A tailored Handwritten-Text-Recognition System for Medieval Latin

Philipp Koch[1♢] **Gilary Vera Nuñez**[1♢] **Esteban Garces Arias**[1♠]
Christian Heumann[1♠] **Matthias Schöffel**[2♣] **Alexander Häberlin**[2,3♡]
Matthias Aßenmacher[1,4♠]

[1] Department of Statistics, LMU, Munich, Germany
[2] Bavarian Academy of Sciences, BAdW, Munich, Germany
[3] Universität Zürich, Zurich, Switzerland
[4] Munich Center for Machine Learning (MCML), LMU, Munich, Germany

♢{philipp.koch,gi.vera}@campus.lmu.de ♣matthias.schoeffel@badw.de ♡alexander.haeberlin@sglp.uzh.ch
♠{esteban.garcesarias,chris,matthias}@stat.uni-muenchen.de

Abstract

The Bavarian Academy of Sciences and Humanities aims to digitize its Medieval Latin Dictionary. This dictionary entails record cards referring to lemmas in medieval Latin, a low-resource language. A crucial step of the digitization process is the Handwritten Text Recognition (HTR) of the handwritten lemmas found on these record cards. In our work, we introduce an end-to-end pipeline, tailored to the medieval Latin dictionary, for locating, extracting, and transcribing the lemmas. We employ two state-of-the-art (SOTA) image segmentation models to prepare the initial data set for the HTR task. Furthermore, we experiment with different transformer-based models and conduct a set of experiments to explore the capabilities of different combinations of vision encoders with a GPT-2 decoder. Additionally, we also apply extensive data augmentation resulting in a highly competitive model. The best-performing setup achieved a Character Error Rate (CER) of 0.015, which is even superior to the commercial Google Cloud Vision model, and shows a more stable performance.

1 Introduction

The Medieval Latin Dictionary (MLW)[1] deals with Latin texts that were created between 500 and 1280 in the German-speaking region. The foundations for this project have been developed from 1948 onwards and since then, the dictionary has been continuously published in individual partial editions since 1959. The basis of the dictionary consists of 50 selected texts that have been fully transcribed onto DIN-A6 record cards (cf. Fig. 1) constituting about 40% of the note material. Later, another 2,500 texts were excerpted and transcribed manually onto DIN-A6 record cards, using a typewriter. In addition, there are so-called "index cards", a type of record card, that helps to uncover often

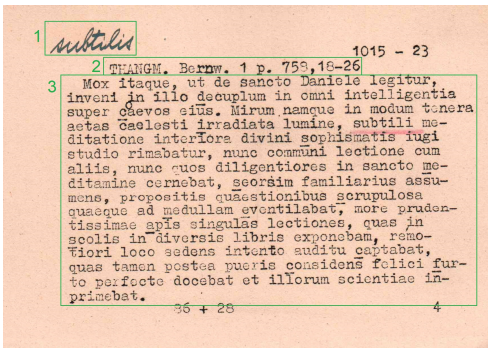


Figure 1: Record card from the MLW data set.

hundreds of additional references. In total, it is estimated that 1.3 million reference points have been recorded for the MLW. These record cards were sorted alphabetically by the first letter of the keyword (lemma), and serve as the foundation for creating a dictionary. Around 200,000 record cards have been scanned and annotated with their respective lemma. The accurate extraction and transcription of the lemma present a challenge, which is further compounded by the limited resources available for medieval Latin.

Our contributions are as follows: **(1)** We present a novel end-to-end HTR pipeline specifically designed for detecting and transcribing handwritten medieval Latin text. Notably, it surpasses commercial applications currently considered SOTA for related tasks. **(2)** We train a lemma-detection model without relying on human-annotated bounding boxes. **(3)** We conduct extensive experiments to compare various vision encoders and evaluate the effectiveness of data augmentation techniques.

2 Related Work

We provide an overview for HTR, which is the main challenge of this work. For object detection, which is an intermediate step of this work, we refer to Zaidi et al. (2021) for a detailed overview.

