- **F5**: A scene graph of an image must be constructed. Relations in a scene graph must have a confidence score. A link to a knowledge base from the relationship must exist. A scene graph must be serialized and persisted in a text-based datatype.
- **F6**: An image containing humans must be searched by face recognition. A recognized face must have a link to a knowledge base.

- **F7**: Metadata about an image must be persisted. Metadata must contain the following information: filename, image size, image height, image width, image format, image mode, and all other available Exchangeable Image File Format (EXIF) information.
- **F8**: All persisted information about an image must be queryable.

### 4.1.2. Non-functional Requirements

Since the approach's design is referred to as a pipeline, the concepts working together in the pipeline will be referred to as services.

- **Compatibility**: The pipeline should be capable of being deployed on Azure.
- **Scalability**: The pipeline should be capable of scaling horizontally. Therefore, the pipeline should be composed of services. The communication between the services should be established by a message queue to guarantee asynchronous communication.
- **Independence of Services**: Services should work independently and perform all given tasks on their own. From a single input, one service must be able to perform a single output on its own.
- **Storage**: Images should not be stored, but only their links. The key points, descriptors, and the output of the system will be stored of an image. Larger files should be available online. Relations between an image and a file should be saved in a database. Both the database and the online file system must be capable of storing an increasing number of files and data. Therefore, these systems must be able to scale vertically.

## 4.2. Description of Particular Concepts

In Section 4.2.1 the used algorithm to categorize images into *originals*, *duplicates* and *replicates* will be presented and the categories themselves will be defined. There will be a discussion about the benefits and drawbacks of the design decisions and the opportunities of using this design in a FactCheck application. In Section 4.2.2 a reasoning for using scene graphs will be presented. It will be presented why scene graphs are best used for detecting objects in images and detecting relations between objects in images. There will be definitions for *subject*, *object* and *predicate* as these terms are derived from $\langle S, P, O \rangle$ triplet structure. A comprehensive analysis of what a fact in an image could be will be performed. In Section 4.2.3 there will be a debate about why it is necessary to use face recognition, what are the benefits and drawbacks of face recognition and why it is necessary to use face recognition for entity linking. In Section 4.2.4 an approach for combining the bounding boxes of face detection and results and bounding boxes of scene graph results will be presented. In Section 4.2.5 the concept of linking entities in

images to entities in knowledge bases will be presented. It will be argued why entity linking is necessary and what the design decisions resulting in using entity linking are. In Section 4.2.6 the use of metadata in the concept will be discussed. In Section 4.2.7 the design of the data model resulting from the key concepts presented before and resulting from the use intentionally in FactCheck will be presented.

### 4.2.1. Similarity Measurement

Similarity Measurement is used to categorize images into three different categories: *originals*, *duplicates*, and *replicates*. However, before these categories can be defined the properties of *originals*, *duplicates*, and *replicates* needs to be defined. The properties of the categories are their two features, exact features and approximate features.

**Definition 1 (Exact features)** *An exact feature is a description of an image as a number where the probability of a collision of two equal numbers is as low as possible.*

To generate a number from an image, the term hashing is used. The advocated hashing algorithm to generate hashes with a low collision probability is Secure Hash Algorithm-1 (SHA)-1 [75]. The output of SHA-1 is a 160-bit (20-byte) hash value. The probability of a collision in SHA-1 is:

$$p \leq \frac{n\,(n-1)}{2} \times \frac{1}{2^b} \tag{4.1}$$

where $n$ is the number of different data blocks and $b$ is the bits outputted by the hash function. Consider numerous images (e.g., $10^{14}$). The probability of a collision is less than $3.4 \times 10^{-21}\%$ using the SHA-1 hash function with $n = 10^{14}$ data blocks and $b = 160$ bits. The reason SHA-1 was chosen over other hash functions was the computation time of the SHA-1 hashing function compared to collision probability. Since there is a wide variety of hash functions and their best use cases, there might be better suited hash functions.

**Definition 2 (Approximate features)** *An approximate feature refers to a collection of descriptors associated with key points in an image. These descriptors are generated by applying a computer vision algorithm that aims to detect and describe local features within the image. The parameters for these features should be selected in such a way that they provide a wide range of potential K-nearest neighbor matching outcomes. This implies that the chosen parameters should allow for different levels of similarity and dissimilarity among the descriptors, enabling a flexible and diverse matching process.*

To obtain key points and descriptors of an image, there are several ways to compute them, e.g., SIFT [76] or Accelerated KAZE (AKAZE) [77]. Since descriptors are independent of key-point position, robust against image transformations, and scale independent, it makes them comparable for different angles, positions, and scales of the image. Hence, in this thesis, these features are called approximate since they can be matched by K-nearest neighbor matching.

Now that the key terms are defined, we can define the appropriation of these features in the categories.

**Definition 3 (Original)** *An image is called original if the exact features and approximate features of the image have not been found in the database at the time of processing the image.*

Originals are used to identify an image as the ground truth for other similar images. An original can only claim to be the first image processed by the pipeline. The image cannot claim to be the original image in the real world. There may be other images that are the real-world original, however, this image which is tagged as original in the database is the first one of that kind to be processed by the pipeline.

**Definition 4 (Duplicate)** *An image is called duplicate if the exact features of the image have been found in the database at the time of processing the image.*

A *duplicate* is an exact copy of an image that is tagged as original in the database. A duplicate needs to be a pixel-wise copy of an original. In other words, a duplicate must be the same image as the original, an image that creates the same SHA-1 fingerprint as the original. Any deviation that would result from different colors, transformations, rotations, scaling or any other image manipulation would result in the image being a replicate, not a duplicate.

**Definition 5 (Replicate)** *An image is called replicate if the approximate features of the image have been found in the database at the time of processing the image.*

A replicate can be an image depicting the same scene as an original, but something has been manipulated. The manipulation can be different colors, transformations, rotations, scaling or any other image manipulation.

The difference between an *original* and a *replicate* can be illustrated by an image of an apartment building like in Figure 4.1 (a) and (b). There might be a tampered version of

(a) Original (b) Replicate

Figure 4.1.: Examples for original and replicate Images[1]

[1] https://www.kaggle.com/datasets/saurabhshahane/cg1050

the original image (a) in (b) where the building has no windows. The absence of windows in (b) will result in different exact features of an image. However, the approximate features will stay the same for the original and the replicate.

These categories of *original*, *duplicate* and *replicate* are used to fulfil Functional Requirement **F3**: *An image must be in one of the following categories: original, Copy, Tampered.* Furthermore, with the application of these categories, the pipeline can fulfill Functional Requirement **F1**: *The pipeline must be executed exactly once for one image* and Functional Requirement **F2**: *The pipeline must not be executed for copies of an image.* The design motivation for the Functional Requirements F1-F3 was the following. First, to be able to decide for which images the pipeline needs to be executed. Second, to know for which images the pipeline has already been executed. Since the pipeline should not be executed twice for the same image, this behavior should reduce unnecessary resource consumption. Furthermore, the categorization of images into *originals*, *duplicates*, and *replicates* will benefit a future application of FactCheck. The application will be able to get the *original* and all other *replicates* for an already known image. For an unknown image, the application would be able to tell that this is the first occurrence of that image in the FactBase. It could be stated that an original does not need to be a real-world original, but is only the first image to be processed by the database. The terms *originals*, *duplicates*, and *replicates* don't need to be presented to the user. However, they describe the category from a technical perspective at best.

The idea for the used algorithm was taken from [38]. Their implementation has the advantage that the used algorithm was implemented on a blockchain. Its implementation on a blockchain is a benefit if the full trust of the image categorization is needed.

### 4.2.2. Scene Graph

This section describes how to fulfill **F5** and why this requirement has been chosen. Since scene graph generation always uses some object detection at first, as shown in Sections 2.3.1 - 2.3.5, this approach will fulfill **F4** as well.

The purpose of the scene graph is to create a relation between the objects or entities in an image. Consider a person on a sidewalk in an image. Is the person standing or walking on that sidewalk? There are different approaches how to achieve a connection between objects in an image. As proposed by [52], one would use knowledge-based linked detected objects in an image and use the graph structure of a knowledge base to compute the most likely connection between the two objects. This and other possible methods are described in Section 3.4. However, these methods are not as extendible and customizable as a self-learn deep neural network. Other methods than deep neural networks are not as extendible because they depend on an external knowledge graph. On the one hand, such an external knowledge graph needs to be remote to be updated (which is a benefit). Still, the knowledge graph cannot be updated easily due to the politics of Wikipedia or similar open-source projects. Every change needs to be reviewed by other members of the Wikipedia. Such politics can also benefit because the information gets more resilient. However, suppose the problem arises that the information provided is too high-level and the targeted information cannot be extracted (e.g., $\langle person, walking\ on, sidewalk \rangle$). In

that case, adding the information to the knowledge graph is hard. In such a way, a self-learned neural network is much more customizable because it can be learned to fulfill certain needs.

In 2015, Johnson et al. [9] proposed their work to generate scene graphs from images. Formally defined, a scene graph consists of a set of object classes $C$, a set of attributes $A$ and a set of relations $R$. Johnson et al. [9] define a scene graph $G$ to be a tuple $G = (O, E)$ where $O = \{o_1, \ldots, o_n\}$ is a set of objects, and $E \subseteq O \times R \times O$ is a set of edges. Each object has the form $o_i = (c_i, A_i)$ where $c_i \in C$ is the class of the object, and $A_i \subseteq A$ are the attributes of the object. This definition of a scene graph shows the benefit of scene graphs in generating facts from images. Since Kalchgruber defined facts as being the object values in a triplet [2], a scene graph consists of facts. The relation between objects could be a fact, the class of an object could be a fact, and attributes of objects could be considered facts because each of these occurrences of entities can be the object value in a triplet output by the scene graph. But how is a scene graph related to a real-world image?

Johnson et al. [9] state that a scene graph itself is not associated with an image; it merely describes a scene that the image could depict. There is a way of grounding images to scene graphs. Grounding an image to a scene graph means that each bounding box in the image can be linked to an object in the scene graph. This means that facts in a scene graph must also be grounded in the image. Facts need to have a bounding box to ground them to the image. If facts are grounded in an image, they could be considered as describing the image. What is the distribution of possible groundings of objects, and hence facts, in bounding boxes?

We note a scene graph as, $G = (O, E)$ where $O$ is a set of objects and $E$ is a set of edges. Let $B$ be a set of bounding boxes for ROIs in an image. Let $\gamma$ be a grounding, a link from the bounding box of an ROI in the image to an object in the scene graph. The variable $\gamma_o$ results from each object $o \in O$. Where the domain of definition, the range of defined values for the variable, for $\gamma_o$ is B. As a result, setting $\gamma_o = b \in B$ is the formulation of grounding an object $o$ in the scene graph to a box $b$ in the image. The distribution over possible groundings can be modeled [9] by Equation 4.2

$$P(\gamma \mid G, B) = \prod_{o \in O} (\gamma_o \mid o) \prod_{\langle o,r,o' \rangle \in E} P(\gamma_o, \gamma_{o'} \mid o, r, o') \tag{4.2}$$

$P(\gamma_o|o)$ can be rewritten by using Bayes' rule as $P(o|\gamma o)P(\gamma o)/P(o)$. $P(\gamma o)$ and $P(o)$ become constants when assuming uniform priors over the bounding boxes and object classes. Since they are constants, they can be ignored. Therefore, the final objective by [9] has the form.

$$\gamma^* = \arg\max_{\gamma} \prod_{o \in O} P(o \mid \gamma_o) \prod_{\langle o,r,o' \rangle \in E} P(\gamma_o, \gamma_{o'} \mid o, r, o') \tag{4.3}$$

**Unary potentials.** The term $P(o \mid \gamma_o)$ in Equation 4.3 is the unary potential modeling to which degree the figure in the bounding box $\gamma_o$ corresponds to the deduced object class and attributes of the single object $o$. If $o = (c, A)$ then [9] decomposes $P(o \mid \gamma_o)$ as Let $c$

be the object class and let $a$ be the attribute of a bounding box $o$ in Equation 4.4 [9]. The terms $P(c \mid \gamma_o)$ and $P(a \mid \gamma_o)$ are the probabilities that the bounding box $\gamma_o$ shows $c$ and $a$. Such probabilities can be modeled by using object detection like R-CNN [3] or Faster-R-CNN [7] as shown in [11], [16], [19], [15].

$$P(o \mid \gamma_o) = P(c \mid \gamma_o) \prod_{a \in A} P(a \mid \gamma_o) \tag{4.4}$$

**Binary potentials.** The term $P(\gamma_o, \gamma_{o'} \mid o, r, o')$ in Equation 4.3 is a binary potential modeling to which degree the pair of bounding boxes $\gamma_0$, $\gamma_{o'}$ corresponds to the tuple $\langle o, r, o' \rangle$.

Currently, we stated that a fact about an object $o$ in a scene graph could be the related object $o'$, and the grounding of that fact in the image could be the binary potential. To prevent misunderstanding, the following defined terms *subject*, *predicate*, and *object* will be written in italics. In contrast, the terms used in their general meaning will be written in plain text. The relation of each defined term to a knowledge base will be explained in Section 4.2.5.

**Definition 6 (Subject)** *A subject $s$ from the set of available subjects $S$ in an image is the grounding $\gamma_o$ of a bounding box $b$ in an image to an object $o \in S$ in a scene graph, which has a link to an entity in a knowledge base.*

In a tuple $\langle o, r, o' \rangle$, the grounding of the *subject* $s$ corresponds to the grounding $\gamma_o$, the grounding of the bounding box $b$ in the image to the object $o$ in the tuple $\langle o, r, o' \rangle$ of the scene graph. The *subject* $s$ of a triplet $\langle o, r, o' \rangle$ is always the entity $o$ performing something on another entity $o'$. The degree to which a *subject* is grounded in a scene graph is the unary potential.

**Definition 7 (Predicate)** *A predicate $p$ from the set of available predicates $P$ in an image is the grounding of the relation $r \in P$ between a bounding box $b$ of an object $o$ and a bounding box $b'$ of an object, $o'$ which has a link to an entity in a knowledge base.*

In a tuple $\langle o, r, o' \rangle$, the *predicate* $p$ corresponds to $\gamma_r$, the grounding of the relation $r$ in the tuple $\langle o, r, o' \rangle$. The *predicate* of a triplet $\langle o, r, o' \rangle$ describes the action performed by the *subject* on the object. The degree to which a *predicate* is grounded in a scene graph is the binary potential.

**Definition 8 (Object)** *An object $ob$ from the set of available objects $O$ in an image is the grounding $\gamma_{o'}$ of a bounding box $b'$ in an image to an object $o' \in O$ in a scene graph, which has a link to an entity in a knowledge base.*

In a tuple $\langle o, r, o' \rangle$, the grounding of the *object ob* corresponds to $\gamma_{o'}$, which represents the grounding of the bounding box $b'$ in the image to the object $o'$. The object $o$ of a triplet $\langle o, r, o' \rangle$ is always the entity on which something is performed. It is always the passive part of the relation between the *subject* and the *object*. The degree to which an *object* is grounded in a scene graph is the unary potential. The *object ob* can be a fact about the *subject* $s$.

In this section, it has been shown how facts are related to scene graphs and how they can be grounded in an image. How an experiment compares the quality of the facts generated by the scene graph could be designed as shown in Section 6.2.3 and Section 6.2.5.

### 4.2.3. Face Recognition

As described in Section 2.4 it is not trivial for machines to recognize faces. However, in 2014 DeepFace [78], a deep learning face recognition system, achieved an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) benchmark [79] where humans achieve an average accuracy of 97.53%. State-of-the-art face recognition methods are even better on the LFW benchmark, achieving a near-perfect result, as can be seen in Table 4.1.

| Method | Public. Time | Training Set | Accuracy % |
|---|---|---|---|
| Baidu [80] | 2015 | Baidu (1.2 M,18 K) | 99.77 |
| Range Loss [81] | 2016 | MS-Celeb-1 M, CASIA-WebFace (5 M,100 K) | 99.52 |
| CoCo loss [82] | 2017 | MS-Celeb-1 M (3 M,80 K) | 99.86 |
| Arcface [83] | 2018 | MS-Celeb-1 M (3.8 M,85 K) | 99.83 |

Table 4.1.: Comparison of Different Face Recognition Methods

In general, there are four major steps each deep face recognition model performs to recognize a face [23]: Face detection, face normalization/augmentation, feature computation, and feature comparison. In step one, the bounding box of a face is computed. The face detection step is a classification problem that can be solved by using machine learning like Faster-RCNN. The result of this step is a bounding box. Inside this bounding box, a face should be located. In step 2, faces are pre-processed to overcome problems of poses, illumination, expression, and occlusion since these factors can affect the performance of face recognition [84]. In [23] such preprocessing steps are categorized into one-to-many augmentation and many-to-one normalization. One-to-many augmentation means that the system generates many images of all poses a face could adopt from a single image. These generated images are used to learn the deep networks, a pose-invariant representation of the face. Many-to-one normalization recovers a canonical view of the face from one or many non-frontal images of the face, so the system can operate as if the image of the face was taken under controlled conditions. Step three can be thought of as a CNN-like architecture. However, there are other approaches as well, like multitask networks, where multiple tasks are solved at the same time. Still, commonalities are exploited across different tasks or multi-input networks, where convolutional and pooling layers are applied separately to each input. Step four uses a distance measurement like cosine distance or L2 distance to compute the similarity between a newly presented face image and the features known in the gallery for other face images already computed. Step four can have the purpose of face verification or face identification. Face identification computes the one-to-many similarity, resulting in a specific identity for the newly presented face. Face

verification computes the one-to-one similarity between the newly presented face and the face images already known.

The detected bounding box from the face detection will be used for combining face recognition and scene graph in the next Section 4.2.4. Hence, this method of detecting faces before recognizing them has the advantage that the position of the face is only computed once but can be used in the next steps. Since the person in the bounding box is already grounded in the scene graph, the face of the person can be considered grounded in the scene graph.

### 4.2.4. Combining Face Recognition and Scene Graph

This section proposes a method to combine the result of the scene graph procedure from Section 4.2.2 and the face recognition from Section 4.2.3. When combining the result of these two steps to produce an output as described in Section 4.2.7 two problems can occur: if there is no class for faces in the deep-learned classifier, the bounding boxes for the face from the face recognition and the bounding box of the class describing a person will be different in size in a position. E.g., if the class for the person is "woman", the bounding box for this class will describe the whole woman. Only a small percentage of the bounding box describing the woman will correspond to the face. The second problem is that even if the face class is present in the Faster R-CNN classifier, it is not guaranteed that the result of both bounding boxes for the face is equal. Hence, the problem is that in either case, the positional coordinates of the bounding boxes and the size of the bounding boxes can differ between the scene graph result and the face recognition result. As a result, it is not trivial to match the correct face to the correct person. To overcome this problem, the Jaccard coefficient Equation 4.5 can be used.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4.5}$$

The Jaccard coefficient is a statistical metric for the similarity of two sample sets. In this case, the sample sets are the pixels inside a bounding box. The Jaccard coefficient calculates the similarity of two areas by comparing the area of overlap of the two sample sets and the Area of Union of the two sample sets. If the two sample sets are not overlapping, the Jaccard coefficient is 0. If both sample sets represent the bounding box of the same face, the Jaccard coefficient would be 1 or near 1. The result of the latter example results from the accuracy of the used implementation for face recognition and scene graph generation. When using the Jaccard coefficient in this use case a threshold can be defined for the overlapping of the two bounding boxes or sample sets. If the Jaccard coefficient for these two bounding boxes is above this threshold, it is assumed that the face detected in the face recognition step belongs to the person detected in the scene graph step. To guarantee that a face of a person A is not wrongly attached to a person B if the two persons A and B are, e.g., cuddling, it is necessary to keep the highest Jaccard coefficient for each combination of face and person in memory. These entries need to be continuously updated if subsequent calculations yield higher Jaccard

coefficients. This approach helps prevent incorrect face-person associations and ensures more accurate results.

### 4.2.5. Entity Linking

Entity Linking is the process of attributing an entity with a link to a knowledge base. An entity can be a word in a sentence or a group of pixels representing an object in an image. The latter is the case in this thesis. A knowledge base should be a structured graph representing as much knowledge about the physical world as possible, e.g., Wikipedia, DBpedia, Wikidata, or ConceptNet. In this thesis, the entity to be linked will be referred to as an object in a scene graph or an object or bounding box in an image, and the linked entity in the knowledge base will be referred to as an entity.

There are different possibilities to link objects to a knowledge base. A feature-based entity linking could link objects by features computed from the pixel values representing the object to an entity in a knowledge base. This could be achieved by comparing the computed feature of an image available in the knowledge base against the computed feature of the image to be linked. Different methods which could be used are described in Section 3.3. However, these methods probably lack invariance due to the limited number of ground truth pictures in the knowledge base to be compared against.

Another approach is to generate an image caption for a sub-set of pixels in the image representing an object. From this subset of pixels, a caption could be generated. This caption could be linked via textual entity linking like [85] or similar. However, this approach has drawbacks, such as a lot of computing having to be done, and two possible fields of failure could arise: the image caption and the textual entity linking.

Other thinkable methods are the following. If there is already text in an image, OCR could be used, but this is not always true. In social media, user-added tags to an image could be used. However, like for OCR, this is not always the case. In Chapter 8, this thesis will present a method using CLIP [74] for image entity linking. A much simpler

| Linking Method | Precision % | Recall % |
|---|---|---|
| String2Article | 90 | 97 |
| String2Category | 100 | 27 |
| TagMe | 70 | 83 |
| Wikipedia index | 81 | 98 |

Table 4.2.: Precision and Recall for Different Simple Entity Linking Methods [52]

approach would be to take the objects from the object detection and link an article to the entity by string comparison of the class name in the object detection and the article's title in the knowledge base. Table 4.2 is a comparison of linking a class name to an article name and to a category name, the TagMe [85] System (an on-the-fly text annotation system) and the search via the Wikipedia index. Searching via the Wikipedia index corresponds to using the search function of Wikipedia and using the first result. This

table shows that string-wise comparison can deliver adequate results.

Having defined a method of entity linking, a *subject*, *predicate*, or *object* can be related to an entity in a knowledge base by entity linking. This process of entity linking is relating each *subject*, *predicate* or *object* to a fact because the entity in the knowledge base can be considered as a fact about the *subject*, *predicate* or *object* since the entity in the knowledge base is the object value of the triplet.

### 4.2.6. Metadata

The metadata of an image is data about the content of the image. It is not the pixel values themselves, but information about those values. This could be the file size, the time of creating the file or taking the picture, the image's resolution, the geo-coordinates where the image was taken, information about the camera, and much more. Such information is helpful since it can be regarded as facts about an image. This information can be compared even if the image is considered a duplicate as defined in Section 4.2.1. If the image is duplicated, the pixel values would be identical. However, the metadata could differ.

A suitable format for storing metadata would be the EXIF[2] format. This format contains information about the camera settings when taking a photo. While this is not useful for reproducing a photo on the internet, it is very useful for original photographs uploaded to Flickr or elsewhere on the internet.

Another possibility to store metadata of an image would be to use the Moving Picture Experts Group-7 (MPEG-7)[3] standard. However, this standard would be more useful if descriptors of image features had been used. Consequently, this standard could be used if, in Section 4.2.5, another method to link entities had been used.

For storing the metadata, a specific format should be used. This format should have the ability of an ontology to use the benefits of an ontology described in Section 4.2.7. It is beneficial if an already existing ontology or RDF is used to store the metadata. An already existing ontology or RDF would make the metadata more accessible. Metadata becomes a fact about the image itself because it is related to an image by the relations of the used ontology.

