Ref. Ares(2023)8889955 - 28/12/2023

VOXReality

VOICE DRIVEN INTERACTION IN XR SPACES

Advanced AI multi-model for XR analysis

WP3

22-12-2023

Funded by the European Union

Version	1.1
WP	WP3
Dissemination level	Public
Deliverable lead	UM
Authors	Yusuf Can Semerci, Pawel Maka, Abderrahmane Issam, Gerasimos Spanakis (UM), Georgios Papadopoulos, Athanasios Ntovas, Stefanos Biliousis, Petros Drakoulis, Konstantinos Konstantoudakis, Dimitris Zarpalas (CERTH), Apostolos Maniatis, Stavroula Bourou (SYN), Jiahuan Pei, Irene Viola, Pablo Cesar (NWO-I)
Reviewers	Olga Chatzifoti (MAG); Carina Pamminger (HOLO)
Abstract	This document describes the work done in the first 15 months of the project regarding the natural language processing models.
Keywords	Natural Language Processing, Automatic Speech Recognition, Neural Machine Translation, Visual Language Models, Conversation Agents
License	CC BY ND This work is licensed under a Creative Commons Attribution-No Derivatives 4.0 International License (CC BY-ND 4.0). See: https://creativecommons.org/licenses/by-nd/4.0/

Dissemination Level: PU	
PU	**Public**
PP	Restricted to other programme participants (Including the Commission Services)
RE	Restricted to a group specified by the consortium (Including the Commission Services)
CO	Confidential, only for members of the consortium (Including the Commission Services)

Nature : RE	
PR	Prototype
RE	**Report**
SP	Specification
TO	Tool
OT	Other

Version History

Version	Date	Owner	Editor(s)	Changes to previous version
0.1	2023-11-10	UM	Yusuf Can Semerci	Outline
0.2	2023-12-07	UM	Yusuf Can Semerci	Related works
0.21	2023-12-09	SYN	Apostolos Maniatis	Related works
0.3	2023-12-12	UM	Pawel Maka	Added Sec. 2.2
0.4	2023-12-12	CERTH	Petros Drakoulis	Added Sec. 2.3
0.41	2023-12-13	SYN	Stavroula Bourou	Updated Related works
0.5	2023-12-14	SYN	Apostolos Maniatis	Added Sec. 2.4.1
0.6	2023-12-15	UM	Abderrahmane Issam	Added Sec. 2.1, Updated Sec. 2.2
0.61	2023-12-15	UM	Yusuf Can Semerci	Updated Sec. 2.1 and Sec 2.2
0.62	2023-12-19	CERTH	Petros Drakoulis	Updated Sec. 2.3
0.63	2023-12-19	SYN	Stavroula Bourou	Updated Sec. 2.4.1
0.7	2023-12-20	NWO-I	Jiahuan Pei	Added Sec. 2.4.2
1.0	2023-12-20	UM	Yusuf Can Semerci	Final formatting and controls
1.01	2023-12-22	MAG	Olga Chatzifoti	Internal review complete
1.02	2023-12-22	HOLO	Carina Pamminger	Internal review complete
1.1	2023-12-22	UM	Yusuf Can Semerci	Revisions complete