Connectionist Temporal Classification (CTC)

[1] In German: ***M**ittel**l**ateinisches **W**örterbuch (MLW)*

Proceedings of the Ancient Language Processing Workshop associated with RANLP-2023, pages 103–110,
held in Varna Bulgaria, Sept 8, 2023.
https://doi.org/10.26615/978-954-452-087-8_012

(Graves et al., 2006) is a technique in which a neural network – initially a Recurrent Neural Network (RNN) but other networks might also be used (Chaudhary and Bali, 2022) – is trained to predict a matrix of conditional transition probabilities. The input image, represented as a vector representation through a Convolutional Neural Network (CNN), is fed to the network, and for each input (i.e. the activation maps of the CNN) the network predicts the character. CTC, combined with CNNs and RNNs, often yielded competitive results, such as shown by Puigcerver (2017) and Bluche and Messina (2017). Furthermore, approaches applying only CNNs and CTC also exist (Chaudhary and Bali, 2021, 2022). Easter2.0 achieved competitive results on IAM (Marti and Bunke, 2002), a frequently used HTR data set consisting of English handwritten text.

A recent work that achieved SOTA results on IAM is TrOCR (Li et al., 2022), based on the transformer (Vaswani et al., 2017), consisting of a vision encoder and a text decoder. This deviated from previous approaches where primarily CNNs and RNNs were used. This development is closely linked to the emergence of the transformer in the vision domain (Dosovitskiy et al., 2021; Bao et al., 2022). Barrere et al. (2022) introduce another transformer model also using CTC, with the main difference to TrOCR being a different embedding technique for visual features based on a CNN. The results have also been shown to be competitive on the IAM data set. Diaz et al. (2021) compare encoder-decoder models' performance on HTR, using different models in the encoder and decoder parts, e.g. a transformer encoder plus a CTC-based decoder. Furthermore, they found that enriching this architecture with a language model yields SOTA results on IAM. The TrOCR framework has already been successfully applied to historical data akin to our task. Ströbel et al. (2022) fine-tune a TrOCR instance to handwritten Latin from the 16th century (Stotz and Ströbel, 2021, referred to as *Gwalther*), achieving competitive results.

3 Data

Our data set comprises 114,653 images, holding 3,507 distinct lemmas. All images are in RGB, but not uniform in size. The information on the corresponding lemma is available on the image level. Most lemmas start with the letter `"s"`, followed by a large number of lemmas starting with the letters `"m"`, `"v"`, `"t"`, `"u"`, `"l"`, and `"n"`. We observe lemmas from a length of one character up to 19 characters, with an average length between five and six characters. A total of 2,420 lemmas (69%) appear on ten record cards or less; 854 lemmas (24.4%), on 10 to 100 record cards, and just 233 lemmas (6.6%) on more than 100 record cards. 1,123 lemmas (36.7%) only occur on one card.

4 Lemma Extraction Pipeline

4.1 Visual Detection

Since the lemmas are always located in the upper left corner, but not annotated with their exact locations, training a custom object detection model for extraction is not feasible. In order to still retrieve the locations of the bounding boxes for some lemmas, we use the One For All (OFA) transformer (Wang et al., 2022), fine-tuned on RefCOCO (Kazemzadeh et al., 2014). To ensure the quality of the extracted lemma, we experiment with multiple prompts and examine their results (cf. Appendix A). After obtaining a training data set of 20,000 instances, we train a YOLOv8 model (Jocher et al., 2023) based on the You Only Look Once (YOLO) architecture (Redmon et al., 2016). The model predictions from our YOLO model, are then subject to two post-processing steps to ensure the quality of the images:

For 17,674 images (15.42% of the data), the model predicted **multiple bounding boxes**. We visually examined the cases and found that other handwritten text was often recognized as a lemma, sometimes scattered throughout the record cards (cf. Fig. 5, Appendix B). We further visually examined 202 cases where **no bounding box** was detected, stemming mostly from machine writing or scanning errors. For some images that follow the standard layout of the record cards, the model also failed. We disregard this set constituting less than 0.2% of the entire data set.