### 4.2.7. Overall Data Model

The data model should be an ontology that should be capable of combining and embedding the results of the previous steps:

1. detected objects and their position (object detection)
2. the relation between the detected objects (scene graph)
3. links of detected objects to knowledge bases (scene graph)

[2]https://www.cipa.jp/std/documents/e/DC-X008-Translation-2019-E.pdf
[3]https://www.iso.org/standard/34228.html

4. metadata about the image

The key requirements here are **F4**-**F8** and the non-functional requirement **Storage**. Following these requirements, the ability to query the results is achieved by the format of an ontology. By using an ontology, a result set can be transformed in a RDF[4] which

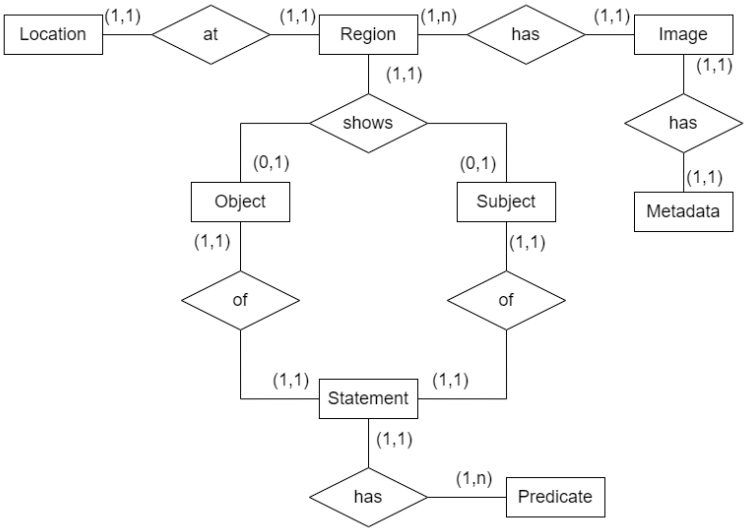



Figure 4.2.: Data Model

can be queried using SPARQL Protocol And RDF Query Language (SPARQL)[5]. A conceptual design for the data model can be seen in Figure 4.2. The image entity should hold a Uniform Resource Identifier (URI)[6] of the image. In an ideal world, the URI would always be a URL to reference the image in the World Wide Web (WWW). It is preferable not to be responsible for hosting and saving every searched and referenced image in its data store. If there are 200,000,000 websites (according to some websites this is the number of active websites in 2022) and every website would only host one image with a size of 1 Megabyte (MB), the required disk space to save these 200,000,000 images would be 2000 Terabyte (TB). Hosting 2000 TB of data in 2022 can cost an estimate between $19,000 and $300,000 per month on Azure[7] if the data should be accessible (the cost could be lowered to under $1000 a month if the images would be archived, which means inaccessible at any moment). Therefore, it is advisable not to store all images oneself.

Each image entity should have region entities that correspond to the bounding boxes of the scene graph object detection. The position and size of the bounding boxes are

[4]https://www.w3.org/TR/WD-rdf-syntax-971002/
[5]https://www.w3.org/TR/rdf-sparql-query/
[6]https://www.rfc-editor.org/rfc/rfc3986
[7]https://azure.microsoft.com/en-us/pricing/calculator/

saved in the location entity. Hence, each Region is at a Location and each Region must have a location entity.

A Region can show a *subject* and an *object*. The *subject* is the object $o$ from the scene graph, the relation between an object $o$ and an object $o'$ in the form of $\langle o, r, o \rangle'$. The *object* is the object $o'$ from the scene graph relation between an object and an object in the form of $\langle o, r, o \rangle'$. A statement is the produced relation between a *subject* and an *object* from the scene graph in the form of $\langle S, P, O \rangle$. This relation can hold one-to-many *predicates,* which are the *predicates* from the scene graph. This is necessary because a lot of scene graph implementations cannot only give one *predicates* as a result for a relationship but give a ranked list of all possible relations with a score.

Each entity *subject*, *object* and *predicates* can hold a reference to one or more knowledge bases and a score for the object detection or relation prediction at least. Each entity *subject*, *object* and *predicates* can hold a score.

Finally, each image node can contain a metadata entity that holds metadata about the image.

## 4.3. Description of Associated Process Flow

To make use of each step's advantages, it is necessary to perform the steps in a certain order. This section explains why this order is necessary. A diagram of the process flow is shown in Figure 4.3. The process flow starts when an image is received as input. After an image is received, the state of the image is determined. The state can be *original*, *duplicate*, or *replicate*. If the image is a duplicate, the determination of the state of the image is necessary to assure that no computational work will be done unnecessarily for the same image twice. Hence, if the image is a duplicate, the link of the image and the state are saved to the storage and the process ends. If the image is an original or replicated, the next step is performed. In the next step, a scene graph of the image is created. Next, entity linking for the human node of the scene graph will be performed. Entity linking means that the nodes of the scene graph are linked to their corresponding entities in a knowledge base. E.g., if the image shows dogs, each region representing a dog is linked to the article for a dog in the used knowledge graph. If a scene graph contains humans, the faces of the humans will be detected and recognized, which is the entity linking part. The result of the entity linking, which is a link to a knowledge base, will be added to the node for the detected human in the scene graph. If the image contains no humans, the entity linking will be performed without face recognition. In both cases: whether the image shows humans, entity linking will be performed. In other words, if the image shows humans, these humans should be recognized by face detection and face recognition, however, this step can be left out if the image does not show any humans. If face recognition has found faces to link the faces, they have to be combined with the correct region in the image. After the entity linking, the image's metadata is analyzed and added to the output. Finally, the output is put in the form of the overall data model and is saved.

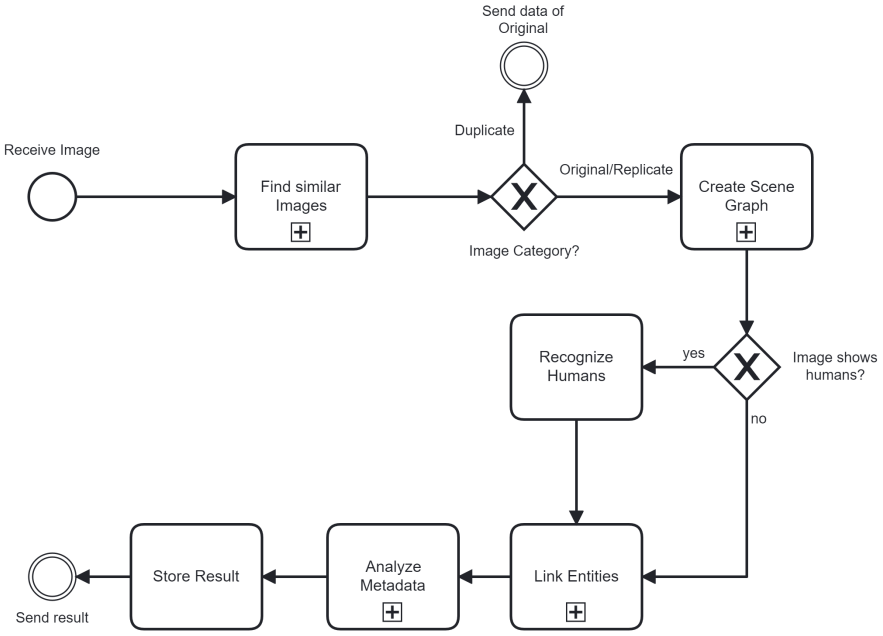



Figure 4.3.: Process Flow Overview

## 4.4. Summary

Thus far, the thesis has argued that facts are the object values in a triplet $\langle o, r, o' \rangle$. An image can be grounded in a scene graph. Hence, the object $o'$ of a scene graph can be a fact about the object $o$. The object $o$ and the object $o'$ have been defined as *subject* and *object* if they are grounded in the image and have a link to a knowledge base. The relation $r$ was defined as a *predicate* if it is grounded in the scene graph and has a link to a knowledge base. The link to the knowledge base is seen as fact about the *subject*, *object* or *predicate*. The link to the knowledge base is achieved by entity linking. The bounding box of a face can be seen as grounded in the scene graph transitively since the face's person is grounded to the image graph. It has been shown how the link between the bounding box of an object which is a person and a face can be computed. It was shown which facts could be computed and why they can be considered facts. It has been shown why the data model is implicitly creating facts by creating statements. A method of categorizing images into *original*, *duplicate* and *replicate* has been introduced and discussed, and the terms have been defined. An approach was presented to how to align the concepts in a process flow which would probably be the best fit.

# 5. Implementation

In this chapter, an implementation of the proposed design in Chapter 4 will be presented. It will be shown which technologies have been used to achieve the requested requirements from Section 4.1. A discussion will take place if the used technologies are the best, or if there would be room for improvement. In fact, there will be two implementations of the design. One implementation for Azure and one implementation for Jupyter Notebooks. Both implementations make use of the same class libraries. The properties and details of the used class libraries will be presented in the Jupyter Notebook section of this thesis. The execution process, some methods, and the communication between the services, or in the case of Jupyter Notebook, the notebooks, are different between the two implementations. At first, the Azure implementation was developed. The Azure implementation meets all the requirements. The Jupyter Notebooks implementation does not meet all the requirements. However, this implementation has other advantages which will be presented. The Jupyter Notebooks implementation does not meet all requirements because this implementation cannot scale and it has no message queue. However, the benefit of the Jupyter implementation is that the used techniques to achieve the required result, the output of each method and the result of each step can be displayed. A display in such a fashion could not be achieved by an Azure-based implementation. Jupyter Notebooks shine the most when presenting code that is responsible for a certain result. Showing code parts and their different impacts on the result works undisputed well with Jupyter Notebooks. A second benefit of Jupyter Notebooks is that debugging and testing can be done uncomplicated, and light weighted. With the implementation in Jupyter Notebooks there is no need to deploy artifacts on a server, have a working database and have a message queue up and running. In an environment where the non-functional requirements count, these negative properties of not having techniques to scale are a disadvantage. However, in a testing environment where flexibility and deployment speed are more impactful than scalability and execution time, these properties are crucial. The benefits of the Azure-based implementation are that it showcases that the design can be implemented and deployed efficiently. All the functional and non-functional requirements are met for the Azure-based implementation. Furthermore, the target fact-checking framework for which this prototype is developed is deployed on Azure. Hence, it was necessary to showcase that an implementation of the proposed design can be implemented and deployed on Azure. The whole implementation is available on the university's GitLab[1].

[1] https://git01lab.cs.univie.ac.at/thesis/2021ss/1209967-adrian-hofer

## 5.1. Libraries

This section describes the libraries implemented to be used by both implementations: the Jupyter Notebook implementation 5.3 and the Azure-based implementation 5.4. These libraries built the cornerstones of the functionality implemented.

### 5.1.1. Similiarty Library

This library is an implementation of the design and concepts described in Section 4.2.1. The library can be used with Python > 3.4. While development, different implementations of features to compute the approximate features were used from OpenCV. However, when using SIFT [76] features[2] and SURF [86] features[3] Python $\leq$ 3.4.0.14 only can be used. This usage restriction of SIFT and SURF features is due to a licensing issue. The implementations for the use of SIFT and SURF remain in the implementation but are not used anymore due to performance decrease when using these features. This performance decrease can be explained by the optimization of the implementation for the use with AKAZE [77] features[4].

The similarity-features library can be used to compute the exact SHA-1 features with Python's hash lib[5] and the approximate AKAZE key points and descriptors of an image. The AKAZE OpenCV implementation, which is used to compute the key points and descriptors, utilizes the parameters described in Table 5.1. These parameters have been defined by examination. Different stages of the examination will be presented in Section 6.2.1 and in Table 6.2. From the descriptors of AKAZE features, the approximate

| Parameter | Value | Description |
|---|---|---|
| descriptor_type | MLDB | Modified Local Difference Binary |
| descriptor_size | 64 | Size of the descriptor in bits |
| descriptor_channels | 3 | Number of channels in the descriptor |
| threshold | 1e-12 | Detector response threshold to accept point |
| nOctaves | 4 | Maximum octave evolution of the image |
| nOctavesLayers | 2 | Default number of sublevels per scale level |
| diffusity | PmG2 | |

Table 5.1.: AKAZE Parameter

features are computed. This is done by normalizing the descriptors by the Frobnius norm given by [87] in Equation 5.1.

$$\| A \|_F = \left[ \sum_{i,j} abs(a_{i,j})^2 \right]^{1/2} \tag{5.1}$$

[2] https://docs.opencv.org/3.4/d7/d60/classcv_1_1SIFT.html
[3] https://docs.opencv.org/3.4/d5/df7/classcv_1_1xfeatures2d_1_1SURF.html
[4] https://docs.opencv.org/3.4/d8/d30/classcv_1_1AKAZE.html
[5] https://docs.python.org/3/library/hashlib.html

The normalization in Equation 5.1 is done to create a comparable scalar from the descriptor. The Frobnius norm was used because it can be seen as the Euclidean norm applied to the vectorized version of a matrix.

The descriptors of images are matched by k-Nearest-Neighbor. The used implementation for descriptors[6] of OpenCV returns the k ($k = 2$) best matches for each descriptor set ordered by distance. The comparison of the features is executed by the algorithm shown

---

For each image $IC$ in the list of all images and the input image $I$, compare $IC$ to $I$:

- Do not compare images of the same image-id.
- Compare the normalized approximate feature of image $I$ to the normalized approximate feature of image $IC$.
- If $AC$ is the approximate feature of image $IC$ and $A$ is the approximate feature of image $I$ and $T$ is a threshold, the two images are candidates for a duplicate or replicate if: $A - T < AC < A + T$ or $AC - T < A < AC + T$.
- If the exact feature of image $I$ is equivalent to the exact feature of image $IC$ image $I$ is a **duplicate**.
- If image $I$ is no duplicate, do a k-Nearest-Neighbor matching of the two image's exact descriptors.
- If $ld$ is the lower distance between the two matches and $hd$ is the higher distance of the two matches and $p$ is a configurable percentage between 0 and 1: The key point's descriptor is defined as a found match if $ld < p * hd$.
- Get the fraction of matched key points compared to the total amount of key points.
- If this fraction is within a certain threshold, the image is marked as **replicate**.
- If all images are compared and the image is neither a duplicate nor a replicate, the image is an **original**.

---

Table 5.2.: Similarity Feature Comparison

in Table 5.2. Important parameters are the following: The threshold for approximate features is called the approximate threshold. This threshold is used while constructing the AKAZE features with OpenCV. The approximate threshold is responsible for the detector[7] to accept a point. The threshold T in $A - T < AC < A + T$ is called threshold. The configurable percentage $p$, ranging between 0 and 1 in the inequation $ld < p * hd$, is employed to determine the proportion of the higher distance that should not exceed the lower distance. This introduces variability to the distance between the lower match

[6] https://docs.opencv.org/3.4/db/d39/classcv_1_1DescriptorMatcher.html
[7] https://docs.opencv.org/3.4/de/db6/classcv_1_1Detector.html

and the higher match in the k-Nearest-Neighbor algorithm. By adjusting this percentage, the number of k-Nearest-Neighbor matches included in the matched key points can be controlled. Only matches with a lower distance smaller than the specified percentage of the higher distance are considered. A lower percentage leads to the exclusion of key points that are in close proximity. This is important as these features should be approximate rather than exact, and including key points that are too close would be counterproductive.

The inlier ratio threshold refers to the fraction of matched key points compared to the total number of key points. If this inlier ratio threshold falls within a configurable threshold, the images are considered as matched.

### 5.1.2. Scene Graph Library

The scene graph library uses the implementation of [19] available on GitHub[8]. This is a Python reimplementation of different scene graph algorithms for benchmarking. Implementations include [19], [11], [15], [16]. Since it is a reimplementation, it uses an older PyTorch library[9]. For some dependencies (e.g., Pillow[10]) of PyTorch, the reimplementation needs to use Python 3.6. The learned models were taken from the scene graph RCNN's GitHub page.

To apply a model, it is necessary to have a Cuda compatible graphics card installed. The used Cuda version needs to be compatible with the used PyTorch version 1.0.2. Hence, for this implementation, the Cuda versions 9.0 and 10.0 were used. Some newer graphics cards use a different architecture and are therefore not compatible anymore with these older Cuda versions. By applying the model to an image, two outputs are generated, one with predictions for object detection by Faster R-CNN and one with predictions for relations and predicates between the objects. A configuration for the number of top relations to be added to the resulting graph has been added to the implementation. This parameter is called $\text{top}_k$. If $\text{top}_k$ is set to one, only the best scoring relation is added to the output. If $\text{top}_k$ is set to 20, the 20 best scoring relations are added to the output. Another configuration for the threshold of object scores has been added as well. This parameter is called threshold and can be set between 0 and 1. The parameter controls which objects are added to the output. E.g. if the threshold is set to 0.6 only object detections equal to or higher than 0.6 are added to the output.

The Visual Genome data set to test the model is taken from [17] and can be found on the Visual Genome homepage[11]. The Visual Genome data set is prepared by [11] and their data tools[12]. Using these data tools results in a benchmarking data set from Visual Genome, shown in Table 5.3. The table shows the types of annotations, the number of objects and the number of predicates.

[8] https://github.com/jwyang/graph-rcnn.pytorch
[9] https://pytorch.org/get-started/previous-versions/
[10] https://python-pillow.org/
[11] https://Visual Genome.org/
[12] https://github.com/danfeiX/scene-graph-TF-release/tree/master/data_tools

| Annotations | Object | Predicate |
|---|---|---|
| Categories | 150 | 50 |

Table 5.3.: Visual Genome Benchmarking Data Set [19]

The scene graph library implements objects and relations of the data model presented in 4.2.7. This implementation is created with the use of RDF lib[13] in version 4.2.2. The same version restrictions mentioned earlier for the scene graph reimplementation apply here for the implementation of the used data model because they are implemented in the same project. RDF lib can parse and serialize RDF/Extensible Markup Language (XML), N3, N Triples, N-Quads, Turtle, TriX, RDFa and Microdata as well as query the data by SPARQL. The implementation also uses the RDF lib to create describers

| Binding | Namespace |
|---|---|
| foaf | FOAF[14] |
| rdf | RDF[15] |
| rdfs | RDFS[16] |
| sdo | SDO[17] |
| img | IMAGE[18] |
| spa | SPATIAL[19] |
| | STATEMENT |

Table 5.4.: RDF Graph Bindings

for the generated classes. Describers prefix the namespace of an XML-Node. The used describers are shown in Table 5.4. The IMAGE and SPATIAL namespaces are adapted to reimplementation with an extended number of nodes. The STATEMENT namespace was created for this thesis. The output of the scene graph library is formatted as JSON-LD[20] generated by RDF lib. JSON-LD is a lightweight-linked data format based on JSON. JSON-LD was chosen as the preferred output format because FactCheck uses JSON-LD for formatting data about facts. Since the scene graph can be seen as a collection of facts about an image, the output of the scene graph is formatted as JSON-LD.

### 5.1.3. Entity Linking Library

The entity linking library makes use of the entity linking API 5.2 and the Azure face recognition service. The implementation uses SPARQL to query the RDF graph deserialized from the JSON-LD for regions. SPARQL stands for SPARQL Protocol and RDF Query Language and is a query language for RDF which is a semantic query language for databases and can retrieve and manipulate data stored in RDF. Since 2018, SPARQL

[13]https://rdflib.readthedocs.io/en/4.2.2/
[20]https://json-ld.org/

is an official W3C recommendation[21].

The representation node for the image in the graph database is seen as the parent entity for all regions, as can be understood from Figure 4.2. All regions are queried by SPARQL and sent to the entity linking API to add links to knowledge bases to the nodes for the objects, subjects, and predicates of the graph. For all links to knowledge bases, the RDF-Schema property "see also"[22] is used. The "see also" property should be [89] "*used to indicate a resource that might provide additional information about the subject resource*".

Metadata is added as facts about the image to the image's node's metadata node. Metadata is extracted with exifread[23] and saved to the graph with EXIF tags from Pillow. Not all metadata is EXIF metadata if extracted with Pillow. The non-EXIF metadata includes: Filename, Image Size, Image Height, Image Width, Image Format, Image Mode, Image is Animated, Frames in Image. For EXIF metadata, everything available by EXIF is used except image thumbnails because of storage capacity restrictions, the filename because the filename is already extracted by Pillow and the EXIF maker notes because they could be subjective.

Regions that show humans are queried by their own SPARQL query. In these regions' bounding boxes, faces are detected and identified by the Azure face service. If faces are found, these found faces' bounding boxes are compared against the bounding boxes of humans in the image. If the intersection over union is the highest for a combination of detected human and face, it is assumed that this face belongs to this human. When intersecting the two bounding boxes, the metadata of the image is used to compute a factor for the width and the height of the image to be able to guarantee that the requirements of the image's resolution for the scene graph and the Azure face detection do not interfere with each other. When faces are found which belong to a detected human object, they are sent to the entity linking API to link them to the found person's ID from the Azure face recognition service with the corresponding article in the knowledge base. The Azure face recognition service is trained by a self-created subset of the Pub Fig data set [90] which is a subset of 150 celebrities of the larger Pub Fig data set. Details of the data set will be presented in Section 6.1.3.

## 5.2. Entity Linking API

The entity linking is decoupled from of thesis repository because this entity-linking API should work as a single entity linking API for all services which need entity linking in the FactCheck ecosystem. The API is set up as a Flask app[24] using blueprints for the modules. A module is the highest level of resource supplied by the API. Each service for which the entity linking API provides entity linking obtains its module.

To link objects in images to knowledge bases, a static index structure file was designed

[21] https://www.w3.org/TR/sparql11-overview/
[22] http://www.w3.org/2000/01/rdf-schema#seeAlso
[23] https://github.com/ianare/exif-py
[24] https://flask.palletsprojects.com/en/2.2.x/

for this thesis. This index structure file contains all 150 objects the Faster-RCNN is trained for and all 50 predicates the scene graph is trained for. Each of the 150 objects has a link to Wikidata and Concept.Net, and each of the 50 predicates has a link to Wikidata and Concept.NET. The relation between the link and the object is a key value relation, where the value is the link and the key is the object's name. The link is represented in the form of a URL. The object's name is represented as a string. To reduce look-up time, there are all potential key values pairs already saved in the index structure. An example of the static index structure file in JSON can be found in the Listing A.1.

To link the faces of individuals in the collection of the Azure Face API, the person ID of a person from Azure is used. The person ID in Listing A.2 is the key of the parent JSON object. The link to the Wikipedia article is looked up by the person ID and added to the node for the detected object on the same level as the links for the more general category of the node for the detected object. The whole implementation is available on the university's GitLab[25].

## 5.3. Jupyter Notebook Implementation

Jupyter Notebook[26] are easy to deploy Python script collections where each snippet of code, which is called cells, can be executed single-handedly. In this thesis, three Jupyter Notebooks have been implemented to showcase a possible implementation of the design. One implementation of the similarity design described in Section 4.2.1, one implementation for the scene graph described in Section 4.2.2 and one implementation for entity linking which realizes the design and concepts presented in Sections 4.2.3, 4.2.4, 4.2.5, and 4.2.6.

### 5.3.1. Similarity Jupyter Notebook

This Jupyter Notebook is an implementation of the design and concepts described in Section 4.2.1 using the library from Section 5.1.1. The Jupyter Notebook works with local memory. The input files are on disk and the results from each step are kept in memory. Input files come in the form of image files.

There is a cache Python class based on Python dictionaries that works as a cache for the results. The caches can be made persistent with the use of pickle[27], an object serialization tool for Python. Pickle is used to serialize the cache objects. These serialized cache objects can be saved to disk, loaded from disk and deserialized to cache objects again. After the features of an image have been compared, the state of the image is updated in the cache.

The whole purpose of the Jupyter Notebook is to showcase a typical use of the similarity-features library.

[25] https://git01lab.cs.univie.ac.at/a1305573/entitylinking
[26] https://jupyter.org/
[27] https://docs.python.org/3/library/pickle.html

### 5.3.2. Scene Graph Jupyter Notebook

This implementation makes use of the scene graph library described in Section 5.1.2. The Jupyter Notebook shows how the model is configured and built. The configuration is a mixture of Python variables and YAML Ain't Markup Language (YAML) configuration files. How the model can be built with this configuration is shown in the Build Model cell. After the model has been built, images can be passed to the model to be parsed for scene graphs. Before an image can be parsed, it needs to align with the requirements of the model. These requirements are presented in the Prepare Image cells. These requirements are:

- The image is resized to a fixed size of 1024×768 pixels.
- The color space is converted from Red Green Blue (RGB) to Blue Green Red (BGR) because OpenCV is using BGR.
- The image is normalized to change the range of the pixel intensity values to values more similar to the trained pixel intensity values.
- Convert image storage ordered from channel height width to height width channel.
- Convert the image from a NumPy array to a torch tensor.