Table of Contents

List of Abbreviations & Acronyms 5

List of Figures 7

List of Tables 8

Executive Summary 9

1. Introduction 10

2. Related Works 11

  2.1. Automatic Speech Recognition (ASR) 11

  2.2. Neural Machine Translation (NMT) 13

  2.3. Visual Language Models (VL) 15

  2.4. Conversation Agents (CA) 17

  2.5. Evaluation Metrics 20

3. VOXReality Natural Language Processing Models 23

  3.1. Automatic Speech Recognition (ASR) 23

  3.2. Neural Machine Translation (NMT) 27

    3.2.1. Context-aware Machine Translation 27

    3.2.2. Robust Machine Translation 33

    3.2.3. Simultaneous Machine Translation 35

    3.2.4. Deployment 38

    3.2.5. Next steps 39

  3.3. Vision-Language Models (VL) 40

    3.3.1. Early models delivery 40

    3.3.2. Metrics and limitations 42

    3.3.3. The Vision (RGB)-Language case 44

    3.3.4. Next Steps 51

  3.4. Conversation Agents (CA) 51

    3.4.1. VR Conference Conversation Agent 51

    3.4.2. AR Training Assistant 67

4. Conclusions 78

References 79

Appendix I: Endpoints for the ASR and NMT components 86

Appendix II: Endpoints for the VL components 89

Appendix III: Endpoints for the navigation assistant 91

Appendix IV: Endpoints for the training assistant 92

List of Abbreviations & Acronyms

AI	:	Artificial Intelligence
API	:	Application Programming Interface
AR	:	Augmented Reality
ARTA	:	Augmented Reality Training Assistant
ASR	:	Automatic Speech Recognition
BART	:	Bidirectional and Auto-Regressive Transformers
BEA	:	Building Educational Applications
BERT	:	Bidirectional Encoder Representations from Transformers
BLEU	:	Bilingual Evaluation Understudy
C4	:	Colossal Clean Crawled Corpus
CA	:	Conversation Agent
CNN	:	Convolutional Neural Network
COCO	:	Common Objects in Context
CTC	:	Connectionist Temporal Classification
DDPG	:	Deep Deterministic Policy Gradients
DM	:	Dialogue Management
DP	:	Dialogue Policy
DST	:	Dialogue State Tracking
FCE	:	First Certificate in English corpus
FFN	:	Feed-forward Network
GA	:	Grant Agreement
GAtt	:	Ghost Attention
GMM	:	Gaussian Mixture Model
GPT	:	Generative Pre-trained Transformer
GPU	:	Graphics Processing Unit
GUG	:	Grammatical/Ungrammatical
HMM	:	Hidden Markov Model
IC	:	Image Captioning
JFLEG	:	JHU FLuency-Extended GUG corpus
JHU	:	John Hopkins University
LASER3	:	Language-Agnostic SEntence Representations
LR	:	Learning Rate
LCPT	:	Large Contrastive Pronoun Test
LLM	:	Large Language Models
LSTM	:	Long Short-Term Memory
LXMERT	:	Language-visual Model for Efficient Representations of Transformers
LoRA	:	Low-Rank Adaptation
M2M-100	:	Many-to-Many Multilingual
METEOR	:	Metric for Evaluation of Translation with Explicit Ordering
MLM	:	Masked Language Modeling
MMS	:	Massively Multilingual Speech
MRM	:	Masked Region Modeling
MT	:	Machine Translation

MoE	:	Mixture of Experts
NAS	:	Neural Architecture Search
NLG	:	Natural Language Generation
NLLB	:	No Language Left Behind
NLP	:	Natural Language Processing
NLU	:	Natural Language Understanding
NMT	:	Neural Machine Translation
OC	:	Open Calls
PEFT	:	Parameter-efficient fine-tuning
PPO	:	Proximal Policy Optimization
QLoRA	:	Quantized Low Raw Adaptor
ReAct	:	Reason and Act
RGB	:	Red-Green-Blue
RL	:	Reinforcement Learning
RLHF	:	Reinforcement Learning with Human Feedback
RNN	:	Recurrent Neural Network
ROUGE	:	Recall-Oriented Understudy for Gisting Evaluation
RelU	:	Rectified Linear Unit
SD	:	Scene Description
SKS	:	Semantic Knowledge Similarity
SMT	:	Statistical Machine Translation
SOTA	:	State-of-the-art
T5	:	Text-To-Text Transfer Transformer
TEACH	:	Task-driven Embodied Agents that Chat
TEACH-EDH	:	TEACH - Execution from Dialogue History
UNITER	:	UNiversal Image-TExt Representation
VL	:	Vision-Language
VQA	:	Visual Question Answering
VR	:	Virtual Reality
WER	:	Word Error Rate
WP	:	Work Package
XR	:	eXtended Reality

List of Figures

Figure 2.1. The architectures of a) Deep Speech and b) Deep Speech 2 12
Figure 2.2. The architectures of a) wav2vec, b) XLSR, and c) HuBERT 12
Figure 2.3. OpenAI's Whisper architecture 13
Figure 2.4. Seq2Seq architecture 13
Figure 2.5. The architectures of a) BART, b) T5, and c) M2M-100 14
Figure 2.6. Show and Tell architecture 16
Figure 2.7. The architectures of a) LXMERT, b) VisualBERT, and c) UNITER 17
Figure 2.8. The architecture of GPT model [Radford et al., 2018] 18
Figure 2.9. Training process of Llama 2-Chat 18
Figure 3.1. Whisper Model with Adapter Modules 23
Figure 3.2. The speech transcription endpoint 25
Figure 3.3. The speech translation endpoint 25
Figure 3.4. The context-aware speech translation endpoint 26
Figure 3.5. The architecture of latent grouping and latent selecting 28
Figure 3.6. The architecture of VOXReality context-aware MT model 28
Figure 3.7. The terminology-constrained architecture 29
Figure 3.8. BLEU scores on the OpenSubtitles2018 test set in all consortium languages 33
Figure 3.9. Training a Robust Model Using Adapters 34
Figure 3.10. The architecture of wait-k adapter model 36
Figure 3.11. BLEU-Average lagging results for 37
Figure 3.12. The text translation endpoint 38
Figure 3.13. The context-aware translation endpoint 38
Figure 3.14 The context-aware terminological translation endpoint 39
Figure 3.15 The terminology configuration endpoint 39
Figure 3.16. Early endpoints for a) image captioning, b) visual question answering 41
Figure 3.17. The example image with a flower in a vase and a kitten next to it 43
Figure 3.18. Example RGB and Depth image that includes a person and a fire hydrant 46
Figure 3.19. The architecture with ViT encoder and GPT2 decoder for the IC task 48
Figure 3.20. Progression across training epochs of a) standard evaluation metrics, ... 49
Figure 3.21. Data distribution across categories 53
Figure 3.22. Conference agent workflow 55
Figure 3.23. Illustration of LoRA during and after training 61
Figure 3.24. Training and validation loss per epoch 62
Figure 3.25. The intend and destination recognition endpoint 66
Figure 3.26. The response generation endpoint 66
Figure 3.27. Autonomous agent powered by large language models 67
Figure 3.28. The workflow of ARTA 68
Figure 3.29. The illustration of conversational ReAct walkthrough 69
Figure 3.30. Interactive information flow between XR test application and AI agent 69
Figure 3.31. Comparison of full fine-tuning, LoRA, and QLoRA 70
Figure 3.32. Examples of TEACH-EDH dataset 71
Figure 3.33. Examples of VOX-ARTA-LEGO dataset 72
Figure 3.34. The learning curve of Llama-2-7b-chat-hf model finetuning on 74
Figure 3.35. Interactive WebSocket API 77