Taking all aspects into account, we introduce two rules to determine the appropriate bounding box: (1) choose the largest bounding box in (2) the upper left quarter of the entire image. The result after applying these rules is displayed in Figure 6 (Appendix B). The final data set consists of 114,451 samples, exhibiting a difference of the 202 samples to the initial 114,653 image-label pairs. We make our data available on HuggingFace.[2]

[2] https://huggingface.co/misoda

4.2 HTR Model

We use a transformer as the main model akin to TrOCR. For the encoder, we consider three different architectures, while we use GPT-2 (Radford et al., 2019) as a decoder model for all setups. All models are trained from scratch, although we use pre-trained image processors for the encoder models and train a tokenizer for our custom alphabet.

Tokenizer We use a customized byte-level BPE tokenizer (Sennrich et al., 2016) (trained on the labels from our data) for the dictionary's vocabulary.

Vision Encoders We consider three different encoder architectures, namely Vision Transformer (ViT) (Dosovitskiy et al., 2021), Bidirectional Encoder representation for Image Transformers (BEiT) (Bao et al., 2022), and Shifted Window Transformer (Swin) (Liu et al., 2021).

Text Decoder We use the GPT-2 (Radford et al., 2019) architecture, a decoder-only transformer, which we train from scratch, i.e., we do not use the pre-trained weights since we deal with a specific task in a low-resource language setting.

Implementation Details We use the HuggingFace transformers library (Wolf et al., 2020) and PyTorch (Paszke et al., 2019) to train the HTR pipeline. Our codebase, containing all scripts (experiments and training) is available via GitHub[3], and the final model is on pypi.[4] All the experiments were conducted using a Tesla V100 GPU (16 GB).

5 Experiments

5.1 Standard Training Settings

After shuffling the data, we randomly split it into a train (85% – 97,283 samples) and a test (15% – 17,168 samples) set. In the train split, 94.53% (3,315) of the lemmas are present. For all training procedures, we use the AdamW optimizer (Loshchilov and Hutter, 2019) and did not engage in hyperparameter tuning. Further details are reported in Appendix C. For standard training, the model is trained using a data set that includes the cut images from the record cards as input and their respective lemmas as the labels to be predicted. We train each of the models for a total of 5 epochs. We assess the model performance using the CER, which is computed by summing up edit operations and dividing by the label length. To account for the varying length, we further utilize the weighted CER.

[3] https://github.com/slds-lmu/mlw-htr

[4] https://pypi.org/project/mlw-lectiomat/

5.2 Data Augmentation

To increase the diversity of the training data, we apply random rotation, blurring, or modifications related to color perception. For the augmentation setting, we increase the number of epochs to 20 (compared to 5 for the standard training). We use three different augmentation pipelines, one of which is randomly chosen with $p = \frac{1}{3}$.

Pipeline A applies blurring and modifications to sharpness. The intensity of these modifications is determined randomly and can range from no modification to higher intensity. **Pipeline B** alters brightness, contrast, saturation, sharpness, and hue. The specific alterations for each instance are again determined randomly, also including the possibility of no modifications at all. **Pipeline C** combines the modifications from the previous two. In addition to the described techniques, all augmentation pipelines include random masking, where rectangles of the images are blackened, and random rotation within a range of -10 to 10 degrees.

Decoder Pre-Training We experiment with decoder pre-training (10 epochs) on a corpus of the concatenated lemmas to incorporate prior knowledge about medieval Latin. After pre-training, we combine it with the encoder and continue training for 20 epochs as described in Section 5.1, using the same augmentation techniques outlined before.

5.3 Experimental Results

The main results of our work are reported in Table 1. The BEiT+GPT-2 architecture achieved the best results in case of the standard training regime, exhibiting a CER of 0.258, followed by Swin+GPT-2 (0.349) and ViT+GPT-2 (0.418). Applying the augmentation pipelines notably improves model performance compared to the standard training for all three models. The best model with augmentation is Swin+GPT-2, achieving a CER of 0.017. As for the other two models, the CER is 0.073 for ViT+GPT-2 and 0.110 for BEiT+GPT-2.

	ViT	Swin	BEiT
Standard	0.418	0.349	0.258
+ Data Augmentation	0.073	**0.017**	0.110
+ Decoder Pre-Training	0.049	0.018	0.114

Table 1: CERs for different encoder configurations.