The output of the scene graph model can be transformed by visualizing functions that were created for this thesis. An example of the scene graph visualization can be seen in Figure 5.1. The visualization contains a legend with the scores of the detected objects by Faster R-CNN in different colors and scores of the relation between objects in red. The object nodes in the graph have the same color as the score in the legend. The edges

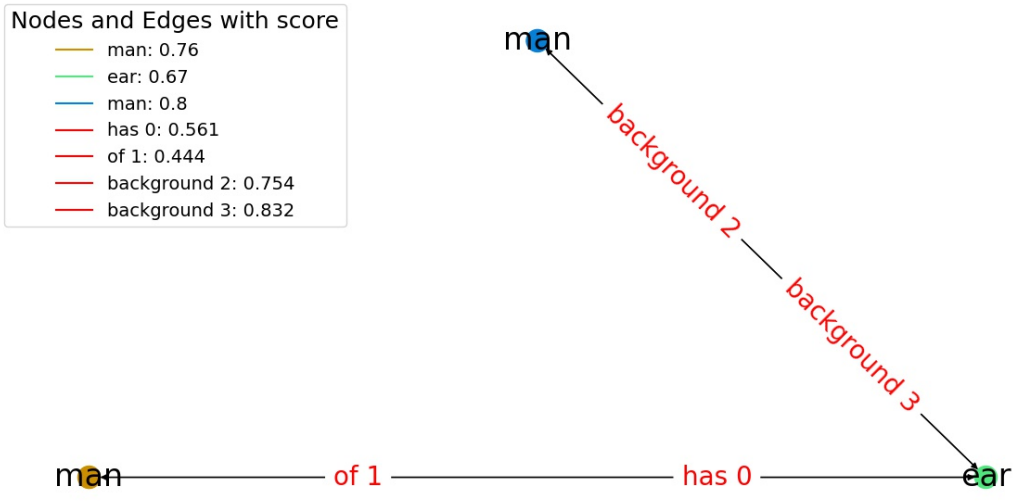



Figure 5.1.: Scene Graph with Objects as Nodes and Relations as Edges

between the nodes that reassemble the relation have the same name as the relation with the score in the legend. The property which is closer to the arrow of an edge is the label of this direction of the edge. In Figure 5.2 a visualization of the object detection of an image applied to the scene graph model can be seen. The same colors as for the nodes are used for the corresponding objects in the image. For the visualization, NetworkX[28] is used in version 2.5.1. This version has a drawback to newer releases when creating a layout for an image of a graph because some newer layouts cannot be used that would generate graphs which have fewer overlapping edges. However, because of the PyTorch restriction to version 1.0.2, no newer version of Python can be used and hence no newer version of NetworkX can be used.

The last cell shows how the scene graph library can be used to generate an output as JSON-LD. To make the complexity of the data model understandable, a visualization as depicted in Figure 5.3 has been developed using RDF lib.


Figure 5.2.: Object Detection in an Image

### 5.3.3. Entity Linking Jupyter Notebook

This notebook shows a typical use of the linked data generated by the scene graph and entity linking API. The implementation makes use of the entity linking library described in Section 5.1.3. RDF lib is used with SPARQL to query the graph for linked results. Examples of nodes are printed before and after the linking.

The entity linking Jupyter Notebook shows as well how faces and their detected objects are linked with the use of the Jaccard Coefficient and visualizes the result of matching faces to detected objects. An example of such a match is Figure 5.4. A detailed analysis of different cases is presented in Section 6.2.5. Therefore, the Figures 5.1 - 5.4 will be reused in Section 6.2.5.

[28] https://networkx.org/

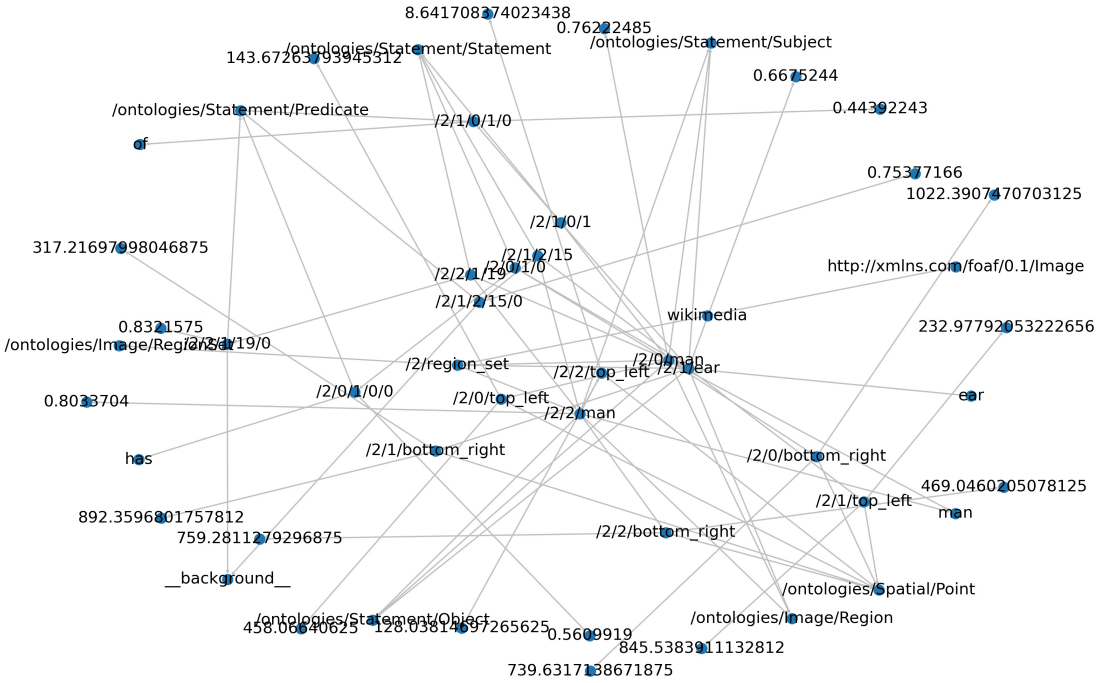



Figure 5.3.: Graph of JSON-LD


Figure 5.4.: Linked Faces

## 5.4. Azure-based Implementation

### 5.4.1. System Overview

The system overview of the Azure implementation is depicted in Figure 5.5. A User or a crawler creates a request to the API which acts as the management service. The API is responsible for handling the process flow described in Section 4.3. The similarity service is responsible for categorizing the image into *original*, *duplicate* and *replicate*, as described in Section 4.2.1. The scene graph service is responsible for creating a scene graph described in Section 4.2.2. The entity linking API is a restful API to link detected objects to a knowledge base. This entity linking API should bundle all entity linking for the FactCheck framework in one place. This process has already been started since the implantation of image entity linking and text entity linking is bundled in this API. The text entity linking is implemented as the process of another thesis. The implementation of the entity linking API is described in Section 5.2. The face API is the Azure face recognition API described in Section 2.4.2 and designed in Section 4.2.3. The

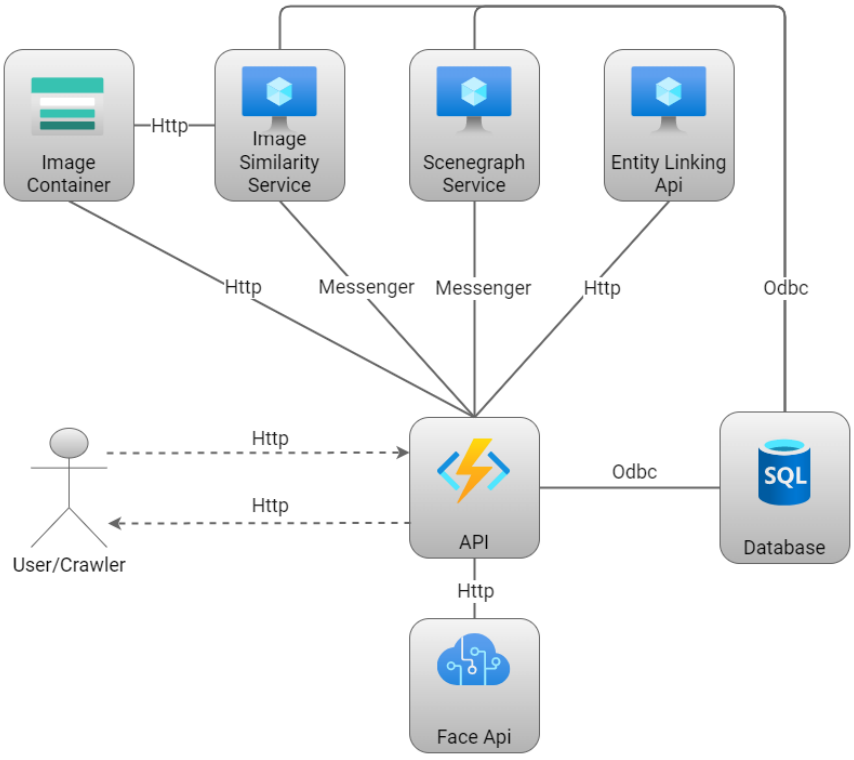



Figure 5.5.: Azure Architecture

communication between the user or crawler and the API and the communication between the API and all other services will be described in Section 5.4.4. The image container, which is a data lake and the database, which is a SQL-database will be described in

Section 5.4.6. To be able to use the same functions for storage access, message queue management and database access across the Azure implementation, a Python library[29] was developed and uploaded to PyPI's test environment.

### 5.4.2. Similarity Azure Implementation

This implementation makes use of the similarity library described in Section 5.1.1. The process flow of the implementation is described in Figure 5.6. After an image Data Transfer Object (DTO) is received, the image is loaded from the URL. The exact SHA-1 features are computed and the approximate AKAZE features are computed. These features are saved to an Azure Storage described in Section 5.4.6. All exact features' storage paths and all approximate features' ones are loaded from the database described in Section 5.4.6. Next, a loop through the tuples of loaded exact and approximate features starts. At each iteration of the loop, the current exact feature and the exact feature of the image computed before are compared. If the two features are equal, the image is considered a duplicate. If not, the current approximate feature and the

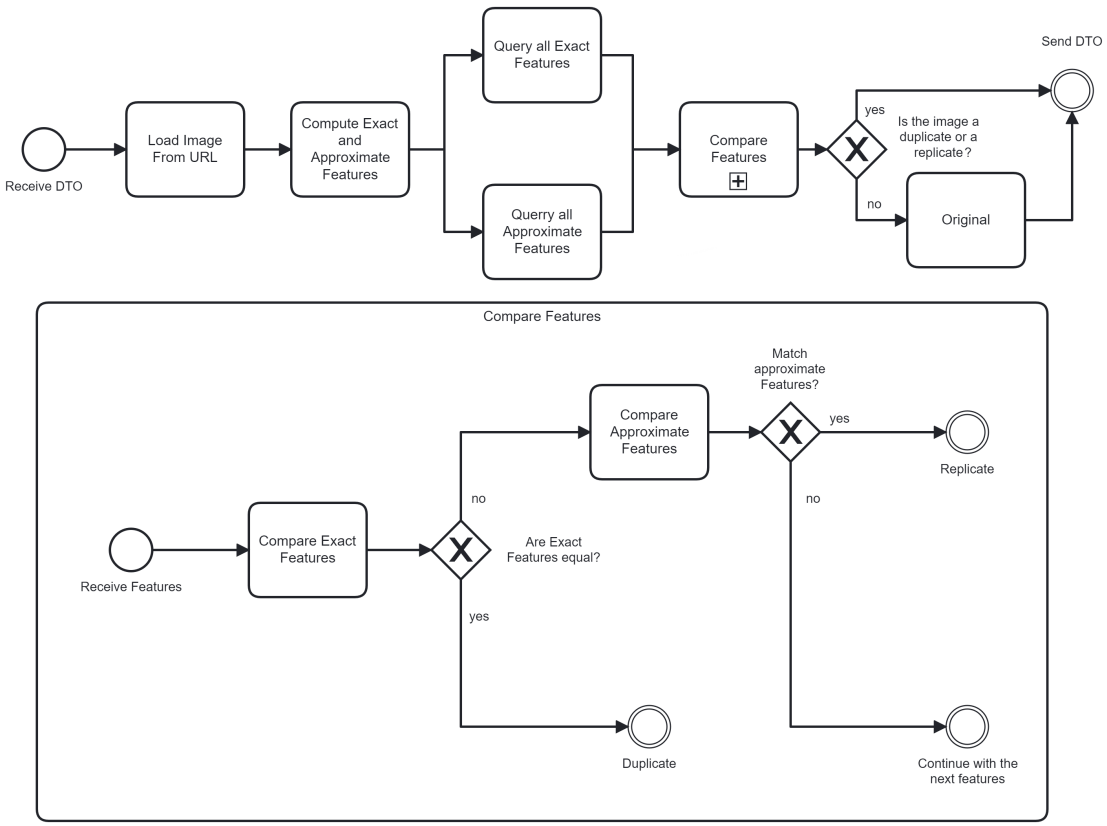



Figure 5.6.: Similarity Flow

[29] https://test.pypi.org/simple/imagefact-azure-lib/

approximate feature of the image computed before are compared. If they match, the image is considered a replicate. If the loop terminates without the image being considered a replicate or a duplicate, the image is considered an original. This implementation uses threading[30] to process multiple images in parallel. The number of threads grows dynamically to a configurable number of maximal threads.

### 5.4.3. Scene Graph Azure Implementation

This implementation makes use of the scene graph library described in Section 5.1.2. The main script was extended by a *run* function, which can run the scene graph service as a message receiver. The process of generating a scene graph is triggered by receiving a message from the message queue described in Section 5.4.4. After a DTO is received from the message queue, the image is downloaded from the image's URL. The same requirements for the image as described in Section 5.3.2 apply to this implementation. The scene graph's reimplementation model script mentioned in Section 5.1.2 was extended by a *run* function that can apply the model on an image and return the scene graph's data model serialized as JSON-LD. If configured, the *run* function can visualize the scene graph and the detection like the Jupyter Notebook. Since the deployment of the scene graph services requires a Virtual Machine (VM) equipped with an NVIDIA graphics card, a detailed description of how to deploy this service on Azure was created, and it was showcased that it is possible to deploy and use this service on Azure.

### 5.4.4. Azure API and Management

This is an Azure Functions[31] implementation of an API that manages the communication between the similarity service, scene graph service and the entity linking API. Azure Functions were chosen because they can take Hyper Text Transfer Protocol (HTTP) requests as inputs and produce messages on a message queue as outputs. Another reason for choosing Azure Functions was because they can scale and are cheaper to use than Azure's API Management. The entity linking library described in Section 5.1.3 resides in the management service to preprocess the data before sending the data as DTOs to the entity linking API. There are three functions implemented with Azure Functions: one which receives a DTO and sends a similarity ingress DTO in a message on the Azure Service Bus to the similarity service, one which receives a similarity egress DTO from the Azure Service Bus and sends a scene graph ingress DTO in a message on the Azure Service Bus and one which receives a scene graph egress DTO from the Azure Service Bus. The last function preprocesses the content of the scene graph egress DTO to be able to communicate with the entity linking API to add links to knowledge bases to the output, which is a JSON-LD.

[30] https://docs.python.org/3/library/threading.html
[31] https://azure.microsoft.com/de-de/products/functions/

### 5.4.5. Azure Message Queue

The message queue in the system utilizes the Azure Service Bus[32], which enables the sending and receiving of messages between different services. Each service within the system subscribes to two message buses: one for receiving incoming messages (ingress) and another for sending outgoing messages (egress).

The message buses can be configured to handle failed messages in different ways. They can be set up to automatically resend failed messages, giving them another opportunity for successful delivery. Additionally, if a message cannot be delivered even after retries, it can be placed in an error message queue for further analysis and troubleshooting. This allows for proper handling of failed messages and helps ensure reliable communication between services in the system.

### 5.4.6. Azure Storage and Database

To store images, an Azure Blob Storage[33] was used. Blob Storage was used because they are optimized for data lakes, which can be used to analyze the data which is stored in the lake. The data in the storage blob can be switched between hot, cold, and archived to minimize the cost or optimize the retrieval time of the data. To store relational data, the Azure Structured Query Language (SQL) Database[34] on an Azure SQL Server was used. The Azure SQL Database was used because it is a fully managed database engine with automated updates, provisioning, and backups. The database may scale if needed.

### 5.4.7. Azure Deployment

For the automatic deployment of all Azure services which are needed (Azure Service Bus, Azure SQL Database, Azure Storage Container) an Azure Resource Manager (ARM) template was developed. This ARM template not only can automatically deploy the Azure services but can configure all self deployed services (similarity service, scene graph service, Azure Functions API) with the used connection string for the Azure SQL Database and the connection string for the Azure Service Bus.

## 5.5. Summary

This chapter has described the implementation of the proposed design in Jupyter Notebooks and on Azure. It was shown which libraries are shared by the implementation. Reasons for the different implementations were given. First, Jupyter Notebooks showcase the execution of the pipeline and build a potential environment for students in this field to work on. Second, the Azure implementation showcases the use of the pipeline in a cloud environment. The Azure implementation is relevant for using this approach in the FactCheck framework and on large scale. It was discussed why certain technologies have

[32] https://azure.microsoft.com/en-us/products/service-bus
[33] https://azure.microsoft.com/de-de/products/storage/blobs/
[34] https://azure.microsoft.com/en-us/products/azure-sql/database/

been used, e.g., as a result of dependencies. If there were certain methods used, e.g., for similarity measurement's feature normalization, it was discussed how these methods work.

# 6. Experimental Analysis

## 6.1. Data Sets

In this section, the used public available datasets are described. There are some other datasets that were generated by rotating and masking an image collection. The Python scripts and the original collection of these images can be found in the GitLab repository.

### 6.1.1. Visual Genome

The Visual Genome data set [17] is a collection of annotated images where the objects in an image are annotated. The relation between the two objects is annotated if the relation makes sense (e.g., $\langle Woman, throwing, Frisbee \rangle$). The dataset includes a total of 108,077 Images with 5.4 million region descriptions, 3.8 million object instances, 2.8 million attributes and 2.3 million relationships. Region descriptions are descriptions of the whole image or certain regions in an image. Attributes are properties of an object instance, e.g., an elephant is large. However, region descriptions and attributes were not used in this thesis. The annotations were produced by humans using Amazon Mechanical Turk. The annotation process was designed in a way that each worker got paid $6-$8 per hour, which aligned the process with the ethical work standard for human intelligence presented by [91]. The precision and recall in Table 6.1 were computed by comparing a subset of the results (N=200) from Amazon Mechanical Turk with self-annotated images by the research group. A subset of the Visual Genome dataset was used to train the scene graph model. This subset contained 150 cases for objects and 50 predicates. Each of these classes and predicates was annotated for this thesis with a link to Wikidata and a link to concept.net. The full annotation can be regarded in Listing A.1

| | Precision % | Recall % |
|---|---|---|
| Objects | 88.0 | 98.5 |
| Attributes | 85.7 | 95.9 |
| Relationships | 92.9 | 88.5 |

Table 6.1.: Precision and Recall in Visual Genome [17]

### 6.1.2. Original and Tampered Image Data Set

The used data set of original and tampered images is an amended version of [92] available on Kaggle[1]. The original dataset consists of 100 original images, 1050 tampered images and the corresponding masks of the images. The original images are real photos captured indoors and outdoors in different places. The tampered images are created by using Adobe Photoshop, using a natural effect not visible to human eyes. An example is Figure 4.1. The amended Kaggle version of the dataset consists of 750 original images and 730 tampered images. The higher number of original images was produced by cropping the 100 images into smaller images showing only a certain area of the image. These smaller areas always have an original and tampered version. In other words, the cropped areas are chosen in a fashion that there is always an original and a tampered slice.

### 6.1.3. PubFig: Public Figures Face Database

The PubFig: Public Figures Face Database [90] is a dataset containing 58,797 images of 200 people collected from the internet. The images are taken in uncontrolled situations. Hence, there is a wide variety of poses, lighting, expression, scene, camera etc. In contrast to the LFW [79] data set, there is a fewer number of different humans, which is an advantage to this thesis. A subset of 150 people with 1,645 images in the training data set and 10,302 images in the test data set are available. Each of the 150 people is a so-called celebrity. In the folder for the images a JSON file is stored with a link to the celebrities' Wikipedia article. This annotation could be used when the dataset would be used for further development with CLIP. The advantage of having a fewer number of humans was that annotating the learned persisted face IDs from Azure with links from Wikipedia like in Listing A.2 was a good mean value between having a sufficient number of test cases and being capable of finishing the annotation in an acceptable time.

## 6.2. Experiments

For transparency reasons, there are Jupyter Notebooks for all experiments in the GitHub repository. The notebook can take a JSON file as an input parameter. In the JSON file the settings for the current experimental setup can be defined (image ID, $\text{top}_k$, threshold). A list of images where the position of the image corresponds to the image ID given earlier can be defined and a ground truth for the list of images can be defined. The JSON file for the setup of the experiments in Section 6.2.3 can be found in Listing A.3.

### 6.2.1. Similarity Experiments

There are two experimental setups to be presented. First, the experimental steps to define the used AKAZE parameters will be shown. Second, a comparison of the used baseline against the findings of this thesis will be presented.

[1] https://www.kaggle.com/datasets/saurabhshahane/cg1050

| Parameter | #1 | #2 | #3 | #4 | #5 | #6 |
|---|---|---|---|---|---|---|
| approx threshold | 1.00E-10 | 1.00E-10 | 1.00E-10 | 0.0001 | 1.00E-12 | 1.00E-10 |
| inlier ratio | 0.3 | 0.23 | 0.28 | 0.28 | 0.28 | 0.28 |
| nn match ratio | 0.84 | 0.84 | 0.84 | 0.87 | 0.87 | 0.87 |
| Original % | 94.17 | 92.50 | 92.50 | 87.50 | 92.50 | 92.50 |
| Duplicate % | 94.17 | 92.50 | 92.50 | 87.50 | 92.50 | 92.50 |
| Repliacte % | 75.83 | 85.00 | 80.83 | 87.50 | 89.17 | 89.17 |
| Time[s] | 859.09 | 930.31 | 951.29 | 1464.60 | 908.34 | 829.53 |

Table 6.2.: Iterations #1 to #6 of AKAZE Parameter Optimization N=120

The parameters of the similarity algorithm which were optimized are the following: threshold, approximate threshold, inlier ratio, k-nn match ratio. The impact of these parameters in Table 5.2 are discussed in Section 5.1.1. The parameters to construct an AKAZE feature are descriptor channels, threshold, nOctaves, nOctavesLayers and descriptor n size. How these parameters work is explained in Section 5.1.1. All parameters besides the threshold were optimized before the following experiments, since their impact on the performance of the similarity algorithm could be clearly identified from the documentation of AKAZE and a few experiments. The used data set for the experiments shown in Table 6.2 is a subset of the tampered image collection presented in Section 6.1.2. The results for the row **original %** describes how many of the original images used as input for the algorithm were detected as original images. The images not identified as original were detected as duplicate or replicate. The goal of this optimization was to find a setting for parameters. The algorithm used with these parameters should return a result where above 90% of original and duplicate images are detected as the correct state. A maximum defined as close to 90% of replicates are detected as replicates. The time for processing all images should be minimal. The first three iterations in Table 6.2 show that the inlier ratio threshold is impacting the replicate result, while the duplicate result is nearly kept the same. Furthermore, the table shows that a high approximate threshold slows down the algorithm and is not optimizing the results, as shown in Iteration #4. However, too low approximate a threshold is slower than a slightly higher approximate threshold, as can be observed when comparing Iteration #5 and #6.

The second set of experiments was the comparison of the findings of this thesis against a baseline. The baseline was taken from [38] where the original algorithm was published first. Since the used dataset in [38] was not referenced directly but described, the goal was to create a similar dataset. Creating a similar dataset could be achieved by using Python scripts to mask images with the same masks described in [38]: Fish eye distorted, salt and pepper, JPEG comprised, 45 degrees rotated and 135 degrees rotated. Intensity change was replaced with tampered images because tampering an image seemed like the more realistic use case. The parameters used for the findings in Table 6.4 are the parameters from Table 6.2 iteration #6. The findings from Table 6.3 differ from Table 6.4 in several important ways. The difference between the results of the baseline and the

| Mod. Type | Intensity change | Fish eye distorted | Salt and pepper | JPEG | Rotation 45 | Rotation 135 |
|---|---|---|---|---|---|---|
| SURF (avg ratio %) | 33 | 16 | 64 | 32 | 31 | 30 |
| Detection accuracy % (threshold = 30) | 63 | 36 | 100 | 72 | 81 | 40 |
| Detection accuracy % (threshold = 20) | 63 | 66 | 100 | 72 | 100 | 90 |

Table 6.3.: Similarity Measurement Baseline [38]

| Parameter | JPEG | Fish eye distorted | Salt and pepper | Rotation 45 | Rotation 135 | Tampered |
|---|---|---|---|---|---|---|
| N | 100 | 100 | 100 | 100 | 100 | 120 |
| Original % | 100 | 100 | 100 | 100 | 100 | 92.50 |
| Duplicate % | 100 | 100 | 100 | 100 | 100 | 92.50 |
| Replicate % | 100 | 99 | 89 | 100 | 100 | 89.17 |
| Time[s] | 206.58 | 220.14 | 258.76 | 249.42 | 246.10 | 829.53 |

Table 6.4.: Similarity Measurement Results

findings of this thesis shows that a correct use of AKAZE features can result in a better performance of Table 5.2 than the implementation presented in [38].

### 6.2.2. Scene Graph Experiments

This section describes a comparison between the findings of the scene graph paper and the findings of this thesis. For comparing the performance of the scene graph implementation, the same metric as in [19] was used. This metric shown in Equation 6.1 is a modified recall which takes the count of found object nodes $C(O)$, predicates $C(P)$, and triples $C(T)$ into account and compares the sum of those against the total number $N$ of objects, predicates, and triplets in the ground truth. Since this metric takes three parameters where six possible points of failure could occur, this metric is harder to beat than a metric where only one possible point of failure can occur.

$$RecallSGPlus = \frac{C(O) + C(P) + C(T)}{N} \tag{6.1}$$

The model is trained and compared against a split data set from Visual Genome [17] where the training data makes 80% of the samples and the test data makes 20% of the samples. The difference between the baseline in Table 6.5 and Table 6.6 can probably be explained by the difference in hardware performance between the baseline and the findings of this thesis. The rows are different scene graph detection algorithms. The

columns are the recall results for these algorithms where a specific recall metric is used. This recall metric RecallSGPlus@k describes the percentage of the correct detected scene graphs in the top-k recommendations in percent presented in Equation 6.1. For this thesis, an NVIDIA 1060 with 6 MB memory was used. The author of [19] states that he used 8 Graphics Processing Units (GPU)s. Another explanation for this gap could be the different-used version of PyTorch. In contrast to the baseline where PyTorch 1.0.0 was used, in this thesis PyTorch 1.0.2 was used. The reason in this thesis a lower hardware

| | MSDN | IMP | GraphRCNN |
|---|---|---|---|
| RecallSGPlus@50 % | 25.8 | 25.6 | 28.5 |
| RecallSGPlus@100 % | 28.2 | 27.7 | 35.9 |

Table 6.5.: Scene Graph Detection Baseline [19] N = 21877

configuration was used is because the goal of the thesis was not to benchmark scene graphs but to show that scene graphs can be used in a pipeline to extract facts from images. Hence, only a similar performance of the scene graph implementation is enough to show if scene graphs can work for this thesis. Since the difference in performance is only 10% for a challenging metric, it can be said that the performance of the two implementations is similar enough to be comparable. Following the argumentation of [19]

| | MSDN | IMP | GraphRCNN |
|---|---|---|---|
| RecallSGPlus@20 % | 15.13 | 16.74 | 19.48 |
| RecallSGPlusl@50 % | 17.87 | 20.01 | 24.32 |
| RecallSGPlus@100 % | 19.37 | 22.05 | 26.87 |

Table 6.6.: Scene Graph Detection N=1000

RecallSGPlus@100 of 35.9% corresponds to a predicate classification of 59.1% and a phrase classification of 31.6%. This means that 59.1% of relations between two objects have been detected correctly, compared to the ground truth. 31.6% of relations between two object categories have been detected correctly, compared to the ground truth.

### 6.2.3. Entity Linking Experiments

For the analysis of entity linking, a quantitative analysis of the identified objects and relations in the scene graph regarding the configured parameters will be performed. The resulting JSON-LD and RDF output will be discussed because of their textual and visual representation. There have been twenty images analyzed. Each of the analyzed images was processed with nine different configurations for parameters. A $\text{top}_k$ of 1, 5 or 20 to obtain the $\text{top}_k$ best performing predicates from the scene graph. A threshold of 0.0, 0.4, and 0.6 to obtain only the objects detected with a higher score than the threshold. All possible combinations of these parameters have been used in the experiments. For the

experiments on entity linking, a ground truth of a set of twenty images has been created. The ground truth is shown in Listing A.3. There are links to sources of images and the number of people, man, woman, celebrity, celebrity man and celebrity woman noted in the ground truth.

### Recall

The recall metric used in this section in Equation 6.2 is a different one than described in Equation 6.1. The used recall in this section in Equation 6.2 is the sensitivity or hit rate.

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

The precision has not been used since there are no false positive results in these experiments. The results shown in Table 6.7 are the mean of all 20 images for different configurations. The threshold has a significant ($p<0.001$) negative correlation with the people, man, and woman recall, a significant ($p<0.05$) negative correlation with the celebrity recall and the threshold is correlating negatively with the celebrity woman recall but not significantly. The correlation was computed with Pearson Correlation Coefficient. It seems possible that a lower threshold where more objects are found is producing a higher recall. This would mean that the recall is of lower quality for lower thresholds because the score of detected objects is lower. The $top_k$ parameter seems not to correlate with the recall at all.

| | Mean | Std. Deviation |
|---|---|---|
| People Recall % | 73.6 | 34.3 |
| Man Recall % | 77.2 | 30.2 |
| Woman Recall % | 62.1 | 47.1 |
| Celebrity Recall % | 70.1 | 41.2 |
| Celebrity Man Recall % | 70.6 | 42.1 |
| Celebrity Woman Recall % | 38.9 | 49.2 |

Table 6.7.: Entity Linking Recall N=20

What is interesting is that the recall scores a higher result for man than woman. Man are the most common object in the visual genome dataset, woman are the third most common objects and the second most common objects are people. What is striking is that the woman recall is different from celebrities woman recall. The celebrities woman recall is much lower than the celebrities man recall. This inconsistency may be due to the assumption that the Azure Face API cannot handle woman in crowded photos as well as man. A possible proof for that might be that the Azure Face API was performing lower for the Recall experiments in this table than the excellent results for face images where only the face is visible, as shown in Section 6.2.4. The results of the Recall Experiments could show that the configuration of $top_k$ does not affect the recall of people. The $top_k$

affects the predicate of a relation, and it would not be desired that different $\text{top}_k$ show different results for people and celebrities.

### Completeness

The completeness shows to which degree the available knowledge is present in the output of the pipeline [93]. The available knowledge is related to the knowledge represented in the used knowledge graph, e.g., Wikidata or Visual Genome.

Predicate Completeness from Equation 6.3 in Wikidata is the Population Completeness for predicates [93]. These are all predicates occurring in the output of the pipeline compared to all predicates of the same identifier occurring in Wikidata. Predicate Completeness in Visual Genome is the same metric, but describes all predicates of the same identifier occurring in Visual Genome.

$$\frac{\text{no. of values represented for a specific property}}{\text{total no. of values represented for a specific property}} \tag{6.3}$$

Object Completeness from Equation 6.4 in Wikidata or Visual Genome is the Population Completeness for objects [93]. This metric is the sum of all subjects and objects compared to the sum of objects of the same identifier occurring in Wikidata or Visual Genome.

$$\frac{\text{no. of real-world objects are represented}}{\text{total no. of real-world objects are represented}} \tag{6.4}$$

The interlinking completeness [93] in Equation 6.5 is the relation of the number of instances in the output of the pipeline that has a link compared to the total number of instances in the output of the pipeline. Instances in this context refers to all nodes of the graph.

$$\frac{\text{no. of instances in the dataset that are interlinked}}{\text{total no. of instances in a dataset}} \tag{6.5}$$

Regarding the results shown in Table 6.8 all results except the interlinking completeness having a high standard deviation means that the single result for each of the test cases was entirely unique. For instance, the object completeness in Visual Genome had a significant ($p<0.001$) negative correlation (-0.734 with Pearson Correlation Coefficient) with the threshold. All other completeness metrics were correlating with the threshold similarly. This observation of a high standard deviation of the completeness results and a high correlation of the completeness results with the used threshold could lead to the conclusion that the different thresholds used to calculate the different completeness results were influencing the completeness result. The reason the object completeness in Wikidata seems relatively high compared to the other completeness results (except interlinking completeness) might be because the SPARQL query for the count of man in Wikidata returns 165 occurrences of man in Wikidata while the count of man in Visual Genome is 46854. The occurrences of P276 in Wikidata, the property that gives a location relation between two objects, occurs 2,836,663 times in Wikidata while the corresponding predicate in Visual Genome occurs 580,995 times. The high interlinking

| | Mean | Std. Deviation |
|---|---|---|
| Predicate Completeness in Wikidata | $1.239 \times 10^{-4}$ | $1.866 \times 10^{-4}$ |
| Predicate Completeness in Visual Genome | 0.001 | 0.002 |
| Object Completeness in Wikidata | 0.175 | 0.306 |
| Object Completeness in Visual Genome | $1.920 \times 10^{-4}$ | $8.212 \times 10^{-5}$ |
| Interlinking completeness | 0.950 | 0.219 |

Table 6.8.: Entity Linking Completeness N = 20

completeness of the output of the pipeline could show that the graph is well-connected. The only outputs of the pipeline which had no connected nodes were those graphs where the threshold of 0.6 was too high to return any result by the object detection.

### Consistency

The quality of the data structure is analyzed by the data structure's consistency, which means that a knowledge base is free of (logical/formal) contradictions regarding particular knowledge representation and inference mechanisms [93]. All the produced RDF-XML files were imported to Protégé[2] and were processed with the reasoner HermiT[3] to determine whether the XML files are free of (logical/formal) contradictions. All the XML files passed the reasoner, meaning all the XML files are free of (logical/formal) contradictions. A high percentage (95.5%) of the outputs of the pipeline were consistent. The only classes which were not inconsistent were classes from the inheriting parent Image and Spatial Ontology, which were not amended because of sovereignty concerns. Other classes, like the FOAF Image class, were not disjointed because this was the parent class for the whole graph.

### Graph Theoretical Link Prediction

For the following Link Prediction, a sub-graph of the output of the pipeline was used. The sub-graph consisted of only the statements about the image. In other words, the sub-graph only showed the knowledge of the graph without using the whole data structure. Since all statements are connected, all the sub graphs were connected as well. An example of such a sub-graph can be seen in Figure 6.4. The following metrics have been used to analyze the link prediction. The implementations of the metrics have been taken from NetworkX[4].

$$\sum_{w \in \{\Gamma(u) \cap \Gamma(v)\}} \frac{1}{\log |\Gamma(w)|} \tag{6.6}$$

[2] https://protege.stanford.edu/
[3] https://www.cs.ox.ac.uk/isg/tools/HermiT/
[4] https://networkx.org/

The Academic-Adar Index in Equation 6.6 denotes how strongly two nodes $u$ and $v$ are related [94]. This is computed based on the set of common neighbors $\Gamma(w)$ of all available neighbors of $u$ denoted as $\Gamma(u)$ and all available neighbors of $v$ denoted as $\Gamma(v)$. The Academic-Adar Index for Statements(AAI St.) is crucially different from the Academic-Adar Index for SPO(AAI $\langle S, P, O\rangle$). The proposed model suggests that the mean AAI St. with 0.274 is higher for statements than the mean AAI $\langle S, P, O\rangle$ with 0.059. Statements are probably closer related than $\langle S, P, O\rangle$ nodes. The AAI St. seems to be in an average range compared to other studies [94]. The AAI $\langle S, P, O\rangle$ seems relatively low compared to other studies [94]. This difference in AAI might conclude that $\langle S, P, O\rangle$ nodes are more heterogeneous than statement nodes. The heterogeneity could result from $\langle S, P, O\rangle$ nodes representing more heterogeneous circumstances than statements. Statements are always representing the connection between $\langle S, P, O\rangle$s in the image, while $\langle S, P, O\rangle$s represent objects and relations in the image.

$$|\Gamma(u)||\Gamma(v)| \tag{6.7}$$

The Preferential Attachment Score (PAS) in Equation 6.7 gives a probability of how likely a new edge is involved between two nodes $u$ and $v$. The probability is correlated to the number of edges $|\Gamma(u)|$ and $|\Gamma(v)|$ of the two nodes [94]. The value of PAS in Table 6.9 may be a result of some $\langle S, P, O\rangle$ nodes being attached to most of the statement nodes. This high value of PAS suggests that there are dominant objects in the images that

| | Mean | Std. Deviation |
|---|---|---|
| Adamic-Adar Index Statements (AAI St.) | 0.274 | 0.290 |
| Adamic-Adar Index $\langle S, P, O\rangle$ (AAI $\langle S, P, O\rangle$) | 0.059 | 0.092 |
| Preferential Attachment Score (PAS) | 13.199 | 7.578 |
| CCPA score Statements (CCPA St.) | 63.769 | 96.847 |
| CCPA score $\langle S, P, O\rangle$ (CCPA $\langle S, P, O\rangle$) | 39.869 | 60.582 |

Table 6.9.: Entity Linking Link Prediction N=20

interact with a significant number of other objects. This indicates that certain objects have strong associations and interactions with a large portion of the other objects present in the images.

In comparison to other studies on networks, such as [94], the PAS value observed in this context appears relatively high. This high PAS value supports the hypothesis that there are dominant objects in the images, indicating that these objects play a crucial role in the overall object interactions within the network.

$$\alpha \cdot (|\Gamma(u) \cap \Gamma(v)|) + (1 - \alpha) \cdot \frac{N}{d_{uv}} \tag{6.8}$$

The Common Neighbor Centrality [95] in Equation 6.8 is based on two properties, the number of two nodes' common neighbors and their centrality. A common neighbor is a neighbor node that two nodes share. Centrality denotes the importance of a node in a

network. Centrality is defined as the closeness of centrality, which refers to the average shortest distance between two nodes. The alpha controls the weight between common neighbor and centrality. The alpha for the experiments in Tables 6.9 and 6.14 was set to 0.5 to obtain a balanced score because it was not known which property influences the outcome more. Common Neighbor Centrality for Statements (CCPA St.) tends to be higher than Common Neighbor Centrality for SPOs (CCPA $\langle S, P, O \rangle$). It is likely that statements are more centric to the graph than $\langle S, P, O \rangle$ Nodes. Both the CCPA St. and the CCPA $\langle S, P, O \rangle$ seem relatively high compared to other studies [95]. A high CCPA for $\langle S, P, O \rangle$ would back up the argument about dominant objects in the images.

### Link Prediction

Link prediction is used to score a triplet on a subgraph of the Wikidata knowledge graph. The used subgraph contains five million triplets from the Wikidata knowledge graph [96]. The used Python library to compute the link prediction was Pykeen[5]. The score is computed by predicting all possible subjects, predicates, or objects for two known elements of a triplet. E.g., if subject and predicate are known, the score for all possible objects in the triplet is computed. The score of the actual object in the analyzed triplet is taken as the score for that triplet as the tail prediction score. The rank for a prediction is the place a prediction took in the available list of all predictions. The best rank would be 0 and the worst rank would be around 4,500,000. Table 6.10 does not include results for the combination of parameters for an object detection score threshold of 0.0 and $\text{top}_k$(k=0) predicate predictions because that would have been over 300,000 triplets to predict per experiment. Due to constraints related to computational time and cost, the computation of predictions for these parameters was not conducted within the scope of this thesis. The complexity and resource requirements associated with such computations would have exceeded the available resources and time limitations of the research. However, it is acknowledged that exploring these predictions could be a valuable avenue for future research or extended studies. The results in Table 6.10 show the average (Rows 4, 7, 10), minimum (Rows 5, 8, 11) and maximum (Rows 6, 9, 12) rank and scores (Rows: 1, 2, 3) over all configurations of object score threshold and $\text{top}_k$ predicates (except t=0 and k=0). A detailed discussion about the implications of different configurations will be presented. The results for the score in Table 6.10 show that a maximum score of about -3.5 (see row 3) is corresponding to a relatively good result because the corresponding minimum rank is at 0 (see row 8), which would be the best result. The correspondence of the two results has been examined by referring to the computed list of results rather than being displayed in the table. The table may not explicitly show the direct comparison or correspondence of these two specific results. It is necessary to consult a computed list of results to determine the correspondence between the two outcomes. A mean minimum score of about -10.1 (see row 2) corresponds to a maximum tail rank of 1,431,000 (see row 6) which would mean that this triplet, although in comparison to other triplets, it achieved a low score, in general was still satisfying. As

[5] https://github.com/pykeen/pykeen

| No. | | Mean | Std. Deviation | Minimum | Maximum |
|---|---|---|---|---|---|
| 1 | Average score | −6.958 | 0.590 | −8.408 | −4.957 |
| 2 | Min score | −10.113 | 1.328 | −13.134 | −5.416 |
| 3 | Max score | −4.206 | 0.822 | −7.529 | −3.539 |
| | **Tail** | | | | |
| 4 | Average rank | 117240.488 | 78970.288 | 165.200 | 520281.750 |
| 5 | Min rank | 1398.725 | 10332.016 | 0.000 | 107281.000 |
| 6 | Max rank | $1.431 \times 10^{+6}$ | 911060.827 | 477.000 | $4.236 \times 10^{+6}$ |
| | **Head** | | | | |
| 7 | Average rank | 233872.893 | 151834.680 | 161.400 | 596886.947 |
| 8 | Min rank | 2793.031 | 23131.014 | 0.000 | 255918.000 |
| 9 | Max rank | $2.544 \times 10^{+6}$ | $1.433 \times 10^{+6}$ | 454.000 | $4.580 \times 10^{+6}$ |
| | **Predicate** | | | | |
| 10 | Average rank | 127.514 | 53.383 | 4.400 | 205.250 |
| 11 | Min rank | 2.145 | 4.600 | 0.000 | 34.000 |
| 12 | Max rank | 458.183 | 132.035 | 7.000 | 729.000 |

Table 6.10.: Link Prediction

a reminder, the worst rank the triplet could achieve would have been about 4,500,000. It seems possible that some results were the best available for triplets in Wikidata, which could be showing that this pipeline is producing the best available facts for some images. Top k best predicates are correlating negatively in a range from about -0.4 to -0.6 with Pearson Correlation Coefficient with a high significance ($p < 0.001$). The link prediction metrics are not correlating with the threshold. The correlation suggests that the $\text{top}_k$ best predicates are influencing the quality of the facts. Since the scores are lower and the ranks are higher for higher $\text{top}_k$ higher scoring predicates may produce higher quality facts.

### 6.2.4. Azure Face API Experiments

The used recall is described in Equation 6.2. The used dataset is described in Section 6.1.3. The tests were executed on a free plan for Azure Face API. Table 6.11 shows the mean recall for 1168 tests on Azure Face API. The mean recall for the 1168 tests was just over 99%. The recall rate is similar to that of comparable studies described in Table 4.1. It seems possible to use the Azure face recognition for creating facts about images.

### 6.2.5. Case Study: Lance Armstrong Riding a Bike

The following part of this thesis moves on to describe in greater detail the results of pipeline for a single image. There will be a discussion about object detection, face detection, entity linking, statements and metadata. The quality of the JSON-LD and

| | Azure Face API |
|---|---|
| Relevant | 1168 |
| True positive | 1157 |
| False positive % | 0 |
| Precision % | 100 |
| Recall % | 99.058 |

Table 6.11.: Azure Face API Experiment Results N=1168

RDF will be considered. Regarding the recall, there will be an analysis of the results. In terms of Graph Theoretical Approaches the center of the graph, the Academic Adar Index, the Preferential Attachment Score and the common neighbor centrality will be examined.

### Case Description

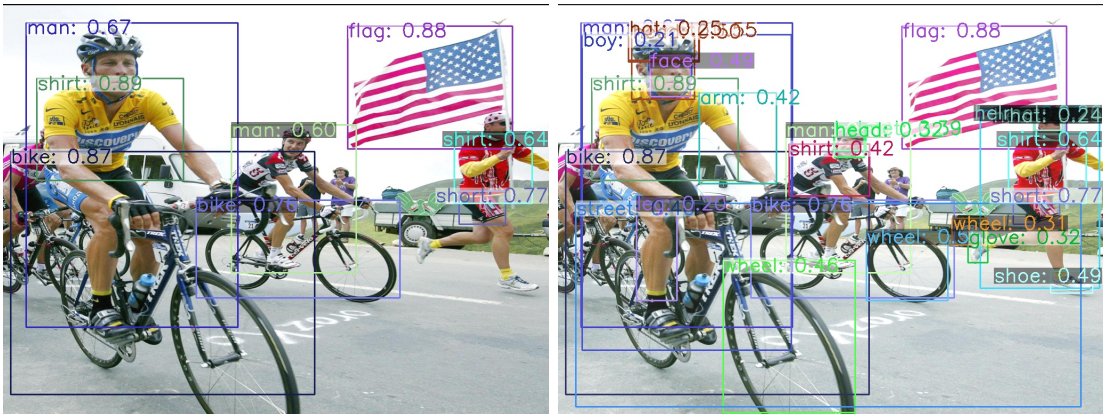
The image of Lance Armstrong riding a bike is showing Lance Armstrong on a bike during the Tour de France, 2005. In the background there are several other bikers, some spectators, cars, and on the right-hand side of the image there is a man carrying the US-American flag. The image subtitle from the page where the image was taken from says: "Dominator in Gelb: Lance Armstrong bei seinem letzten Tour-Triumph 2005" (Translated with deppL[6]: "Dominator in yellow: Lance Armstrong at his last Tour triumph in 2005"). The most important facts about the image are that there is Lance Armstrong in the image, and he is riding a bike at the Tour de France in 2005.

### Object Detection and Face Detection

The object detection with a threshold of 0.6 (t=0.6) with a $\text{top}_k$=1 (k=1) best predicates in Figure 6.1(a) identified two men (one of them Lance Armstrong), two bikes, two shirts, and one flag. It can be that nothing important was left out. The cars in the background may not be important regarding the journalist's subtitle of the image. The object detection with t=0.0 in Figure 6.1(b) shows more objects. It could be interesting that there is a street in the image since that could lead to a conclusion that if the scene showed a bike race, the race could have sections that are held on a street. This would exclude all bike races which are not held on streets. However, the object detection with t=0.0 also showed false detection e.g., Lance Armstrong was detected as a boy as well, which could lead to false assumptions. Besides the street, the object detection with a lower threshold is not giving more useful information about this image when comparing the detected objects with the description of the journalist.

The face detection depicted in Figure 6.2 of Lance Armstrong was successful and did not falsely detect any other known people in the learned model. The used dataset is

[6]https://www.deepl.com/




(a) t=0.6 (b) t=0.0

Figure 6.1.: Object Detection in Lance Armstrong Riding a Bike (k=1)

described in Section 6.1.3. An important fact about the image was found that way. The man in the foreground is Lance Armstrong.

The scene graph detected several relations between the objects in the image. All statements as relations found by the scene graph for t=0.6 and the best predicates for k=1 between the objects are shown in Table 6.12. The graph for these statements as relations is shown in Figure 6.3. If one would only look at the statements and not at the image, it is still obvious that there are two men in the image, each of them riding a bike. Thanks to the entity linking where the node for the man who is Lance Armstrong was linked with the Wikidata article man and the Wikipedia article for Lance Armstrong, it is possible to tell that Lance Armstrong is riding a bike in this image. When examining


Figure 6.2.: Face Detection in Lance Armstrong Riding a Bike (t=0.6, k=1)

the entity linking of the nodes, every node is linked with the corresponding Wikidata page and ConceptNet page. Only people whose faces are detected by face recognition are linked with a Wikipedia page. After examining a possible entity linking, every node was correctly linked with all corresponding pages.

### Statements

The following section will further examine the statements about the image. All statements of the images have been listed in tables and grouped by threshold and $\text{top}_k$ best predicates. If a relation for a threshold was already in a table with a lower k the statement had not been added. Judged by the scores of the predicates, a higher k does not add a better statement. But can it be argued that statements with a lower predicate score are sometimes better? Are all statements with a high predicate score good statements? Since there are over 7000 statements for all configurations, only exemplary statements will be discussed in detail.

The statements which should have the highest quality are listed in Table 6.12 and depicted in Figure 6.3. The figures for threshold 0.0 and 0.4 can be found in the Figures A.1 and A.2. The scene graph implementation often scored relations with the predicate "background" high because the implementation could judge with high confidence that one object is in the background of another object. It could be interesting for further investigation of an image to have a clear structure of the different depths layer of an image. After examining all the statements in Table 6.12 all statements, except for one $\langle man, in\ the\ background\ of, shirt \rangle$, could be true. For this one statement that could not be true, the following could be argued. The shirt, that the man carrying the flag is wearing, is on the same level of depth as the man riding the bike. Evidence for this assumption is the road marking on the street. The three statements about men riding bikes and men wearing shirts are all of high quality. These statements tell facts about the men they are describing.