Pre-training of the decoder does, on average, not lead to further improvement. ViT+GPT-2 is the exception, for which the CER drops to 0.049. We

observe no improvements for the other models. To summarize, the best results are achieved when using a Swin+GPT-2 model with data augmentations, reaching a CER value of 0.017.

5.4 Ablation Study

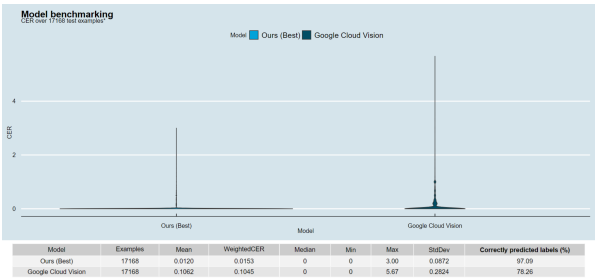
To investigate the impact of data augmentation, we perform three ablations, removing individual steps from the augmentation pipelines. To quantify the individual effects of each augmentation technique, we train the model without a specific augmentation method and report the resulting CER.

Swin+GPT-2 (Full augmentation pipelines)	0.017
w/o masking augmentation	**0.015**
w/o rotation augmentation	0.021
w/o color augmentation	0.017

Table 2: CER-Results of different model configurations.

Excluding the masking step from the pipeline leads to an actual improvement of model performance while excluding random rotations or color-related augmentations results do not (cf. Tab. 2).

5.5 Google Cloud Vision Comparison



Model	Examples	Mean	WeightedCER	Median	Min	Max	StdDev	Correctly predicted labels (%)
Ours (Best)	17168	0.0120	0.0153	0	0	3.00	0.0872	97.09
Google Cloud Vision	17168	0.1062	0.1045	0	0	5.67	0.2824	78.26

Figure 2: Comparison of Swin+GPT-2 to GCV.

To compare the results of our model, we decided to use Google Cloud Vision (GCV) a highly competitive HTR model, which has proven effective in practical applications (Thammarak et al., 2022). GCV often predicts extra characters and/or suffixes that are not part of the true lemma, which is why we post-process the predictions by GCV for a fair comparison by deleting extra characters and words after the first word or after a `'-'` or a `'('`. Figure 2 shows the comparison of our model with GCV. The violin plots of the (unweighted) CERs show a concentration of the CER values around 0 for both models. For our model, the most extreme values are at a CER of 3, for GCV the maximum is nearly twice as high and we observe an overall higher standard deviation compared to our model. To conclude, our best model exhibits a weighted CER of 0.0153, while GCV only reaches 0.1045. Overall, our model correctly predicts 97,09% of all lemmas, while GCV only does so for 78.26%.

5.6 Performance of other HTR systems

Table 3 illustrates the CERs of other systems on different HTR data sets. Ströbel et al. (2022) use Gwalther, while all other papers evaluate their systems on IAM. Our model achieves the lowest CER. However, it must be considered that we did not evaluate the same data set, which makes a direct comparison impossible. In contrast to the other transformer-based models, our best model uses Swin as an encoder.

Model	CER	Data set	Architecture
Ours (Best)	**0.0153**	MLW	Transformer
TrOCR Large (Ströbel et al., 2022)	0.0255	Gwalther	Transformer
TrOCR Large (Li et al., 2022)	0.0289	IAM	Transformer
EASTER2.0 (Chaudhary and Bali, 2022)	0.0621	IAM	CNN+CTC
Light Transformer (Barrere et al., 2022)	0.0570	IAM	CNN+Transformer
Self-Att.+CTC+LM (Diaz et al., 2021)	0.0275	IAM	Trf.+CTC+ LM

Table 3: Performance of contemporary HTR systems.

6 Conclusion and Outlook

Since the record cards include much more information than the one we extracted, we recommend further research into various extraction techniques. With the recent publication of Segment Anything Model, Kirillov et al. (2023) introduce a model that might be able to extract further features from the record cards with much higher accuracy.