When examining the statements for t=0.4 and the best predicates for k=1 in Table A.2, there are new objects introduced. The object detection found Lance Armstrong's head,

| Subject | Predicate | Object |
|---|---|---|
| man | background | shirt |
| man | wearing | shirt |
| man | riding | bike |
| man | background | bike |
| man | background | bike |
| man | riding | bike |
| man | background | short |
| man | background | shirt |
| man | background | shirt |

Table 6.12.: Statements for Lance Armstrong Riding a Bike (t=0.6, k=1)

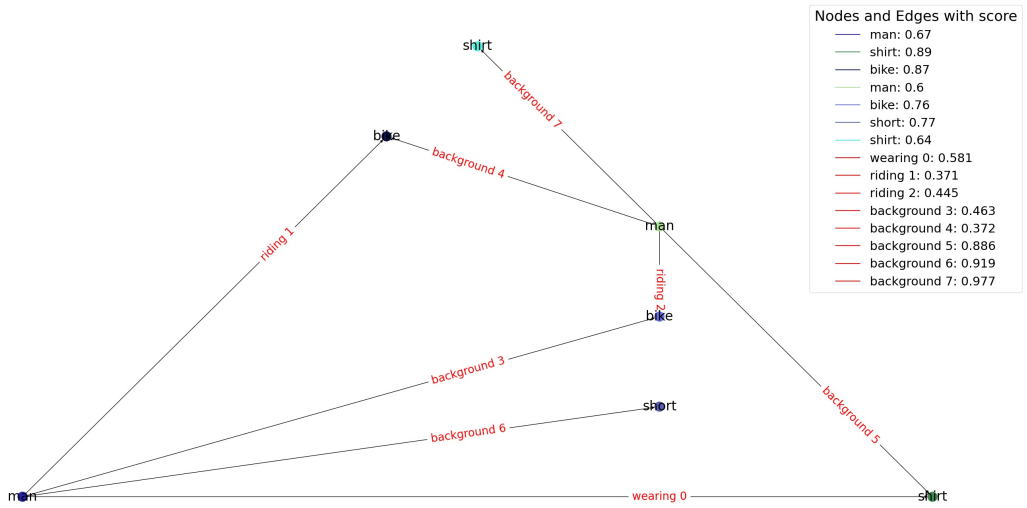



Figure 6.3.: Scene Graph Visualization of Lance Armstrong Riding a Bike (t=0.6, k=1)

face, arm and helmet, the two front wheels of the bikes, the shirt of the bike rider in the background, the man holding the flag as well as his shirt and one of his shoes. There are 16 more background statements about the image and six new statements not about the background. Five of the six statements seem to be of high quality. In these six statements it was recognized that the man holding the flag is wearing his shirt and his shoe. He is on the street. The bike rider in the background is now found to be wearing a shirt. Lance Armstrong wears a helmet. However, he wears his head as well, which is not accurate. It could have several reasons why the head is incorrectly related to Lance Armstrong as wearing. Three possible reasons are the following. First, the scene graph model learned the incorrect relation between head and helmet as wearing because of incorrect expert annotation. Second, the scene graph could have been intimidated by many objects related to Lance Armstrong as wearing. Third, the close distance to objects which are related to Lance Armstrong could have been the reason that the relation wearing has a high score. The correct relation of Lance Armstrong having a head occurs in the best predicates for k=5, with a low score of 0.12. In contrast, the statement about Lance Armstrong wearing his head has a score of 0.55.

When examining the statements for t=0.0 and the best predicates for k=1 in Table A.3, some interesting objects appear, like the head and the helmet of the second biker and people in the background. Some of these detections are not visualized in Figure 6.1(b) because they would overlap too much. However, there are a lot of low-quality detection e.g., a motorcycle, a player, and a skateboard. Lance Armstrong is detected as a boy and as a man. Examining the relations, some new high scoring predicates occur. However, since the detected objects of these relations are of low quality, the relations themselves are of low quality.

Examining the relations with more diverse predicates (k=5, k=20) some interesting

findings appear. Statements for t=0.6 and the best predicates for k=5, t=0.4 and the best predicates for k=5, and t=0.6 and the best predicates for k=20 can be studied in the Table A.4, A.5, A.6. E.g., there is a wheel on the street. However, the bike or car belonging to this wheel could not be found. Another interesting finding was the already mentioned "Lance Amrstron has head" relation. On the downside, there are many new possibilities for already existing relations. E.g., Lance Armstrong could be on the bike, above the bike, or sitting on the bike. All of these statements are not incorrect, but they are of lower quality than Lance Armstrong riding his bike. Other examples are that a lot of wearing relations get a second possibility of being in something $\langle man, in, shorts \rangle$. This is not wrong as well, however, of lower quality than wearing. Overall, the quality of statements is decreasing with a higher k.

In terms of descriptive results, the decreasing quality of statements with a lower threshold and a higher variety in predicates can be observed when considering the results of link prediction for this case. The table can be found in the appendix, Table A.1. The tail, head and predicate mean score decreases 1.6 points from -5.07 to -6.64 when comparing a t=0.6 and the best predicates for k=1 to a t=0.0 and the best predicates for k=20. However, the best ranking results are found for a low threshold and a high k. The best result for a low threshold and a high k is ranked at place 13 while the best result for a high threshold and a low k is ranked at place 18. A similar trend can be observed when comparing the results of the same threshold (t=0.6) for different k. The best score for tail, head, and predicate prediction is the one for k = 1 with -5.07 points. The tail, head, and predicate link prediction score for the best predicates for k=5 rests at -6.22 and the score for the best predicates for k=20 stays at -6.64 points. This might be due to the sheer number of statements in graphs with a low threshold and a high k. There are single high scoring triplet. However, the overall quality is worse.

In summary, it has been shown that the results of the qualitative analysis and the link prediction both show that the highest quality results can be achieved with a high threshold (between 0.6 and 0.4) and a low k (k=1).

### Metadata

Regarding the metadata of the image, it would be interesting to get the geolocation where the image was taken to be able to conclude the fact that the image was taken at the Tour de France. However, since the image does not obtain EXIF data since it is an image in Joint Photographic Experts Group (JPEG) format on the internet, there is no EXIF data present in the image. If the image had been taken with a physical camera and the direct export of the camera's image had been analyzed, it could have been possible to extract the geolocation. The metadata available for this image is the width, height, format, and file size.

### Recall

What follows is an account of the recall of the image. The ground truth for the image was, like all other ground truths for the recall, annotated by an expert. Six people, of

| Threshold | People % | Man % | Celebrity % |
|---|---|---|---|
| 0.0 | 100 | 50 | 100 |
| 0.4 | 50 | 50 | 100 |
| 0.6 | 34 | 34 | 100 |

Table 6.13.: Recall for Lance Armstrong Riding a Bike

which six are men and zero are women, were annotated. One celebrity was annotated. The celebrity was further annotated as Lance Armstrong. Table 6.13 shows that the Celebrity Lance Armstrong was found for each threshold. The increase in the recall with lowering the threshold suggests that a lower threshold leads to a higher recall. What is striking in this table is the difference between the people's recall and the man's recall. This discrepancy could be attributed to not every man being detected. But more men are detected as persons by the object detection.

### Quality of JSON-LD and RDF

Shifting the focus to the quality of the resulting JSON-LD and RDF files, it is essential to evaluate several aspects. These include adherence to the JSON-LD and RDF specifications, correct representation of the data model, consistency in data structure and naming conventions, appropriate use of vocabularies and ontologies, accurate linking of entities and relationships, and compliance with best practices in semantic web standards. As described in Section 5.1.2 the data structure of the output of the scene graph implementation is JSON-LD or RDF. As described in Section 5.1.3 the entity linking adds information to the previously generated data. JSON-LD has been validated by schema.org[7]. The RDF was imported into Protégé and reasoned by the reasoner HermiT. All the reasoning was successful. Moving to the consistency, the range was between 93% and 99%. A possible explanation for these results may be the different numbers of Instances at different threshold levels.

### Graph Theoretical Link Prediction

Regarding graph theoretical metrics this image has been analyzed by its center, the Academic Adar Index, the Preferential Attachment Score and the Common Neighbor Centrality. Figure 6.4 depicts the graph that has been filtered to only show statements and their corresponding subjects, predicates, and objects since these are the facts about the image. The following metrics have been computed from the filtered graphs for the different parameters for threshold and $\text{top}_k$ predicates.

The central node for t=0.0 is the node for man. It might be the node of Lance Armstrong since he is the most prominent figure in the image. With a higher threshold, more centers occur. With t=0.4 the nodes for man and street are in the center. The

[7] https://validator.schema.org/

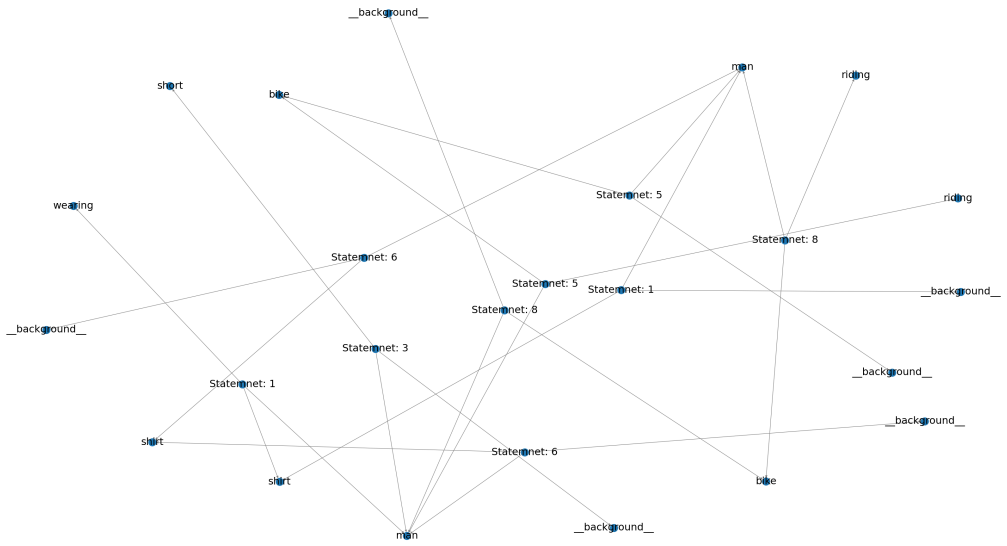



Figure 6.4.: Filtered Graph, Lance Armstrong Riding a Bike (t=0.6, k=1)

addition of the street may lead to the conclusion that many other nodes have edges to the street. With a t=0.6 the center street disappears but four new centers besides man occur. The four new centers are the two bikes and two shirts. In terms of statements and facts, having the node for man of Lance Armstrong in the center of the graph would suggest that the image is about Lance Armstrong. Since we know from the journalist's subtitle of the image that the image is about Lance Armstrong, it could assume a high quality of the graph that the node for Lance Armstrong is its center.

Regarding the Academic-Adar Index, it may be that the AAI St. in Table 6.14 shows that the more objects are detected in an image, the more heterogeneous the statements get. What stands out in this table is the equivalent of the AAI St. for the k best statements. A possible explanation for this might be that k is not implicating the common neighbors, but the weight of the edges. A statement node, for example, always keeps three edges. In contrast, it seems possible that the AAI $\langle S, P, O \rangle$ in this table shows that the more objects are detected in an image, the more heterogeneous the objects get. Another possible explanation might be that objects with a lower score have fewer edges (relations) than objects with a higher score, thus the lower AAI $\langle S, P, O \rangle$. Considering Figure A.2 nodes are introduced with zero edges. The higher the k the more edges are between nodes which might be lowering the AAI $\langle S, P, O \rangle$.

Let us now consider the Preferential Attachment Score. The observed decrease of the PAS when decreasing the threshold could be attributed to the increase in object nodes with a lower threshold. In other words, the decrease in the PAS is a consequence of the increase in edges between $\langle S, P, O \rangle$s and statements when decreasing the threshold. Since

| k | T | Nodes | Edges | AAI St. | AAI $\langle S, P, O \rangle$ | PAS | CCPA St. | CCPA $\langle S, P, O \rangle$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 768 | 555 | 0.04 | 0.02 | 7.71 | 31.17 | 21.67 |
| 1 | 0.4 | 93 | 78 | 0.19 | 0.08 | 9.481 | 5.62 | 3.82 |
| 1 | 0.6 | 27 | 25 | 0.45 | 0.20 | 23.99 | 2.39 | 1.63 |
| 5 | 0.0 | 1792 | 1579 | 0.04 | 0.00 | 5.94 | 88.51 | 57.40 |
| 5 | 0.4 | 217 | 202 | 0.19 | 0.02 | 8.88 | 14.08 | 8.77 |
| 5 | 0.6 | 63 | 61 | 0.45 | 0.07 | 23.59 | 5.19 | 3.27 |
| 20 | 0.0 | 5632 | 5419 | 0.04 | 0.00 | 5.06 | 303.51 | 193.88 |
| 20 | 0.4 | 682 | 667 | 0.19 | 0.01 | 8.48 | 45.83 | 27.96 |
| 20 | 0.6 | 198 | 196 | 0.45 | 0.04 | 23.29 | 15.69 | 9.85 |

Table 6.14.: Graph Statistics for Lance Armstrong Riding a Bike

the decrease in threshold introduces new objects, the PAS could indicate that it is more likely that new edges will be introduced between statements and $\langle S, P, O \rangle$s with a lower threshold. In terms of statements, the PAS could tell us that the lower the threshold is, the higher the possibility that a statement is about a new incident, while a higher threshold will show more likely statements about already known events. Another possible explanation might be that with a lower threshold, most of the statements are about a dominant object. Evidence for the second assumption is Table 6.12 where all subjects are the same.

Another significant aspect of the graph theoretical approach is the Common Neighbor Centrality. The difference between the CCPA St. and the CCPA $\langle S, P, O \rangle$ shown in the table could be indicating that statements are more central to the graph and have more neighbors in common than $\langle S, P, O \rangle$s. As shown in Figure 6.4, there might be a few subject object nodes that have many statements in common, but a lot of other subject object nodes that have only a few or no statements in common. This could result in a higher CCPA score for statements. The increase of the CCPA for higher k might be due to the more edges there are between the same number of nodes, since the overall increase in nodes for higher k are only predicate nodes. The drastic increase in the CCPA while lowering the threshold could be attributed to the centrality of only a few nodes. In terms of the quality of statements, the CCPA could show that only a few nodes become very central while describing the same objects when decreasing the threshold or increasing k.

The previous section has shown that object detection even at a low threshold could return the most important objects in an image. The face detection worked as expected for this image. The quality of statements has not increased for more possible predicates between statements. However, some more accurate statements would have not been found if using fewer predicates. It cannot be said that all statements with a high score for their predicates are high-quality statements, but they tend to be of higher quality. The recall of the object detection performed better for lower thresholds. The produced data was well-defined.

## 6.3. Summary

In summary, the experiments indicated that the similarity implementation worked better than the baseline and could be a promising adaption of the previous approach. The scene graph implementation is not performing to high standards but the best implementation available was used which produced usable outputs. Face recognition is working similarly to other studies. The entity linking experiments and the case study strongly suggested that facts could be produced with the use of scene graphs. Strategies to gain a more profound understanding of the produced graph could be graph theoretical analysis of the graph's structure. It seems possible that high scores of object detection and high scores of predicates in relations lead to high-quality facts about an image. Sometimes not the highest quality statements or facts are scored the highest. However, with a small sample size, caution must be applied, as the findings might not be the same for 20 other images.

# 7. Conclusion

This thesis set out to extract facts from images. The literature study indicates that scene graphs with entity linking are potential candidates to answer the research question. A potential design to extract facts from images with scene graphs and entity linking is presented. This design decouples entity-linking from scene graphs, that was a necessary first step. However, it would be beneficial if the construction of scene graphs could take entity linking into account to produce more adequate results for specific scenes.

This thesis argues that facts are the object values in a triplet $\langle o, r, o' \rangle$. Furthermore, it is argued that an image can be grounded in a scene graph. Hence, the object $o'$ of a scene graph can be a fact about the object $o$. The object $o$ and the object $o'$ have been defined as *subject* and *object* if they are grounded in the image and have a link to a knowledge base. The relation $r$ was defined as a *predicate* if it is grounded in the scene graph and has a link to a knowledge base. The link to the knowledge base is seen as fact about the *subject*, *object* or *predicate*. Such links are achieved by entity linking.

The approach of creating facts by scene graphs seems like the best fit, since triplets that represent facts are explicitly created by scene graph construction. Still, since a particular scene graph is only a possible scene graph for an image, caution must be applied. It seems possible that a learned model is biased in a certain direction and could produce biased facts about an image. The bounding box of a face in an image can be seen as grounded in the scene graph transitively since the bounding box of the face's person is grounded in the scene graph. It has been shown how the link between the bounding box of an object which is a person and a face can be computed. In Chapter 4 it has been established how facts can be extracted and why they can be considered facts.

It has been demonstrated why the data model is implicitly creating facts by creating statements. The data model seemed like the best fit while designing and implementing the prototype. Nevertheless, a unique naming convention for object nodes is highly recommended in the FactCheck framework.

A method of categorizing images into *original*, *duplicate* and *replicate* has been introduced and discussed, and the terms have been defined. An approach was presented of how to align the concepts in a process flow which would probably be the best fit. The usage of an exclusive database for similarity categorization was beneficial while creating the prototype. However, this database structure must be injected into the FactCheck framework.

The design raises the question of how different parameters of a potential implementation influence the quality of the extracted facts. These parameters are tested for a sole implementation of scene graphs. The used scene graph was decided on the recall performance of different scene graph implementations. The comparison of the quality of facts for different implementations of scene graphs is crucial. However, performing

experiments on different implementations of scene graph would have gone beyond the scope of this thesis. It is recommended to use the approach for experiments on the quality of facts presented in this thesis for different scene graph implementations. This approach could ensure that the used scene graph implementation is not biased.

The implementation of the proposed design in Jupyter Notebooks and on Azure was discussed. It was shown which libraries are shared by the implementation. Reasons for the different implementations were given. First, Jupyter Notebooks showcases the execution of the pipeline and builds a potential environment for students in this field to work on. Second, the Azure implementation showcases the use of the pipeline in a cloud environment. The Azure implementation is relevant for using this approach in the FactCheck framework and on large scale. It was discussed why certain technologies have been used, e.g., they are state of the art or as a result of dependencies. If there were certain methods used, e.g., for similarity measurement's feature normalization, it was explained how these methods work. The role of Azure must be seen as critical. While it is a benefit to use services offered by Azure, like face recognition, the implementation of these services is not known. An open-source approach is suggested. The experiments indicate that the similarity implementation works better than the baseline and could be a promising adaption of the previous approach. The scene graph implementation is not performing to high standards. Still, the best implementation available was used, which produced adequate outputs.

Face recognition is working similarly to other studies. The entity linking experiments and the case study strongly suggested that facts could be produced with the use of scene graphs. Strategies to gain a more profound understanding of the produced graph could be graph theoretical analyses of the graph's structure. It seems possible that high scores of object detection and high scores of predicates in relations lead to high-quality facts about an image. Sometimes not the highest quality statements or facts are scored the highest. Nonetheless, with a small sample size, caution must be applied, as the findings might not be the same for 20 other images. Since scene graphs output triplets and the requirement of FactCheck are triplets to create facts, scene graphs are a feasible solution. The main problem with scene graphs is their quality. However, since scene graphs have raised their recall over the last few years, it seems plausible that they can achieve a recall of over 80% in the upcoming years. One essential requirement for neural network models is that such models are inclusive of all genders and skin colors. Scene graphs must not show different results for different genders or skin colors. If scene graphs and face recognition show worse results for women, it would mean that the facts extracted for women are worse than for men. This could result in a bias of facts for women that facts for women are less coherent than for men. A bias of facts for women in a fact-checking framework could draw women further behind in society, politics, science, economics etc. The same is true for skin colors.

# 8. Future Work

This research has thrown up many questions in need of further investigation. Since scene graphs are a topic of ongoing research, new and better implementations arise quickly. While writing this thesis, the used scene graph implementation could have already been updated. A potential candidate for updating the used scene graph implementation would be an implementation by Microsoft[1].

Further research might explore the impact of graph theoretical link prediction on extracted facts for only statements for a larger sample set. This approach could have the benefit of showing the similarity, preferential attachment and centrality of graph nodes without the bias of object nodes. Such experiments could show which statements are central to a graph and if central statements to a graph are similar to other statements not central to the graph. Further, it could show if central statements to the graph are likely to be attached to statements not central to the graph.

Another research should focus on the question if the results of the pipeline could be reproduced for a larger dataset. It would be interesting to see if the facts extracted are still of an average good quality if only the best scoring object detection and the best scoring relations are taken to produce facts, but for a larger dataset. For the FactCheck framework, comparing a larger amount of similarity measurement methods could be beneficial. A benchmark of the similarity measurement methods could come up with the best fit for FactCheck in similarity measurement. Maybe scene graphs themselves could be a good similarity measurement. Given the experimental analysis in this thesis, an evaluation method for extracting facts from images using scene graphs with entity linking should be established.

Further experimental investigations are needed to estimate the influence of different entity linking methods. A potential candidate would be to link the content of an object detected bounding box with the text of a knowledge base. Further research might explore the combination of a scene graph model with such an image-text model. This could benefit learning the scene graph model with image-text model linked entities to get results that are closer to reality.

[1] https://github.com/microsoft/scene_graph_benchmark

# Bibliography


[1] R. Zafarani, X. Zhou, K. Shu, and H. Liu, "Fake News Research: Theories, Detection Strategies, and Open Problems," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[2] P. Kalchgruber, W. Klas, N. Jnoub, and E. M. Roochi, "FactCheck - Identify and Fix Conflicting Data on the Web," in *18TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING JUNE, 2018*, 18TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING, pp. 5–8, June 2018.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 2013.

[6] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[7] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image Retrieval Using Scene Graphs," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3668–3678, 2015.

[10] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A Comprehensive Survey of Scene Graphs: Generation and Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1–26, 2023.

[11] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene Graph Generation by Iterative Message Passing," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3097–3106, 2017.

[12] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials," in *NIPS*, 2011.

[13] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional Random Fields as Recurrent Neural Networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1529–1537, 2015.

[14] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," in *SSST@EMNLP*, 2014.

[15] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene Graph Generation from Objects, Phrases and Region Captions," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1270–1279, 2017.

[16] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural Motifs: Scene Graph Parsing with Global Context," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5831–5840, 2017.

[17] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations," *Int. J. Comput. Vis.*, vol. 123, no. 1, pp. 32–73, 2017.

[18] F. Monti, K. Otness, and M. M. Bronstein, "Motifnet: A Motif-based Graph Convolutional Network for Directed Graphs," in *2018 IEEE Data Science Workshop (DSW)*, pp. 225–228, 2018.

[19] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph R-CNN for Scene Graph Generation," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 690–706, Springer International Publishing, 2018.

[20] S. Li and A. Jain, *Handbook of Face Recognition*. SpringerLink : Bücher, Springer London, 2011.

[21] A. Srivastava, S. Mane, A. Shah, N. Shrivastava, and B. Thakare, "A Survey of Face Detection Algorithms," in *2017 International Conference on Inventive Systems and Control (ICISC)*, pp. 1–4, 2017.

[22] J. Alghamdi, R. Alharthi, R. Alghamdi, W. Alsubaie, R. Alsubaie, D. Alqahtani, R. A. ramadan, L. Alqarni, and R. Alshammari, "A Survey On Face Recognition Algorithms," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, pp. 1–5, 2020.

[23] M. Wang and W. Deng, "Deep Face Recognition: A Survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021.

[24] P. Viola and M. J. Jones, "Robust Real-time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[25] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust Face Detection Using the Hausdorff Distance," in *Audio- and Video-based Biometric Person Authentication* (J. Bigun and F. Smeraldi, eds.), (Berlin, Heidelberg), pp. 90–95, Springer Berlin Heidelberg, 2001.

[26] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A Convolutional Neural Network Cascade for Face Detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5325–5334, 2015.

[27] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 586–591, 1991.

[28] L. Sirovich and M. Kirby, "Low-dimensional Procedure for the Characterization of Human Faces ," *Journal of the Optical Society of America. A, Optics and image science*, vol. 4 3, pp. 519–24, 1987.

[29] T. Hofweber, "Logic and Ontology," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, Spring 2021 ed., 2021.

[30] M. Steup and R. Neta, "Epistemology," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, Fall 2020 ed., 2020.

[31] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[32] "Vocabularies." `https://www.w3.org/standards/semanticweb/ontology`. Accessed: 2022-06-03.

[33] F. Spezzano, A. Shrestha, J. A. Fails, and B. W. Stone, "That's Fake News! Reliability of News When Provided Title, Image, Source Bias & Full Article," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, April 2021.

[34] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[35] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking Sandy: Characterizing and Identifying Fake Images on Twitter During Hurricane Sandy," in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion, (New York, NY, USA), p. 729–736, Association for Computing Machinery, 2013.

[36] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, "Phi.Sh/SoCiaL: The Phishing Landscape through Short URLs," in *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, CEAS '11, (New York, NY, USA), p. 92–101, Association for Computing Machinery, 2011.

[37] A. Bagade, A. Pale, S. Sheth, M. Agarwal, S. Chakrabarti, K. Chebrolu, and S. Sudarshan, "The Kauwa-Kaate Fake News Detection System: Demo," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, CoDS COMAD 2020, (New York, NY, USA), p. 302–306, Association for Computing Machinery, 2020.

[38] N. Jnoub and W. Klas, "Detection of Tampered Images Using Blockchain Technology," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 70–73, 2019.

[39] H. Chi Wong, M. Bern, and D. Goldberg, "An Image Signature for any Kind of Image," in *Proceedings. International Conference on Image Processing*, vol. 1, pp. I–I, 2002.

[40] Y. Liu and Y.-F. B. Wu, "FNED: A Deep Network for Fake News Early Detection on Social Media," *ACM Trans. Inf. Syst.*, vol. 38, May 2020.

[41] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[42] N. Vedula and S. Parthasarathy, "FACE-KEG: Fact Checking Explained Using KnowledgE Graphs," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, (New York, NY, USA), p. 526–534, Association for Computing Machinery, 2021.

[43] X. Wang, C. Yu, S. Baumgartner, and F. Korn, "Relevant Document Discovery for Fact-checking Articles," in *WWW 2018*, 2018.

[44] A. T. Nguyen, A. Kharosekar, S. Krishnan, S. Krishnan, E. Tate, B. C. Wallace, and M. Lease, "Believe It or Not: Designing a Human-AI Partnership for Mixed-initiative Fact-checking," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, (New York, NY, USA), p. 189–199, Association for Computing Machinery, 2018.

[45] K. Popat, S. Mukherjee, J. Strötgen, and G. Weikum, "Where the Truth Lies: Explaining the Credibility of Emerging Claims on the Web and Social Media," in

*Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, (Republic and Canton of Geneva, CHE), p. 1003–1012, International World Wide Web Conferences Steering Committee, 2017.

[46] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," *Work Stat Learn Comput Vision, ECCV*, vol. 1, January 2004.

[47] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[48] V. Viitaniemi and J. Laaksonen, "Spatial Extensions to Bag of Visual Words," in *Proceedings of the 8th ACM International Conference on Image and Video Retrieval, CIVR 2009, Santorini Island, Greece, July 8-10, 2009* (S. Marchand-Maillet and Y. Kompatsiaris, eds.), ACM, 2009.

[49] Z. Jin, J. Cao, Y. Zhang, J. Zhou, and Q. Tian, "Novel Visual and Statistical Image Features for Microblogs News Verification," *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 598–608, 2017.

[50] S. Sun, H. Liu, J. He, and X. Du, "Detecting Event Rumors on Sina Weibo Automatically," in *Web Technologies and Applications* (Y. Ishikawa, J. Li, W. Wang, R. Zhang, and W. Zhang, eds.), (Berlin, Heidelberg), pp. 120–131, Springer Berlin Heidelberg, 2013.

[51] J. Cao, C. Wang, and L. Gao, "A Joint Model for Text and Image Semantic Feature Extraction," in *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2018, (New York, NY, USA), Association for Computing Machinery, 2018.

[52] L. Weiland, I. Hulpus, S. P. Ponzetto, and L. Dietz, "Understanding the Message of Images with Knowledge Base Traversals," in *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, (New York, NY, USA), p. 199–208, Association for Computing Machinery, 2016.

[53] C. Chaudhary, P. Goyal, N. Goyal, and Y.-P. P. Chen, "Image Retrieval for Complex Queries Using Knowledge Embedding," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 16, March 2020.

[54] Z. Battad and M. Si, "Image Sequence Understanding through Narrative Sensemaking," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, (Richland, SC), p. 1458–1460, International Foundation for Autonomous Agents and Multiagent Systems, 2021.

[55] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A Comprehensive Survey of Deep Learning for Image Captioning," *ACM Comput. Surv.*, vol. 51, February 2019.

[56] Q. Wu, C. Shen, P. Wang, A. Dick, and A. v. d. Hengel, "Image Captioning and Visual Question Answering Based on Attributes and External Knowledge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1367–1381, 2018.

[57] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 2048–2057, PMLR, 07–09 Jul 2015.

[58] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical Sequence Training for Image Captioning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195, 2016.

[59] T. Maenishi and K. Tajima, "Identifying Tags Describing Image Contents," in *Proceedings of the 30th ACM Conference on Hypertext and Social Media*, HT '19, (New York, NY, USA), p. 297–298, Association for Computing Machinery, 2019.

[60] S. Song, Q. Miao, and Y. Meng, "Linking Images to Semantic Knowledge Base with User-Generated Tags," in *Proceedings of the 12th International Conference on Semantic Systems*, SEMANTiCS 2016, (New York, NY, USA), p. 93–96, Association for Computing Machinery, 2016.

[61] A. Abdulraheem and L. Qadri Zakaria, "An Automatic Image Tagging Based on Word Co-occurrence Analysis," in *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*, pp. 1–5, 2018.

[62] J. Wang, J. Tang, and J. Luo, "Multimodal Attention with Image Text Spatial Relationship for OCR-based Image Captioning," in *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, (New York, NY, USA), p. 4337–4345, Association for Computing Machinery, 2020.

[63] F. Borisyuk, A. Gordo, and V. Sivakumar, "Rosetta: Large Scale System for Text Detection and Recognition in Images," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, (New York, NY, USA), p. 71–79, Association for Computing Machinery, 2018.

[64] T. Deshpande and N. Mani, "An Interpretable Approach to Hateful Meme Detection," in *Proceedings of the 2021 International Conference on Multimodal Interaction*, ICMI '21, (New York, NY, USA), p. 723–727, Association for Computing Machinery, 2021.

[65] S. Ferrada, B. Bustos, and A. Hogan, "IMGpedia: A Linked Dataset with Content-based Analysis of Wikimedia Images," in *SEMWEB*, 2017.

[66] M. Ousmer, J. Vanderdonckt, and S. Buraga, "An Ontology for Reasoning on Body-Based Gestures," in *Proceedings of the ACM SIGCHI Symposium on Engineering*

*Interactive Computing Systems*, EICS '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[67] M. R. Bashar, S. K. Kang, P. R. Dawadi, and P. K. Rhee, "A Context-aware Statistical Ontology Approach for Adaptive Face Recognition," in *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pp. 698–703, 2007.

[68] K. Tashiro, T. Kawamura, Y. Sei, H. Nakagawa, Y. Tahara, and A. Ohsuga, "Refinement of Ontology-constrained Human Pose Classification," in *2014 IEEE International Conference on Semantic Computing*, pp. 60–67, 2014.

[69] F. Berthelon and P. Sander, "Emotion Ontology for Context Awareness," in *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 59–64, 2013.

[70] O. M. Al-Omair and S. Huang, "A Comparative Study on Detection Accuracy of Cloud-based Emotion Recognition Services," in *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning*, SPML '18, (New York, NY, USA), p. 142–148, Association for Computing Machinery, 2018.

[71] R. Aljamal, A. El-Mousa, and F. Jubair, "A Comparative Review of High-performance Computing Major Cloud Service Providers," in *2018 9th International Conference on Information and Communication Systems (ICICS)*, pp. 181–186, 2018.

[72] Y. Xue, H. Zhang, and H. Ma, "Performance Evaluation of Image and Video Cloud Services," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 733–741, 2018.

[73] W. Ughetta and B. W. Kernighan, "The Old Bailey and OCR: Benchmarking AWS, Azure, and GCP with 180,000 Page Images," in *Proceedings of the ACM Symposium on Document Engineering 2020*, DocEng '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[74] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.

[75] D. E. E. 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)." RFC 3174, 2001.

[76] D. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[77] L. Kalms, K. Mohamed, and D. Göhringer, "Accelerated Embedded AKAZE Feature Detection Algorithm on FPGA," in *Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, HEART2017, (New York, NY, USA), Association for Computing Machinery, 2017.

[78] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-level Performance in Face Verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[79] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.

[80] J. Liu, Y. Deng, T. Bai, Z. Wei, and C. Huang, "Targeting Ultimate Accuracy: Face Recognition via Deep Embedding," 2015. Preprint on webpage at https://arxiv.org/abs/1506.07310.

[81] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range Loss for Deep Face Recognition With Long-tailed Training Data," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[82] Y. Liu, H. Li, and X. Wang, "Rethinking Feature Discrimination and Polymerization for Large-scale Recognition," 2017. Preprint on webpage at https://arxiv.org/abs/1710.00870.

[83] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[84] M. Mehdipour Ghazi and H. Kemal Ekenel, "A Comprehensive Analysis of Deep Learning Based Representation for Face Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[85] P. Ferragina and U. Scaiella, "Fast and Accurate Annotation of Short Texts with Wikipedia Pages," *IEEE software*, vol. 29, no. 1, pp. 70–75, 2011.

[86] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. Similarity Matching in Computer Vision and Multimedia.

[87] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, 1985.

[88] M. Coustaty, N. Tsopze, A. Bouju, K. Bertet, and G. Louis, "Towards Ontology-based Retrieval of Historical Images," *Applied Ontology*, vol. 10, no. 2, pp. 147–167, 2015.

[89] D. Brickley and R. Guha, "RDF Schema 1.1," W3C recommendation, W3C, 2014. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[90] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and Simile Classifiers for Face Verification," *2009 IEEE 12th International Conference on Computer Vision*, pp. 365–372, 2009.

[91] N. Salehi, L. C. Irani, M. S. Bernstein, A. Alkhatib, E. Ogbe, K. Milland, and Clickhappier, "We Are Dynamo: Overcoming Stalling and Friction in Collective Action for Crowd Workers," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, (New York, NY, USA), p. 1621–1630, Association for Computing Machinery, 2015.

[92] M. Castro, D. M. Ballesteros, and D. Renza, "A Dataset of 1050-tampered Color and Grayscale Images (CG-1050)," *Data in Brief*, vol. 28, p. 104864, 2020.

[93] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer, "Quality Assessment for Linked Data: A Survey," *Semantic Web*, vol. 7, no. 1, pp. 63–93, 2016.

[94] D. Liben-Nowell and J. Kleinberg, "The Link Prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[95] I. Ahmad, M. Akhtar, S. Noor, and A. Shahnaz, "Missing Link Prediction using Common Neighbor and Centrality based Parameterized Algorithm," *Scientific Reports*, vol. 10, p. 364, January 2020.

[96] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation," 2019.

# Acronyms

$\langle S, P, O \rangle$ $\langle Subject, Predicate, Object \rangle$. 1, 26, 63, 64, 72, 73

**AAI** $\langle S, P, O \rangle$ Academic Adar Index for $\langle Subjects, Predicates, Objects \rangle$. 63, 72

**AAI St.** Academic Adar Index for Statements. 63, 72, 73

**aGCN** Attentional Graph Convolution Network. 11

**AKAZE** Accelerated KAZE. xi, 27, 40, 41, 50, 56–58

**API** Application Programming Interface. viii, ix, xi, 14, 15, 43–45, 47, 49, 51, 52, 60, 65, 66

**ARM** Azure Resource Manager. 52

**AWS** Amazon Web Services. 21, 22

**BGR** Blue Green Red. 46

**BoV** Bag of Visual Words. 18

**CCPA** Common Neighbor Centrality. 64, 73

**CCPA** $\langle S, P, O \rangle$ Common Neighbor Centrality for $\langle Subjects, Predicates, Objects \rangle$. 63, 64, 73