We present a novel end-to-end pipeline for the Medieval Latin dictionary. Our library includes an image-detection-based model for lemma extraction and a tailored HTR model. We experiment with training different configurations of transformers using the ViT, BEiT, and Swin encoders while using a GPT-2 decoder. Employing data augmentation, our best model (Swin+GPT-2) achieves a CER of 0.015. The evaluation of the results exhibits a weaker performance on longer lemmas and on lemmas that appear less frequently in the training data. Further experiments with generative models to produce synthetic data (not reported in the paper) were not successful, however, we recommend further research into this direction. To conclude, our approach presents a promising HTR solution for Medieval Latin. Future research can build upon our work, and explore its generalizability to other languages and data sets by making use of our pip-installable Python package.

Limitations

Our approach has several limitations that can be addressed to improve its efficiency further. There are issues regarding the data set (cf. Sec. 3) that might be reflected in the model's performance. As discussed in Section 3, some lemmas are stroked out partially or entirely, introducing a notable noise to the data. Further, handwritten comments or other annotations have been added to some of the record cards, and some images are not correctly labeled, which might have distorted the recognition capabilities of our model.

Since our pipeline was mostly trained on data from the *S*-series of the dictionary, many words starting with other letters were not seen by the model during training. Therefore, the performance of the proposed approach, when applied to other series, remains somewhat uncertain. As elaborated in section 6, the model tends to perform weaker on unseen lemmas. Further, there are indications that the model might perform worse on longer lemmas.

The lemma-detection model (YOLOv8) is not guaranteed to predict the correct bounding box for the lemma consistently. Errors at this early stage of the pipeline may severely impact the result. Although the failure rate for the training dataset in which no bounding box was predicted is close to zero, the problem can still appear during inference.

We did neither experiment with the initial TrOCR architecture nor did we fine-tune a pretrained TrOCR instance for this task. However, the results of Ströbel et al. (2022) suggest a strong performance of TrOCR. Thus, we recommend training it on the MLW data set.

Ethics Statement

We affirm that our research adheres to the ACL Ethics Policy. This work involves the use of publicly available data sets and does not involve human subjects or any personally identifiable information. We declare that we have no conflicts of interest that could potentially influence the outcomes, interpretations, or conclusions of this research. All funding sources supporting this study are acknowledged. We have made our best effort to document our methodology, experiments, and results accurately and are committed to sharing our code, data, and other relevant resources to foster reproducibility and further advancements in research.

Acknowledgements

We would like to thank the three anonymous referees for their constructive comments and suggestions, which helped improve this paper considerably. We wish to thank the Bavarian Academy of Sciences for providing us with the guidance and required access to the handwritten material. This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of BERD@NFDI - grant number 460037581.

References

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. Beit: Bert pre-training of image transformers.

Killian Barrere, Yann Soullard, Aurélie Lemaitre, and Bertrand Coüasnon. 2022. A light transformer-based architecture for handwritten text recognition. In *Document Analysis Systems*, pages 275–290, Cham. Springer International Publishing.

Théodore Bluche and Ronaldo Messina. 2017. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 646–651.

Kartik Chaudhary and Raghav Bali. 2021. Easter: Simplifying Text Recognition using only 1d Convolutions. *Proceedings of the Canadian Conference on Artificial Intelligence*. Https://caiac.pubpub.org/pub/fm5sy88o.

Kartik Chaudhary and Raghav Bali. 2022. Easter2.0: Improving convolutional models for handwritten text recognition.

Daniel Hernandez Diaz, Siyang Qin, Reeve Ingle, Yasuhisa Fujii, and Alessandro Bissacco. 2021. Rethinking text line recognition models.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. YOLO by Ultralytics.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, Doha, Qatar. Association for Computational Linguistics.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment anything.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2022. Trocr: Transformer-based optical character recognition with pre-trained models.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Urs-Viktor Marti and H. Bunke. 2002. The iam-database: An english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Joan Puigcerver. 2017. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 67–72.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units.

Peter Stotz and Phillip Ströbel. 2021. bullinger-digital/gwalther-handwriting-ground- truth: Initial release.

Phillip Benjamin Ströbel, Simon Clematide, Martin Volk, and Tobias Hodel. 2022. Transformer-based htr for historical documents.

Karanrat Thammarak, Prateep Kongkla, Yaowarat Sirisathitkul, and Sarun Intakosum. 2022. Comparative analysis of tesseract and google cloud vision for thai vehicle registration certificate. *International Journal of Electrical and Computer Engineering*, 22:1849–1858.

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

Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. 2021. A survey of modern deep learning based object detection models.

Appendix

A Annotatong the Bounding Boxes

This Appendix holds the details of the Visual Detection part of the pipeline, described in Section 4.1, and the challenges we were confronted with.

A.1 The Task

To annotate the bounding boxes, the model is provided with a prompt describing the lemma and the image. The model then returns a bounding box for the requested object, which is the lemma in our case. Different prompts are described in Table 4.

Prompt 1		`Cursive text upper left`
Prompt 2		`Handwritten cursive word upper left`
Prompt 3	Length: 1-5:	`Blue drawing in the upper left`
	Other:	`Handwritten cursive word upper left`
Prompt 4	Length: 1-6:	`Blue drawing in the upper left`
	Other:	`Handwritten cursive word upper left`

Table 4: Different prompts used for OFA.

A.2 Assumption about Bounding Boxes

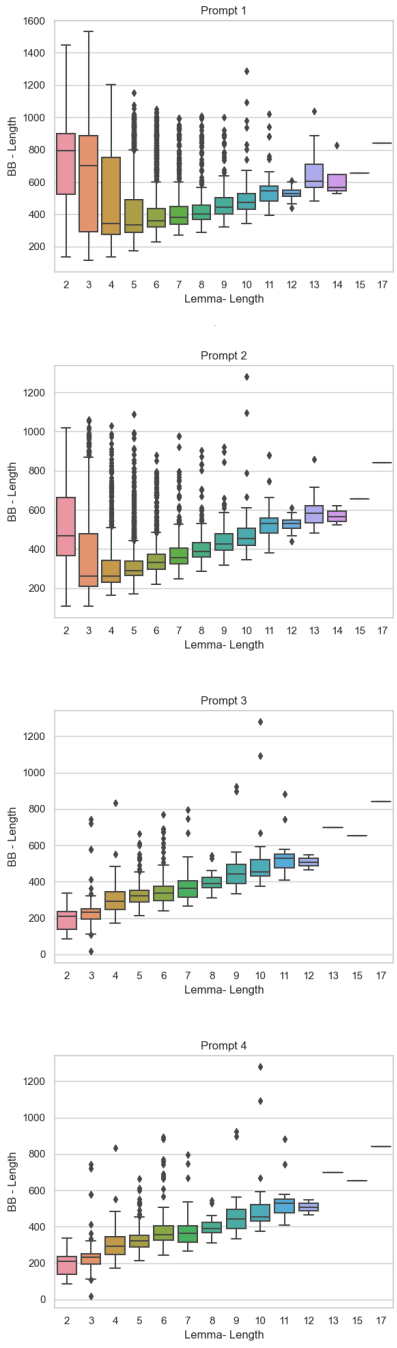
Since we do not have any ground truth about the bounding boxes, we rely on heuristics to verify the correctness of the boxes. One such heuristic is the assumed linear relationship between the lemma length and the bounding box's width. While the height of the boxes is assumed to be similar across instances, the lemma length must significantly impact the bounding box's width. To verify the results of the annotation process, we use box plots to visualize the relationship between lemma length and width (cf. Fig. 3a – 3d).

A.3 Initial Implementation and Results

We use the RefCOCO-OFA model[5] and modify it four our purposes. Prompt one (cf. Tab. 4) is used to obtain the lemmas for all images.

After running the model on the first instances with *Prompt 1*, we find that the relationship between the box's width and the lemma length does not look as expected. Figure 3 illustrates this problem. Investigating the short lemmas, we observe that the model often fails to annotate the record cards appropriately. Often other textual objects are annotated, or the bounding box stretches throughout the entire record card.

[5] Huggingface: `OFA-Base-RefCOCO`



(d) Fourth and final Prompt

Figure 3: Box-Plots for the width of the bounding boxes based on the lemma's length.