**CCPA St.** Common Neighbor Centrality for Statements. 63, 64, 73

**CNN** Convolutional Neural Network. 6–8, 12, 20, 22, 32

**CV** Computer Vision. 6

**DTO** Data Transfer Object. 50, 51

**EXIF** Exchangeable Image File Format. 26, 35, 44, 70

**GPU** Graphics Processing Unit. 59

**GRU** Gated Recurrent Unit. 9

**HTTP** Hyper Text Transfer Protocol. 51

**ID** Identifier. xv, 14, 44, 45, 56, 98

**IoU** Intersection-over-Union. 8, 13

**JPEG** Joint Photographic Experts Group. 57, 58, 70

**JSON** JavaScript Object Notation. 14, 43, 45, 56

**JSON-LD** JavaScript Object Notation for Linked Data. xiii, 1, 43, 47, 48, 51, 59, 65, 71

**LFW** Labeled Faces in the Wild. 32, 56

**MB** Megabyte. 36

**MPEG** Moving Picture Experts Group. 35

**MSDN** Multi-level Scene Description Network. 10, 59

**NMS** Non-maximum Supression. 13

**OCR** Optical Character Recognition. 20, 22, 34

**PAS** Preferential Attachment Score. 63, 72, 73

**PCA** Principal Component Analysis. 13

**R-CNN** Regions with Convolutional Neural Network. 6, 8, 31, 33, 42, 46

**RDF** Resource Description Framework. 1, 9, 35, 36, 43, 44, 47, 59, 62, 66, 71

**RDMA** Remote Direct Memory Access. 22

**RGB** Red Green Blue. 46

**RNN** Recurrent Neural Network. 20

**ROI** Region of Interest. 6, 8, 10–12, 30

**SDK** Software Development Kit. 14

**SHA** Secure Hash Algorithm. 27, 28, 40, 50

**SIFT** Scale-invariant Feature Transform. 18, 27, 40

**SPARQL** SPARQL Protocol And RDF Query Language. 36, 43, 44, 47, 61

**SQL** Structured Query Language. 52

**SSD** Single Shot MultiBox Detector. 6

**SURF** Speeded up Robust Features. 18, 40

**SVM** Support Vector Machine. 7, 8

**TB** Terabyte. 36

**URI** Uniform Resource Identifier. 36

**URL** Uniform Resource Locator. 17, 36, 45, 50, 51

**VM** Virtual Machine. 51

**WWW** World Wide Web. 36

**XML** Extensible Markup Language. 43, 62

**YAML** YAML Ain't Markup Language. 46

**YOLO** You Only Look Once. 6

# A. Appendix

In Listing A.1, the index structure for entity linking is presented. There are object names as indexes for Wikidata pages, object names as indexes for ConceptNet pages, predicate names as indexes for Wikidata property pages, and predicate names as indexes for ConceptNet pages.

```
    "label_to_wiki":{
        "plane": "https://www.wikidata.org/wiki/Q197",
        "animal": "https://www.wikidata.org/wiki/Q729",
        "arm": "https://www.wikidata.org/wiki/Q43471",
        "bag": "https://www.wikidata.org/wiki/Q1323314",
        "banana": "https://www.wikidata.org/wiki/Q503",
        "basket": "https://www.wikidata.org/wiki/Q201097",
        "beach": "https://www.wikidata.org/wiki/Q40080",
        "bear": "https://www.wikidata.org/wiki/Q30090244",
        "bed": "https://www.wikidata.org/wiki/Q42177",
        "bench": "https://www.wikidata.org/wiki/Q204776",
        "bike": "https://www.wikidata.org/wiki/Q11442",
        "bird": "https://www.wikidata.org/wiki/Q5113",
        "surfboard": "https://www.wikidata.org/wiki/Q457689",
        "boat": "https://www.wikidata.org/wiki/Q35872",
        "book": "https://www.wikidata.org/wiki/Q571",
        "boot": "https://www.wikidata.org/wiki/Q190868",
        "bottle": "https://www.wikidata.org/wiki/Q80228",
        "bowl": "https://www.wikidata.org/wiki/Q153988",
        "box": "https://www.wikidata.org/wiki/Q987767",
        "boy": "https://www.wikidata.org/wiki/Q3010",
        "branch": "https://www.wikidata.org/wiki/Q2923673",
        "building": "https://www.wikidata.org/wiki/Q41176",
        "bus": "https://www.wikidata.org/wiki/Q5638",
        "cabinet": "https://www.wikidata.org/wiki/Q2741056",
        "cap": "https://www.wikidata.org/wiki/Q6147804",
        "car": "https://www.wikidata.org/wiki/Q1420",
        "cat": "https://www.wikidata.org/wiki/Q146",
        "chair": "https://www.wikidata.org/wiki/Q15026",
        "kid": "https://www.wikidata.org/wiki/Q7569",
        "clock": "https://www.wikidata.org/wiki/Q376",
        "coat": "https://www.wikidata.org/wiki/Q152574",
        "counter": "https://www.wikidata.org/wiki/Q1420266",
        "cow": "https://www.wikidata.org/wiki/Q830",
        "cup": "https://www.wikidata.org/wiki/Q81727",
        "curtain": "https://www.wikidata.org/wiki/Q49005",
        "desk": "https://www.wikidata.org/wiki/Q1064858",
        "dog": "https://www.wikidata.org/wiki/Q144",
        "door": "https://www.wikidata.org/wiki/Q36794",
        "drawer": "https://www.wikidata.org/wiki/Q223575",
        "ear": "https://www.wikidata.org/wiki/Q7362",
        "elephant": "https://www.wikidata.org/wiki/Q7378",
        "engine": "https://www.wikidata.org/wiki/Q44167",
        "eye": "https://www.wikidata.org/wiki/Q7364",
        "face": "https://www.wikidata.org/wiki/Q37017",
        "fence": "https://www.wikidata.org/wiki/Q148571",
        "finger": "https://www.wikidata.org/wiki/Q620207",
        "flag": "https://www.wikidata.org/wiki/Q14660",
        "flower": "https://www.wikidata.org/wiki/Q506",
        "food": "https://www.wikidata.org/wiki/Q2095",
        "fork": "https://www.wikidata.org/wiki/Q81881",
        "fruit": "https://www.wikidata.org/wiki/Q3314483",
        "giraffe": "https://www.wikidata.org/wiki/Q15083",
        "girl": "https://www.wikidata.org/wiki/Q3031",
        "glass": "https://www.wikidata.org/wiki/Q11469",
        "glove": "https://www.wikidata.org/wiki/Q169031",
        "guy": "https://www.wikidata.org/wiki/Q6581097",
        "hair": "https://www.wikidata.org/wiki/Q28472",
        "hand": "https://www.wikidata.org/wiki/Q33767",
        "handle": "https://www.wikidata.org/wiki/Q200266",
```

```
        "hat": "https://www.wikidata.org/wiki/Q80151",
        "head": "https://www.wikidata.org/wiki/Q23640",
        "helmet": "https://www.wikidata.org/wiki/Q173603",
        "hill": "https://www.wikidata.org/wiki/Q54050",
        "horse": "https://www.wikidata.org/wiki/Q726",
        "house": "https://www.wikidata.org/wiki/Q3947",
        "jacket": "https://www.wikidata.org/wiki/Q849964",
        "jean": "https://www.wikidata.org/wiki/Q83363",
        "kite": "https://www.wikidata.org/wiki/Q42861",
        "lady": "https://www.wikidata.org/wiki/Q1378024",
        "lamp": "https://www.wikidata.org/wiki/Q1138737",
        "laptop": "https://www.wikidata.org/wiki/Q3962",
        "leaf": "https://www.wikidata.org/wiki/Q33971",
        "leg": "https://www.wikidata.org/wiki/Q133105",
        "letter": "https://www.wikidata.org/wiki/Q9788",
        "light": "https://www.wikidata.org/wiki/Q9128",
        "logo": "https://www.wikidata.org/wiki/Q1886349",
        "man": "https://www.wikidata.org/wiki/Q8441",
        "men": "https://www.wikidata.org/wiki/Q1410641",
        "motorcycle": "https://www.wikidata.org/wiki/Q34493",
        "mountain": "https://www.wikidata.org/wiki/Q8502",
        "mouth": "https://www.wikidata.org/wiki/Q9635",
        "neck": "https://www.wikidata.org/wiki/Q9633",
        "nose": "https://www.wikidata.org/wiki/Q7363",
        "number": "https://www.wikidata.org/wiki/Q11563",
        "orange": "https://www.wikidata.org/wiki/Q13191",
        "pant": "https://www.wikidata.org/wiki/Q39908",
        "paper": "https://www.wikidata.org/wiki/Q11472",
        "paw": "https://www.wikidata.org/wiki/Q219588",
        "people": "https://www.wikidata.org/wiki/Q5",
        "person": "https://www.wikidata.org/wiki/Q215627",
        "phone": "https://www.wikidata.org/wiki/Q11035",
        "pillow": "https://www.wikidata.org/wiki/Q99895",
        "pizza": "https://www.wikidata.org/wiki/Q177",
        "plant": "https://www.wikidata.org/wiki/Q756",
        "plate": "https://www.wikidata.org/wiki/Q57216",
        "player": "https://www.wikidata.org/wiki/Q4197743",
        "post": "https://www.wikidata.org/wiki/Q2180428",
        "pot": "https://www.wikidata.org/wiki/Q2366864",
        "racket": "https://www.wikidata.org/wiki/Q240502",
        "railing": "https://www.wikidata.org/wiki/Q3095365",
        "rock": "https://www.wikidata.org/wiki/Q8063",
        "roof": "https://www.wikidata.org/wiki/Q83180",
        "room": "https://www.wikidata.org/wiki/Q180516",
        "screen": "https://www.wikidata.org/wiki/Q327065",
        "seat": "https://www.wikidata.org/wiki/Q2207370",
        "sheep": "https://www.wikidata.org/wiki/Q7368",
        "shelf": "https://www.wikidata.org/wiki/Q2637814",
        "shirt": "https://www.wikidata.org/wiki/Q76768",
        "shoe": "https://www.wikidata.org/wiki/Q22676",
        "short": "https://www.wikidata.org/wiki/Q223269",
        "sidewalk": "https://www.wikidata.org/wiki/Q177749",
        "sign": "https://www.wikidata.org/wiki/Q3695082",
        "sink": "https://www.wikidata.org/wiki/Q140565",
        "skateboard": "https://www.wikidata.org/wiki/Q15783",
        "ski": "https://www.wikidata.org/wiki/Q172226",
        "skier": "https://www.wikidata.org/wiki/Q4270517",
        "sneaker": "https://www.wikidata.org/wiki/Q1929383",
        "snow": "https://www.wikidata.org/wiki/Q7561",
        "sock": "https://www.wikidata.org/wiki/Q43663",
        "stand": "https://www.wikidata.org/wiki/Q56071807",
        "street": "https://www.wikidata.org/wiki/Q34442",
        "table": "https://www.wikidata.org/wiki/Q14748",
        "tail": "https://www.wikidata.org/wiki/Q60960",
        "tie": "https://www.wikidata.org/wiki/Q44416",
        "tile": "https://www.wikidata.org/wiki/Q468402",
        "tire": "https://www.wikidata.org/wiki/Q169545",
        "toilet": "https://www.wikidata.org/wiki/Q7857",
        "towel": "https://www.wikidata.org/wiki/Q131696",
        "tower": "https://www.wikidata.org/wiki/Q12518",
        "track": "https://www.wikidata.org/wiki/Q160342",
        "train": "https://www.wikidata.org/wiki/Q870",
        "tree": "https://www.wikidata.org/wiki/Q10884",
        "truck": "https://www.wikidata.org/wiki/Q43193",
        "trunk": "https://www.wikidata.org/wiki/Q193472",
        "umbrella": "https://www.wikidata.org/wiki/Q41607",
        "vase": "https://www.wikidata.org/wiki/Q191851",
        "vegetable": "https://www.wikidata.org/wiki/Q11004",
        "vehicle": "https://www.wikidata.org/wiki/Q42889",
        "wave": "https://www.wikidata.org/wiki/Q165848",
        "wheel": "https://www.wikidata.org/wiki/Q446",
```

```
        "window": "https://www.wikidata.org/wiki/Q35473",
        "windshield": "https://www.wikidata.org/wiki/Q13693",
        "wing": "https://www.wikidata.org/wiki/Q161358",
        "wire": "https://www.wikidata.org/wiki/Q551997",
        "woman": "https://www.wikidata.org/wiki/Q467",
        "zebra": "https://www.wikidata.org/wiki/Q32789",
        "board": "https://www.wikidata.org/wiki/Q72926076",
        "pole": "https://www.wikidata.org/wiki/Q2180428",
        "child": "https://www.wikidata.org/wiki/Q7569",
        "airplane": "https://www.wikidata.org/wiki/Q197"
    },
    "label_to_conceptnet":{
        "kite": "http://api.conceptnet.io/c/en/kite",
        "pant": "http://api.conceptnet.io/c/en/pant",
        "bowl": "http://api.conceptnet.io/c/en/bowl",
        "laptop": "http://api.conceptnet.io/c/en/laptop",
        "paper": "http://api.conceptnet.io/c/en/paper",
        "motorcycle": "http://api.conceptnet.io/c/en/motorcycle",
        "railing": "http://api.conceptnet.io/c/en/railing",
        "chair": "http://api.conceptnet.io/c/en/chair",
        "windshield": "http://api.conceptnet.io/c/en/windshield",
        "tire": "http://api.conceptnet.io/c/en/tire",
        "cup": "http://api.conceptnet.io/c/en/cup",
        "bench": "http://api.conceptnet.io/c/en/bench",
        "tail": "http://api.conceptnet.io/c/en/tail",
        "bike": "http://api.conceptnet.io/c/en/bike",
        "board": "http://api.conceptnet.io/c/en/board",
        "orange": "http://api.conceptnet.io/c/en/orange",
        "hat": "http://api.conceptnet.io/c/en/hat",
        "finger": "http://api.conceptnet.io/c/en/finger",
        "plate": "http://api.conceptnet.io/c/en/plate",
        "woman": "http://api.conceptnet.io/c/en/woman",
        "handle": "http://api.conceptnet.io/c/en/handle",
        "branch": "http://api.conceptnet.io/c/en/branch",
        "food": "http://api.conceptnet.io/c/en/food",
        "bear": "http://api.conceptnet.io/c/en/bear",
        "vase": "http://api.conceptnet.io/c/en/vase",
        "vegetable": "http://api.conceptnet.io/c/en/vegetable",
        "giraffe": "http://api.conceptnet.io/c/en/giraffe",
        "desk": "http://api.conceptnet.io/c/en/desk",
        "lady": "http://api.conceptnet.io/c/en/lady",
        "towel": "http://api.conceptnet.io/c/en/towel",
        "glove": "http://api.conceptnet.io/c/en/glove",
        "bag": "http://api.conceptnet.io/c/en/bag",
        "nose": "http://api.conceptnet.io/c/en/nose",
        "rock": "http://api.conceptnet.io/c/en/rock",
        "guy": "http://api.conceptnet.io/c/en/guy",
        "shoe": "http://api.conceptnet.io/c/en/shoe",
        "sneaker": "http://api.conceptnet.io/c/en/sneaker",
        "fence": "http://api.conceptnet.io/c/en/fence",
        "people": "http://api.conceptnet.io/c/en/people",
        "house": "http://api.conceptnet.io/c/en/house",
        "seat": "http://api.conceptnet.io/c/en/seat",
        "hair": "http://api.conceptnet.io/c/en/hair",
        "street": "http://api.conceptnet.io/c/en/street",
        "roof": "http://api.conceptnet.io/c/en/roof",
        "racket": "http://api.conceptnet.io/c/en/racket",
        "logo": "http://api.conceptnet.io/c/en/logo",
        "girl": "http://api.conceptnet.io/c/en/girl",
        "arm": "http://api.conceptnet.io/c/en/arm",
        "flower": "http://api.conceptnet.io/c/en/flower",
        "leaf": "http://api.conceptnet.io/c/en/leaf",
        "clock": "http://api.conceptnet.io/c/en/clock",
        "hill": "http://api.conceptnet.io/c/en/hill",
        "bird": "http://api.conceptnet.io/c/en/bird",
        "umbrella": "http://api.conceptnet.io/c/en/umbrella",
        "leg": "http://api.conceptnet.io/c/en/leg",
        "screen": "http://api.conceptnet.io/c/en/screen",
        "men": "http://api.conceptnet.io/c/en/men",
        "sink": "http://api.conceptnet.io/c/en/sink",
        "trunk": "http://api.conceptnet.io/c/en/trunk",
        "post": "http://api.conceptnet.io/c/en/post",
        "sidewalk": "http://api.conceptnet.io/c/en/sidewalk",
        "box": "http://api.conceptnet.io/c/en/box",
        "boy": "http://api.conceptnet.io/c/en/boy",
        "cow": "http://api.conceptnet.io/c/en/cow",
        "skateboard": "http://api.conceptnet.io/c/en/skateboard",
        "plane": "http://api.conceptnet.io/c/en/plane",
        "stand": "http://api.conceptnet.io/c/en/stand",
        "pillow": "http://api.conceptnet.io/c/en/pillow",
        "ski": "http://api.conceptnet.io/c/en/ski",
```

```
        "wire": "http://api.conceptnet.io/c/en/wire",
        "toilet": "http://api.conceptnet.io/c/en/toilet",
        "pot": "http://api.conceptnet.io/c/en/pot",
        "sign": "http://api.conceptnet.io/c/en/sign",
        "number": "http://api.conceptnet.io/c/en/number",
        "pole": "http://api.conceptnet.io/c/en/pole",
        "table": "http://api.conceptnet.io/c/en/table",
        "boat": "http://api.conceptnet.io/c/en/boat",
        "sheep": "http://api.conceptnet.io/c/en/sheep",
        "horse": "http://api.conceptnet.io/c/en/horse",
        "eye": "http://api.conceptnet.io/c/en/eye",
        "sock": "http://api.conceptnet.io/c/en/sock",
        "window": "http://api.conceptnet.io/c/en/window",
        "vehicle": "http://api.conceptnet.io/c/en/vehicle",
        "curtain": "http://api.conceptnet.io/c/en/curtain",
        "kid": "http://api.conceptnet.io/c/en/kid",
        "banana": "http://api.conceptnet.io/c/en/banana",
        "engine": "http://api.conceptnet.io/c/en/engine",
        "head": "http://api.conceptnet.io/c/en/head",
        "door": "http://api.conceptnet.io/c/en/door",
        "bus": "http://api.conceptnet.io/c/en/bus",
        "cabinet": "http://api.conceptnet.io/c/en/cabinet",
        "glass": "http://api.conceptnet.io/c/en/glass",
        "flag": "http://api.conceptnet.io/c/en/flag",
        "train": "http://api.conceptnet.io/c/en/train",
        "child": "http://api.conceptnet.io/c/en/child",
        "ear": "http://api.conceptnet.io/c/en/ear",
        "surfboard": "http://api.conceptnet.io/c/en/surfboard",
        "room": "http://api.conceptnet.io/c/en/room",
        "player": "http://api.conceptnet.io/c/en/player",
        "car": "http://api.conceptnet.io/c/en/car",
        "cap": "http://api.conceptnet.io/c/en/cap",
        "tree": "http://api.conceptnet.io/c/en/tree",
        "bed": "http://api.conceptnet.io/c/en/bed",
        "cat": "http://api.conceptnet.io/c/en/cat",
        "coat": "http://api.conceptnet.io/c/en/coat",
        "skier": "http://api.conceptnet.io/c/en/skier",
        "zebra": "http://api.conceptnet.io/c/en/zebra",
        "fork": "http://api.conceptnet.io/c/en/fork",
        "drawer": "http://api.conceptnet.io/c/en/drawer",
        "airplane": "http://api.conceptnet.io/c/en/airplane",
        "helmet": "http://api.conceptnet.io/c/en/helmet",
        "shirt": "http://api.conceptnet.io/c/en/shirt",
        "paw": "http://api.conceptnet.io/c/en/paw",
        "boot": "http://api.conceptnet.io/c/en/boot",
        "snow": "http://api.conceptnet.io/c/en/snow",
        "lamp": "http://api.conceptnet.io/c/en/lamp",
        "book": "http://api.conceptnet.io/c/en/book",
        "animal": "http://api.conceptnet.io/c/en/animal",
        "elephant": "http://api.conceptnet.io/c/en/elephant",
        "tile": "http://api.conceptnet.io/c/en/tile",
        "tie": "http://api.conceptnet.io/c/en/tie",
        "beach": "http://api.conceptnet.io/c/en/beach",
        "pizza": "http://api.conceptnet.io/c/en/pizza",
        "wheel": "http://api.conceptnet.io/c/en/wheel",
        "plant": "http://api.conceptnet.io/c/en/plant",
        "tower": "http://api.conceptnet.io/c/en/tower",
        "mountain": "http://api.conceptnet.io/c/en/mountain",
        "track": "http://api.conceptnet.io/c/en/track",
        "hand": "http://api.conceptnet.io/c/en/hand",
        "fruit": "http://api.conceptnet.io/c/en/fruit",
        "mouth": "http://api.conceptnet.io/c/en/mouth",
        "letter": "http://api.conceptnet.io/c/en/letter",
        "shelf": "http://api.conceptnet.io/c/en/shelf",
        "wave": "http://api.conceptnet.io/c/en/wave",
        "man": "http://api.conceptnet.io/c/en/man",
        "building": "http://api.conceptnet.io/c/en/building",
        "short": "http://api.conceptnet.io/c/en/short",
        "neck": "http://api.conceptnet.io/c/en/neck",
        "phone": "http://api.conceptnet.io/c/en/phone",
        "light": "http://api.conceptnet.io/c/en/light",
        "counter": "http://api.conceptnet.io/c/en/counter",
        "dog": "http://api.conceptnet.io/c/en/dog",
        "face": "http://api.conceptnet.io/c/en/face",
        "jacket": "http://api.conceptnet.io/c/en/jacket",
        "person": "http://api.conceptnet.io/c/en/person",
        "truck": "http://api.conceptnet.io/c/en/truck",
        "bottle": "http://api.conceptnet.io/c/en/bottle",
        "basket": "http://api.conceptnet.io/c/en/basket",
        "jean": "http://api.conceptnet.io/c/en/jean",
        "wing": "http://api.conceptnet.io/c/en/wing"
```

```
    },
    "predicate_to_wiki": {
        "__background__": "https://www.wikidata.org/wiki/Property:P129",
        "and": "https://www.wikidata.org/wiki/Property:P527",
        "says": "https://www.wikidata.org/wiki/Property:P1412",
        "belonging to": "https://www.wikidata.org/wiki/Property:P127",
        "over": "https://www.wikidata.org/wiki/Property:P129",
        "parked on": "https://www.wikidata.org/wiki/Property:P129",
        "growing on": "https://www.wikidata.org/wiki/Property:P129",
        "standing on": "https://www.wikidata.org/wiki/Property:P129",
        "made of": "https://www.wikidata.org/wiki/Property:P186",
        "attached to": "https://www.wikidata.org/wiki/Property:P2789",
        "at": "https://www.wikidata.org/wiki/Property:P129",
        "in": "https://www.wikidata.org/wiki/Property:P642",
        "hanging from": "https://www.wikidata.org/wiki/Property:P129",
        "wears": "https://www.wikidata.org/wiki/Property:P3828",
        "in front of": "https://www.wikidata.org/wiki/Property:P276",
        "from": "https://www.wikidata.org/wiki/Property:P2210",
        "for": "https://www.wikidata.org/wiki/Property:P642",
        "watching": "https://www.wikidata.org/wiki/Property:P3173",
        "lying on": "https://www.wikidata.org/wiki/Property:P276",
        "to": "https://www.wikidata.org/wiki/Property:P1444",
        "behind": "https://www.wikidata.org/wiki/Property:P276",
        "flying in": "https://www.wikidata.org/wiki/Property:P129",
        "looking at": "https://www.wikidata.org/wiki/Property:P3173",
        "on back of": "https://www.wikidata.org/wiki/Property:P276",
        "holding": "https://www.wikidata.org/wiki/Property:P129",
        "between": "https://www.wikidata.org/wiki/Property:P710",
        "laying on": "https://www.wikidata.org/wiki/Property:P129",
        "riding": "https://www.wikidata.org/wiki/Property:P5317",
        "has": "https://www.wikidata.org/wiki/Property:P527",
        "across": "https://www.wikidata.org/wiki/Property:P276",
        "wearing": "https://www.wikidata.org/wiki/Property:P3828",
        "walking on": "https://www.wikidata.org/wiki/Property:P121",
        "eating": "https://www.wikidata.org/wiki/Property:P129",
        "above": "https://www.wikidata.org/wiki/Property:P276",
        "part of": "https://www.wikidata.org/wiki/Property:P361",
        "walking in": "https://www.wikidata.org/wiki/Property:P121",
        "sitting on": "https://www.wikidata.org/wiki/Property:P129",
        "under": "https://www.wikidata.org/wiki/Property:P276",
        "covered in": "https://www.wikidata.org/wiki/Property:P129",
        "carrying": "https://www.wikidata.org/wiki/Property:P3349",
        "using": "https://www.wikidata.org/wiki/Property:P2283",
        "along": "https://www.wikidata.org/wiki/Property:P276",
        "with": "https://www.wikidata.org/wiki/Property:P1706",
        "on": "https://www.wikidata.org/wiki/Property:P276",
        "covering": "https://www.wikidata.org/wiki/Property:P129",
        "of": "https://www.wikidata.org/wiki/Property:P642",
        "against": "https://www.wikidata.org/wiki/Property:P129",
        "playing": "https://www.wikidata.org/wiki/Property:P129",
        "near": "https://www.wikidata.org/wiki/Property:P276",
        "painted on": "https://www.wikidata.org/wiki/Property:P7937",
        "mounted on": "https://www.wikidata.org/wiki/Property:P3091"
    },
    "predicate_to_conceptnet": {
        "__background__": "https://conceptnet.io/c/en/background",
        "and": "https://conceptnet.io/c/en/and",
        "says": "https://conceptnet.io/c/en/says",
        "belonging to": "https://conceptnet.io/r/HasA",
        "over": "https://conceptnet.io/c/en/over",
        "parked on": "https://conceptnet.io/c/en/parked",
        "growing on": "https://conceptnet.io/c/en/growing_on",
        "standing on": "https://conceptnet.io/c/en/standing_on",
        "made of": "https://conceptnet.io/c/en/made_of",
        "attached to": "https://conceptnet.io/c/en/attached_to",
        "at": "https://conceptnet.io/c/en/at",
        "in": "https://conceptnet.io/c/en/in",
        "hanging from": "https://conceptnet.io/c/en/hanging",
        "wears": "https://conceptnet.io/c/en/wears",
        "in front of": "https://conceptnet.io/c/en/in_front_of",
        "from": "https://conceptnet.io/c/en/from",
        "for": "https://conceptnet.io/c/en/for",
        "watching": "https://conceptnet.io/c/en/watching",
        "lying on": "https://conceptnet.io/c/en/lying_on",
        "to": "https://conceptnet.io/c/en/to",
        "behind": "https://conceptnet.io/c/en/behind",
        "flying in": "https://conceptnet.io/c/en/flying_in",
        "looking at": "https://conceptnet.io/c/en/looking_at",
        "on back of": "https://conceptnet.io/c/en/on_back_of",
        "holding": "https://conceptnet.io/c/en/holding",
        "between": "https://conceptnet.io/c/en/between",
```

```
        "laying on": "https://conceptnet.io/c/en/laying_on",
        "riding": "https://conceptnet.io/c/en/riding",
        "has": "https://conceptnet.io/c/en/has",
        "across": "https://conceptnet.io/c/en/across",
        "wearing": "https://conceptnet.io/c/en/wearing",
        "walking on": "https://conceptnet.io/c/en/walking_on",
        "eating": "https://conceptnet.io/c/en/eating",
        "above": "https://conceptnet.io/c/en/above",
        "part of": "https://conceptnet.io/c/en/part_of",
        "walking in": "https://conceptnet.io/c/en/walking_in",
        "sitting on": "https://conceptnet.io/c/en/sitting_on",
        "under": "https://conceptnet.io/c/en/under",
        "covered in": "https://conceptnet.io/c/en/covered_in",
        "carrying": "https://conceptnet.io/c/en/carrying",
        "using": "https://conceptnet.io/c/en/using",
        "along": "https://conceptnet.io/c/en/along",
        "with": "https://conceptnet.io/c/en/with",
        "on": "https://conceptnet.io/c/en/on",
        "covering": "https://conceptnet.io/c/en/covering",
        "of": "https://conceptnet.io/c/en/of",
        "against": "https://conceptnet.io/c/en/against",
        "playing": "https://conceptnet.io/c/en/playing",
        "near": "https://conceptnet.io/c/en/near",
        "painted on": "https://conceptnet.io/c/en/painted_on",
        "mounted on": "https://conceptnet.io/c/en/mounted"
    }
```

Listing A.1: Static JSON Entity Linking

Listing A.2 provides an example of the persisted data in the Azure face recognition service, linked to a Wikipedia page.

```
    "7f336fe2-7561-42b2-b761-035640ca1ed9": {
        "additional_properties": {},
        "name": "Barack Obama",
        "persisted_face_ids": [
            "1fb13f84-1d81-4e3c-890e-c1f262453742",
            "3fdda2d6-01b9-491c-a24c-96eb27e6cfe4",
            "476e6efa-a227-43fb-917f-67663bd56806",
            "66b8aea1-94ac-4c3a-814b-ad9225db1263",
            "98bf7313-0ac8-4321-aa84-3df4fb3cc4f8",
            "b371715a-1fe5-493f-ab76-33d99e02593f",
            "b45d2d98-7dbb-4b7c-8355-b9d13205bd1f",
            "bf7366ce-4b2c-423f-83f0-c2f317fcb8bb",
            "d6b3e488-4942-4736-a51b-dec28b629890",
            "fe9363b0-b3b9-429e-8e1b-cb0ed82acbad"
        ],
        "wiki_link": "https://en.wikipedia.org/wiki/Barack_Obama"
    }
```

Listing A.2: Barack Obama Person ID

Listing A.3 presents the ground truth used to execute the experiments. The values in the settings are example configurations for a specific experiment.

```

{
    "settings": {
        "image_id": 2,
        "top_k": 1,
        "threshold": 0.6
    },
    "image": {
        "1": "https://s.abcnews.com/images/International/AP_Obama_Abe_BM_20160526_16x9_1600.jpg",
        "2": "https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/3684018560_ca14639e91
     _o_Barack_Obama_with_Silvio_Berlusconi_ORIGINAL_SIZE.jpg/2560px-3684018560_ca14639e91
     _o_Barack_Obama_with_Silvio_Berlusconi_ORIGINAL_SIZE.jpg",
        "3": "https://upload.wikimedia.org/wikipedia/commons/1/1e/
     Silvio_Berlusconi_Barack_Obama_Jose_Manuel_Barroso_Angela_Merkel_and_Nicolas_Sarkozy_cropped_36th_G8
     _summit_member_20100625.jpg",
        "4": "https://upload.wikimedia.org/wikipedia/commons/2/25/Shinzo_Abe_%26_Barack_Obama.jpg",
        "5": "https://media0.faz.net/ppmedia/aktuell/3900987024/1.6210080/mmobject-still_full/hinter-den-
     kulissen-rumort-es.jpg",
        "6": "https://img.welt.de/img/sport/mobile208844621/3152500747-ci102l-w1024/LANCE-ARMSTRONG.jpg",
        "7": "https://www.buzzfeed.de/bilder/2022/05/25/91629645/28964126-donald-faison-und-zach-braff-als-
     turc-und-jd-aus-scrubs-QYBG.jpg",
```

```
        "8": "https://images.bstatic.de/w3vE1yd9ejqFF3ZwK1ePSDJPNgM=/1200x900/filters:focal(371x204:391x224)/images/2bcd5859/d475/44f7/83b6/eed89156fdf8.jpg",
        "9": "https://i.pinimg.com/originals/ff/a3/1c/ffa31c29fd08df4a29862ff6c10b0d16.jpg",
        "10": "https://iconicimages.net/app/uploads/2017/01/JB078.jpg",
        "11": "https://www1.pictures.zimbio.com/gi/Nurse+Jackie+Star+Edie+Falco+Throws+Out+First+e-NDED6v-KJx.jpg",
        "12": "https://dygtyjqp7pi0m.cloudfront.net/i/40821/35088699_3.jpg?v=8D784DBD3797070",
        "13": "https://i.pinimg.com/originals/5b/30/00/5b3000aa1003ca42abe175eaaffc97b3.jpg",
        "14": "https://cdn.justjared.com/wp-content/uploads/2008/07/arod-nyc/alex-rodriguez-nyc-apartment-04.jpg",
        "15": "https://media1.popsugar-assets.com/files/thumbor/MoHUHhwPGL7qvvtWD_HsQICw3hY/fit-in/2048xorig/filters:format_auto-!!-:strip_icc-!!-/2014/02/12/009/n/1922398/857a0162153228fa_b96cf36e934f11e3a5b712a3de486cc5_8/i/Drew-Barrymore-indulged-her-Chinese-food-craving.jpg",
        "16": "https://c8.alamy.com/comp/2D22AMP/israels-prime-minister-ehud-olmert-eats-mufleta-a-traditional-food-during-a-celebration-of-the-jewish-holiday-of-passover-in-ashdod-near-tel-aviv-april-9-2007-reutersgil-cohen-magen-israel-2D22AMP.jpg",
        "17": "https://i.pinimg.com/736x/73/8e/1b/738e1b43169d464b89b84bf1b0b99bae.jpg",
        "18": "https://assets.vogue.com/photos/59b57edef83970107aeda2ee/master/w_1600,c_limit/00-story-image-victoriab.jpg",
        "19": "https://www3.pictures.zimbio.com/gi/Celebrities+At+The+Lakers+Game+S45elxe3jm1x.jpg",
        "20": "https://media1.faz.net/ppmedia/video/603644964/1.109959/default/wieder-geladen-matrix-reloaded.jpg"
    },
    "ground_truth":{
        "1": {
            "people": 2,
            "man": 2,
            "woman": 0,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Barack Obama",
                    "predicate": "__background__",
                    "object": "Shinzo Abe"
                }
            ]
        },
        "2": {
            "people": 10,
            "man": 9,
            "woman": 1,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Barack Obama",
                    "predicate": "__background__",
                    "object": "Silvio Berlusconi"
                },
                {
                    "subject": "man",
                    "predicate": "__background__",
                    "object": "man"
                }
            ]
        },
        "3": {
            "people": 5,
            "man": 4,
            "woman": 1,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Barack Obama",
                    "predicate": "__background__",
                    "object": "Silvio Berlusconi"
                }
            ]
        },
        "4": {
            "people": 6,
            "man": 3,
            "woman": 3,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
```

```
            "statements": [
                {
                    "subject": "Barack Obama",
                    "predicate": "__background__",
                    "object": "Shinzo Abe"
                }
            ]
        },
        "5": {
            "people": 2,
            "man": 2,
            "woman": 0,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Donald Trump",
                    "predicate": "__background__",
                    "object": "Shinzo Abe"
                }
            ]
        },
        "6": {
            "people": 6,
            "man": 6,
            "woman": 0,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Lance Armstrong",
                    "predicate": "riding",
                    "object": "bike"
                },
                {
                    "subject": "man",
                    "predicate": "riding",
                    "object": "bike"
                }
            ]
        },
        "7": {
            "people": 5,
            "man": 5,
            "woman": 0,
            "celebs": 2,
            "celeb_man": 2,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Zach Braff",
                    "predicate": "__background__",
                    "object": "Donald Faison"
                }
            ]
        },
        "8": {
            "people": 2,
            "man": 1,
            "woman": 1,
            "celebs": 2,
            "celeb_man": 1,
            "celeb_woman": 1,
            "statements": [
                {
                    "subject": "Kevin Bacon",
                    "predicate": "__background__",
                    "object": "Kyra Sedgwick"
                }
            ]
        },
        "9": {
            "people": 1,
            "man": 1,
            "woman": 0,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
```

```
                {
                    "subject": "Keanu Reeves",
                    "predicate": "playing",
                    "object": "guitar"
                }
            ]
        },
        "10": {
            "people": 3,
            "man": 1,
            "woman": 2,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Pierce Brosnan",
                    "predicate": "__background__",
                    "object": "mountain"
                }
            ]
        },
        "11": {
            "people": 3,
            "man": 2,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 0,
            "celeb_woman": 1,
            "statements": [
                {
                    "subject": "Edie Falco",
                    "predicate": "throws",
                    "object": "ball"
                }
            ]
        },
        "12": {
            "people": 3,
            "man": 2,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Tom Hanks",
                    "predicate": "sitting",
                    "object": "park bench"
                },
                {
                    "subject": "Tom Hanks",
                    "predicate": "besides",
                    "object": "man"
                },
                {
                    "subject": "woman",
                    "predicate": "sitting",
                    "object": "park bench"
                }
            ]
        },
        "13":{
            "people": 1,
            "man": 0,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 0,
            "celeb_woman": 1,
            "statements": [
                {
                    "subject": "Julia Roberts",
                    "predicate": "riding",
                    "object": "Horse"
                }
            ]
        },
        "14":{
            "people": 11,
            "man": 5,
            "woman": 6,
```

```
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Alex Rodriguez",
                    "predicate": "__background__",
                    "object": "sign"
                }
            ]
        },
        "15":{
            "people": 1,
            "man": 0,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 0,
            "celeb_woman": 1,
            "statements": [
                {
                    "subject": "Drew Barrymore",
                    "predicate": "eating",
                    "object": "food"
                }
            ]
        },
        "16":{
            "people": 4,
            "man": 4,
            "woman": 0,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {
                    "subject": "Ehud Olmert",
                    "predicate": "eating",
                    "object": "pizza"
                }
            ]
        },
        "17":{
            "people": 1,
            "man": 1,
            "woman": 0,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
                {

                    "subject": "Colin Farrell",
                    "predicate": "holding",
                    "object": "racket"
                }
            ]
        },
        "18":{
            "people": 3,
            "man": 2,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 0,
            "celeb_woman": 1,
            "statements": [
                {

                    "subject": "Victoria Beckham",
                    "predicate": "walking",
                    "object": "sidewalk"
                }
            ]
        },
        "19":{
            "people": 7,
            "man": 5,
            "woman": 1,
            "celebs": 1,
            "celeb_man": 1,
            "celeb_woman": 0,
            "statements": [
```

```
                {

                    "subject": "player",
                    "predicate": "__background__",
                    "object": "Danny Devito"
                }
            ]
        },
        "20":{
            "people": 2,
            "man": 1,
            "woman": 1,
            "celebs": 2,
            "celeb_man": 1,
            "celeb_woman": 1,
            "statements": [
                {

                    "subject": " Monica Bellucci",
                    "predicate": "__background__",
                    "object": "Keanu Reeves"
                }
            ]
        }
    }
}
```


Listing A.3: Experimental Setup

Figure A.1 shows the visualization of a scene graph representation for the image of Lance Armstrong riding a bike with a threshold for object detection score of t=0.4 and the top$_k$ predicates for k=1. Figure A.2 shows the visualization of a scene graph representation for the image of Lance Armstrong riding a bike with a threshold for object detection score of t=0.0 and the top$_k$ predicates for k=1. Figure A.1 lists the results of the computed link prediction scores and ranks by the link prediction model for the head, tail, and predicate parts of the analyzed triplets. These experiments were executed on the image of Lance Armstrong riding a bike. Table A.2 shows the statements or triplets extracted for a threshold for object detection score of t=0.4 and the top$_k$ predicates for k=1 of the image of Lance Armstrong riding a bike. Table A.3 shows the statements or triplets extracted for a threshold for object detection score of t=0.0 and the top$_k$ predicates for k=1 of the image of Lance Armstrong riding a bike. Table A.4 shows the statements or triplets extracted for a threshold for object detection score of t=0.6 and the top$_k$ predicates for k=5 of the image of Lance Armstrong riding a bike. Table A.5 shows the statements or triplets extracted for a threshold for object detection score of t=0.4 and the top$_k$ predicates for k=5 of the image of Lance Armstrong riding a bike. Table A.6 shows the statements or triplets extracted for a threshold for object detection score of t=0.6 and the top$_k$ predicates for k=20 of the image of Lance Armstrong riding a bike.