A.4 Two Different Prompts for Shorter and Longer Lemmata

After different experiments, *Prompt 2* turned out to work appropriately for shorter lemmas, but was, however, not suitable for longer ones. To combine the strength of both prompts, we apply a conditional prompt based on the length of the lemma using different cut-offs (5 or 6 characters). We find that using *Prompt 4* is the best-suited approach. The analysis of the relationship between the bound-

ing box widths and the length of the lemma for different prompts can be seen in Figure 3.

B YOLO: Training and Inference

B.1 Training Results

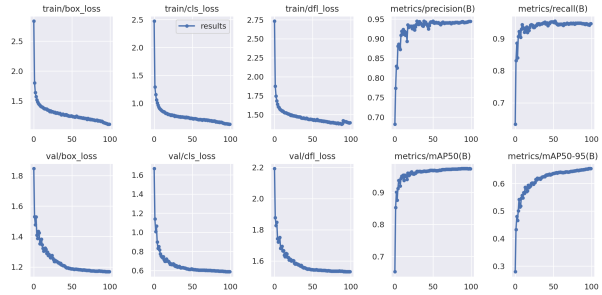


Figure 4: YOLO Training Results.

B.2 Multiple Lemmas Detected by YOLO

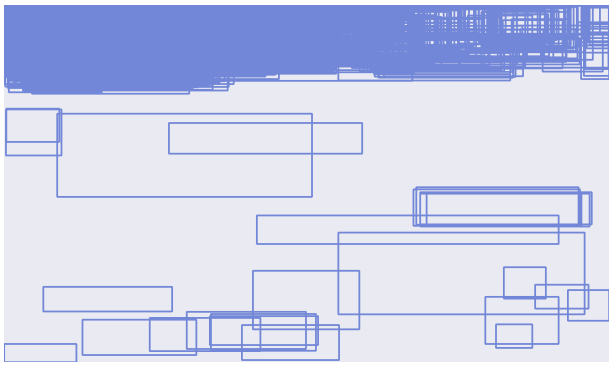


Figure 5: All bounding boxes from instances where YOLO has detected more than one bounding box.

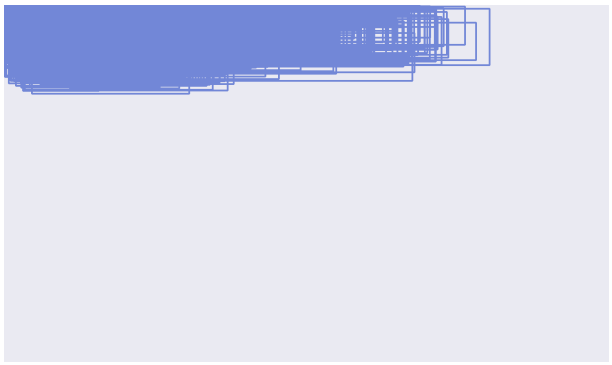


Figure 6: Bounding boxes of all instances to which the rule *largest bounding box in the upper left corner* was applied to.

C Training details

We used the defaults from `transformers` (4.26.1), if not reported otherwise.

C.1 Standard Training

Parameter	Value
Seed	42
Optimizer	AdamW
Epochs	5
Decoder	GPT-2
Encoder	{BEIT, Swin, ViT}
Batch Size (Train & Test)	64

Table 5: Parameters for the standard training.

C.2 Training with Augmentation

Parameter	Value
Seed	42
Optimizer	AdamW
Epochs	{5, 20}
Decoder	GPT-2
Encoder	{BEIT, Swin, ViT}
Batch Size (Train & Test)	64

Table 6: Parameters for training with augmentation.

C.3 Natural Language Generation

Parameter	Value
Max Length	32
Early Stopping	True
No Repeat Ngram Size	3
Length Penalty	2.0
Number of Beams	4

Table 7: Parameters for natural language generation.

C.4 Decoder Pre-Training

Parameter	Value
Seed	42
Epochs	10
Batch Size (Train & Test)	192

Table 8: Parameters for pre-training of the decoder.