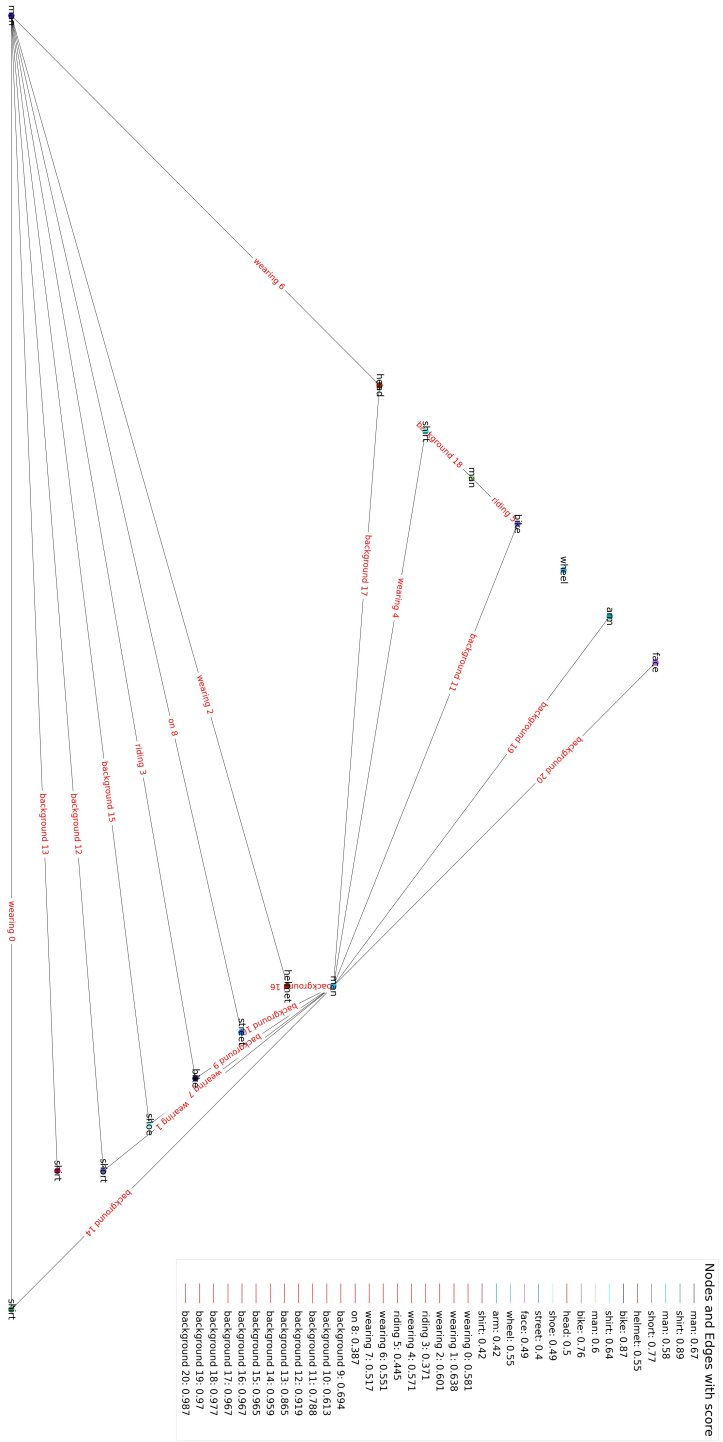



Figure A.1.: Scene Graph Visualization Lance Armstrong Riding a Bike (t=0.4, k=1)

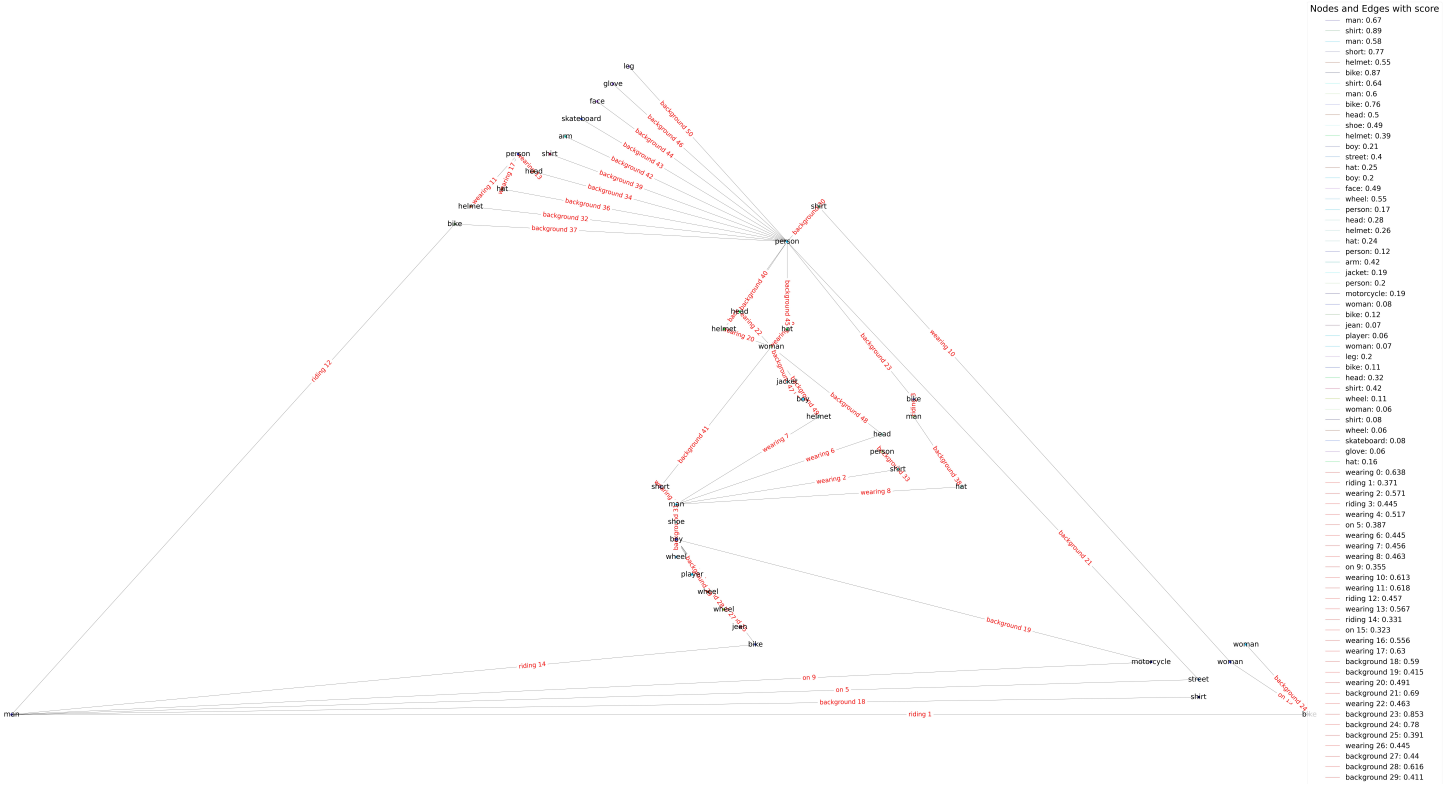



Figure A.2.: Scene Graph Visualization Lance Armstrong Riding a Bike (t=0.0, k=1)

| top_$k$ | 1 | 1 | 1 | 5 | 5 | 5 | 20 | 20 | 20 |
|---|---|---|---|---|---|---|---|---|---|
| threshold | 0 | 4 | 6 | 0 | 4 | 6 | 0 | 4 | 6 |
| Number of statements | 256 | 31 | 9 | 1280 | 155 | 45 | 5120 | 620 | 180 |
| Tail mean row number | $7.054 \times 10^{+4}$ | $3.499 \times 10^{+4}$ | $6.599 \times 10^{+2}$ | $8.637 \times 10^{+4}$ | $3.128 \times 10^{+4}$ | $1.366 \times 10^{+4}$ | $1.172 \times 10^{+5}$ | $5.032 \times 10^{+4}$ | $1.723 \times 10^{+4}$ |
| Tail median row number | 3066 | 1294 | 143 | 3324 | 1593 | 1294 | 8043 | 2757 | 1834 |
| Tail min row number | 13 | 18 | 18 | 13 | 18 | 18 | 13 | 18 | 18 |
| Tail max row number | $3.057 \times 10^{+6}$ | $3.311 \times 10^{+5}$ | $1.834 \times 10^{+3}$ | $4.456 \times 10^{+6}$ | $3.311 \times 10^{+5}$ | $6.218 \times 10^{+4}$ | $4.456 \times 10^{+6}$ | $1.489 \times 10^{+6}$ | $1.228 \times 10^{+5}$ |
| Tail mean score | -6.640 | -5.754 | -5.070 | -7.043 | -6.522 | -6.221 | -7.332 | -6.859 | -6.641 |
| Tail median score | -6.415 | -5.835 | -5.063 | -7.127 | -6.353 | -5.835 | -7.322 | -7.022 | -6.579 |
| Tail min score | -9.886 | -8.897 | -5.987 | -11.135 | -8.897 | -8.072 | -11.202 | -10.137 | -9.548 |
| Tail max score | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 |
| Head mean row number | $1.883 \times 10^{+5}$ | $2.936 \times 10^{+4}$ | $2.949 \times 10^{+2}$ | $2.237 \times 10^{+5}$ | $1.071 \times 10^{+5}$ | $9.342 \times 10^{+3}$ | $1.803 \times 10^{+5}$ | $8.950 \times 10^{+4}$ | $1.596 \times 10^{+4}$ |
| Head median row number | 5306 | 477 | 16 | 8568 | 1013 | 696 | 7447 | 1125 | 860 |
| Head min row number | 6 | 6 | 10 | 6 | 6 | 10 | 6 | 6 | 10 |
| Head max row number | $4.189 \times 10^{+6}$ | $3.326 \times 10^{+5}$ | $6.960 \times 10^{+2}$ | $4.331 \times 10^{+6}$ | $2.820 \times 10^{+6}$ | $3.095 \times 10^{+5}$ | $4.366 \times 10^{+6}$ | $2.820 \times 10^{+6}$ | $3.915 \times 10^{+5}$ |
| Head mean score | -6.640 | -5.754 | -5.070 | -7.043 | -6.522 | -6.221 | -7.332 | -6.859 | -6.641 |
| Head median score | -6.415 | -5.835 | -5.063 | -7.127 | -6.353 | -5.835 | -7.322 | -7.022 | -6.579 |
| Head min score | -9.886 | -8.897 | -5.987 | -11.135 | -8.897 | -8.072 | -11.202 | -10.137 | -9.548 |
| Head max score | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 |
| Predicate mean row number | 72.343 | 27.034 | 8.286 | 137.074 | 126.469 | 127.512 | 182.536 | 174.288 | 187.509 |
| Predicate median row number | 55 | 12 | 7 | 105 | 69 | 20 | 163 | 137 | 157 |
| Predicate min row number | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Predicate max row number | 399 | 248 | 20 | 437 | 377 | 369 | 545 | 545 | 545 |
| Predicate mean score | -6.640 | -5.754 | -5.070 | -7.043 | -6.522 | -6.221 | -7.332 | -6.859 | -6.641 |
| Predicate median score | -6.415 | -5.835 | -5.063 | -7.127 | -6.353 | -5.835 | -7.322 | -7.022 | -6.579 |
| Predicate min score | -9.886 | -8.897 | -5.987 | -11.135 | -8.897 | -8.072 | -11.202 | -10.137 | -9.548 |
| Predicate max score | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 | -4.257 |

Table A.1.: Link Prediction for Lance Armstrong Riding a Bike

| Subject Predicate Object |
|---|
| man wearing helmet |
| man background arm |
| man background bike |
| man wearing short |
| man background helmet |
| man background face |
| man background arm |
| man background shirt |
| man wearing shirt |
| man background shirt |
| man background bike |
| man background street |
| man wearing head |
| man background head |
| man background face |
| man background shirt |
| man background shoe |
| man background helmet |
| man riding bike |
| wheel background street |
| man background shirt |
| man wearing shirt |
| man background short |
| man riding bike |
| man background bike |
| man on street |
| man wearing shoe |
| man background shirt |
| man background bike |
| man background head |

Table A.2.: Statements for Lance Armstrong Riding a Bike (t=0.4, k=1)

| Subject Predicate Object | Subject Predicate Object | Subject Predicate Object | Subject Predicate Object |
|---|---|---|---|
| man background head | boy background arm | boy background bike | wheel background street |
| boy background helmet | player background head | woman background helmet | woman wearing hat |
| man background hat | man background helmet | person on street | boy background face |
| boy background wheel | boy wearing jacket | woman background short | person background street |
| boy background head | person background shirt | boy background shirt | person background leg |
| person background head | woman background shoe | boy background shirt | person wearing helmet |
| player background hat | boy background face | player background bike | man background bike |
| woman background short | man background shirt | man background hat | boy wearing head |
| person background shirt | person background hat | person riding bike | woman wearing hat |
| player background head | man riding bike | jean wearing head | boy background hat |
| woman background helmet | boy background arm | boy on bike | woman background helmet |
| woman background head | man wearing shirt | man background head | man background head |
| motorcycle background hat | boy background glove | person background bike | woman background hat |
| boy background shirt | boy background head | person wearing hat | person wearing hat |
| woman riding bike | person riding bike | woman wearing hat | boy background skateboard |
| man background arm | woman background street | person background head | motorcycle background bike |
| person background helmet | man background face | man background shoe | man wearing helmet |
| person background jacket | man background leg | player wearing hat | player wearing helmet |
| woman on bike | jean wearing shirt | person background glove | woman background street |
| man background helmet | boy background leg | woman background jacket | man background head |
| person wearing shoe | woman background head | boy background skateboard | woman background hat |
| person background jean | woman wearing head | person background hat | man wearing helmet |
| man background shirt | woman riding bike | boy background street | woman background bike |
| woman background head | man background helmet | man background bike | woman background helmet |
| man background hat | woman background bike | man background bike | motorcycle background head |
| woman wearing head | boy background motorcycle | boy background head | woman background bike |
| person wearing shirt | boy background hat | man background helmet | player wearing shoe |
| jean background bike | person background arm | man background short | woman background head |
| man background wheel | person background hat | player wearing short | person wearing shirt |
| man background bike | person background helmet | person background wheel | person background shirt |
| man riding bike | man riding bike | person wearing head | boy background wheel |
| man background glove | motorcycle background hat | boy background shoe | woman wearing shirt |
| person background hat | boy background bike | person wearing jacket | person background face |
| man background wheel | woman background head | player background street | boy background leg |
| man background hat | boy wearing hat | woman wearing short | boy riding bike |
| man background face | man riding bike | woman background hat | man wearing helmet |
| person background skateboard | jean wearing helmet | motorcycle background shirt | man background hat |
| boy background bike | boy background short | person background short | person background bike |
| man background bike | player background shirt | person background wheel | boy background hat |
| person background helmet | man wearing jacket | man background street | person background helmet |
| jean wearing hat | person background leg | man background glove | woman background hat |
| jean background hat | boy background jean | man background skateboard | boy background bike |
| person background arm | person background street | player wearing head | jean background shirt |
| woman background face | person background bike | woman wearing helmet | person background face |
| person background shoe | boy background hat | boy wearing shoe | person wearing short |
| woman background skateboard | player background hat | person background head | man background arm |
| player background bike | person background helmet | person background shirt | man wearing short |
| player wearing shirt | boy wearing short | man wearing hat | boy background helmet |
| person on motorcycle | person background glove | man wearing shirt | boy background helmet |
| person background shirt | boy wearing hat | man background leg | boy background street |
| person riding bike | person riding bike | boy background bike | woman wearing helmet |
| person background head | boy wearing helmet | man background shirt | man wearing head |
| woman background shirt | boy background jacket | person background bike | man background head |
| woman background jacket | boy wearing shirt | man background bike | boy wearing shirt |
| man background helmet | man background shirt | man background street | man wearing head |
| motorcycle background shirt | person wearing head | woman background shirt | motorcycle background helmet |
| woman background hat | player background bike | woman background shirt | boy wearing head |
| man background hat | woman wearing shirt | boy wearing helmet | man on street |
| person background shirt | woman background helmet | man wearing hat | boy background helmet |
| woman background bike | man wearing shoe | man background shirt | person background bike |
| woman background shirt | boy background head | player wearing jacket | man background jacket |
| man background jean | woman background street | person background skateboard | person wearing helmet |
| boy background glove | player background helmet | player background helmet | man background shirt |
| person background hat | man on motorcycle | man background skateboard | person background head |

Table A.3.: Statements for Lance Armstrong Riding a Bike (t=0.0, k=1)

| Subject Predicate Object |
|---|
| man wearing shirt |
| man in shirt |
| man wears shirt |
| man background shirt |
| man has shirt |
| man wears shirt |
| man background shirt |
| man in shirt |
| man has shirt |
| man wearing shirt |
| man on bike |
| man above bike |
| man sitting on bike |
| man riding bike |
| man background bike |
| man background shirt |
| man wearing shirt |
| man in shirt |
| man on shirt |
| man wears shirt |
| man on bike |
| man above bike |
| man sitting on bike |
| man background bike |
| man riding bike |
| man on bike |
| man background bike |
| man riding bike |
| man sitting on bike |
| man above bike |
| man background bike |
| man riding bike |
| man on bike |
| man above bike |
| man sitting on bike |
| man has short |
| man wearing short |
| man wears short |
| man in short |
| man background short |
| man wearing shirt |
| man on shirt |
| man background shirt |
| man in shirt |
| man wears shirt |

Table A.4.: Statements for Lance Armstrong Riding a Bike (t=0.6, k=5)

| Subject Predicate Object | Subject Predicate Object | Subject Predicate Object | Subject Predicate Object |
|---|---|---|---|
| man on bike | man background street | man wears head | man wears shirt |
| man above bike | man has short | man background head | man background shirt |
| man sitting on bike | man background short | man with face | man has shirt |
| man riding bike | man wearing short | man on face | man has shirt |
| man background bike | man in short | man background face | man background shirt |
| man wearing helmet | man wears short | man wearing face | man wearing shirt |
| man has helmet | man wears shirt | man has face | man in shirt |
| man in helmet | man has shirt | man background shirt | man on shirt |
| man background helmet | man background shirt | man wears shirt | man sitting on street |
| man wears helmet | man in shirt | man wearing shirt | man in street |
| man wearing shoe | man wearing shirt | man on shirt | man on street |
| man riding shoe | man on face | man in shirt | man background street |
| man background shoe | man with face | man riding street | man riding street |
| man on shoe | man has face | man sitting on street | man background shirt |
| man has shoe | man background face | man in street | man has shirt |
| man has arm | man wearing face | man background street | man in shirt |
| man wearing arm | man near bike | man on street | man wears shirt |
| man with arm | man riding bike | man wears helmet | man wearing shirt |
| man in arm | man sitting on bike | man has helmet | man on bike |
| man background arm | man on bike | man wearing helmet | man riding bike |
| man in short | man background bike | man background helmet | man sitting on bike |
| man has short | man above bike | man in helmet | man background bike |
| man wearing short | man riding bike | man background shirt | man above bike |
| man wears short | man on bike | man in shirt | man wearing shoe |
| man background short | man sitting on bike | man wearing shirt | man wears shoe |
| man on bike | man background bike | man wears shirt | man on shoe |
| man sitting on bike | wheel background street | man has shirt | man background shoe |
| man background bike | wheel near street | man has helmet | man has shoe |
| man riding bike | wheel in street | man in helmet | man background head |
| man above bike | wheel of street | man wearing helmet | man has head |
| man on arm | wheel on street | man background helmet | man wearing head |
| man has arm | man in head | man wears helmet | man in head |
| man wearing arm | man has head | man background shirt | man with head |
| man in arm | man wears head | man wearing shirt | man background bike |
| man background arm | man wearing head | man wears shirt | man on bike |
| man standing on street | man background head | man in shirt | man holding bike |
| man riding street | man in head | man on shirt | man near bike |
| man in street | man wearing head | man wearing shirt | man riding bike |
| man on street | man has head | man in shirt | |

Table A.5.: Statements for Lance Armstrong Riding a Bike (t=0.4, k=5)

| Subject Predicate Object | Subject Predicate Object | Subject Predicate Object | Subject Predicate Object |
|---|---|---|---|
| man laying on shirt | man wearing bike | man above bike | man riding bike |
| man with shirt | man behind bike | man with bike | man behind bike |
| man holding shirt | man of bike | man in bike | man of bike |
| man background shirt | man using bike | man sitting on bike | man above bike |
| man of shirt | man has bike | man riding bike | man walking on bike |
| man in shirt | man standing on bike | man carrying bike | man behind shirt |
| man and shirt | man on bike | man wears bike | man background shirt |
| man above shirt | man watching bike | man wearing bike | man and shirt |
| man wearing shirt | man holding bike | man in front of bike | man riding shirt |
| man behind shirt | man looking at bike | man background bike | man wearing shirt |
| man wears shirt | man sitting on bike | man of bike | man standing on shirt |
| man in front of shirt | man with bike | man on bike | man in front of shirt |
| man near shirt | man background bike | man in bike | man looking at shirt |
| man on shirt | man at bike | man has bike | man with shirt |
| man has shirt | man in front of bike | man riding bike | man wears shirt |
| man carrying shirt | man at short | man wearing bike | man has shirt |
| man sitting on shirt | man with short | man sitting on bike | man watching shirt |
| man under shirt | man riding short | man watching bike | man under shirt |
| man standing on shirt | man near short | man walking on bike | man above shirt |
| man riding shirt | man in short | man holding bike | man in shirt |
| man with shirt | man looking at short | man in front of bike | man at shirt |
| man near shirt | man in front of short | man near bike | man near shirt |
| man above shirt | man behind short | man behind bike | man on shirt |
| man carrying shirt | man carrying short | man standing on bike | man sitting on shirt |
| man sitting on shirt | man and short | man background bike | man holding shirt |
| man background shirt | man above short | man with bike | man at shirt |
| man wears shirt | man has short | man above bike | man wearing shirt |
| man in front of shirt | man holding short | man on back of bike | man on shirt |
| man holding shirt | man wearing short | man at bike | man standing on shirt |
| man watching shirt | man sitting on short | man looking at bike | man watching shirt |
| man at shirt | man wears short | man in bike | man near shirt |
| man and shirt | man on short | man standing on bike | man above shirt |
| man riding shirt | man standing on short | man near bike | man in front of shirt |
| man wearing shirt | man background short | man in front of bike | man behind shirt |
| man behind shirt | man watching short | man at bike | man riding shirt |
| man on shirt | man has bike | man on bike | man looking at shirt |
| man in shirt | man on bike | man using bike | man in shirt |
| man looking at shirt | man near bike | man has bike | man holding shirt |
| man standing on shirt | man using bike | man sitting on bike | man wears shirt |
| man has shirt | man at bike | man holding bike | man carrying shirt |
| man in bike | man holding bike | man looking at bike | man with shirt |
| man near bike | man behind bike | man with bike | man background shirt |
| man wears bike | man standing on bike | man background bike | man has shirt |
| man above bike | man of bike | man watching bike | man sitting on shirt |
| man riding bike | man looking at bike | man wearing bike | man and shirt |

Table A.6.: Statements for Lance Armstrong Riding a Bike (t=0.6, k=20)