List of Tables

Table 3.1. Whisper Size Variants.. 24
Table 3.2. Hyperparameters of Whisper Finetuning 24
Table 3.3. WER Results of Finetuning vs. Adapter finetuning of Whisper........................... 24
Table 3.4. MMS and Whisper Comparison Using Average WER on 54 Languages 26
Table 3.5. The hyper-parameters of all models trained on IWSLT 2017 dataset 31
Table 3.6. Results of the models trained on the IWSLT 2017 with EN-DE dataset.............. 31
Table 3.7. Results of the models trained on the IWSLT 2017 with EN-FR dataset 32
Table 3.8. The hyper-parameters of the terminology-constrained model 32
Table 3.9. Accuracy of the models on the ContraPro contrastive dataset........................... 33
Table 3.10. The hyper-parameters of the robust MT model 35
Table 3.11. A Comparison of Finetuning and Adapter based finetuning on JFLEG-es 35
Table 3.12: Hyperparameters of the Simultaneous Machine Translation Models 36
Table 3.13: Evaluation scores on the test-set and training loss across training epochs 49
Table 3.14. Pearson correlation coefficients between all evaluation metrics. 50
Table 3.15. Experimental results of the NLU model 54
Table 3.16. Evaluation of sentence-transformers fine-tuning process with sentence similarity
.... 63
Table 3.17. Evaluation of navigation agent before and after fine-tuning 64
Table 3.18. Evaluation of intent recognition agent before and after fine-tuning 65
Table 3.19. Evaluation of destination recognition agent before and after fine-tuning........... 65
Table 3.20. Descriptions of the functions for the XR tools in the test application 70
Table 3.21. Statistics of TEACH-EDH and VOX-ARTA-LEGO datasets 72
Table 3.22. Evaluation results on TEACH-EDH and VOX-ARTA-LEGO dataset 74

Executive Summary

This document corresponds to the deliverable D3.1 - Advanced AI multi-model for XR analysis, of work package 3 (WP3) and describes the work done in the first 15 months of the project regarding the natural language processing (NLP) models.

In VOXReality, the natural language processing models are developed for the following tasks:
1) Automatic speech recognition (ASR),
2) Machine translation (MT),
3) Image captioning (IC) and scene description (SD),
4) Visual question answering (VQA),
5) Conversation agents (CA) for navigation and training assistance

The document is divided into four chapters:

Chapter 1 provides a brief introduction to the WP3 tasks and to this deliverable.

Chapter 2 describes the background knowledge required to understand the VOXReality NLP models and the related works from the literature for each of the five tasks listed above.

Chapter 3 presents the models developed for each of the five tasks listed above. Section 3.1 describes the work performed regarding automatic speech recognition and the initial results obtained from the evaluation of the performance of the models trained, Section 3.2 describes the context-aware machine translation, simultaneous machine translation and robust machine translation models and their performances with benchmark datasets, Section 3.3 describes the models implemented for spatial image captioning, spatial visual question answering, and spatial scene description, and Section 3.4 presents the work performed on developing the navigation assistant conversation agent and the training assistant conversation agent.

Chapter 4 summarizes the conclusions obtained with this deliverable and with the work done during the first 15 months of the project and provides a prospect of further work regarding the NLP model implementations and modifications.

1.Introduction

The eXtended Reality (XR) technologies, specifically Virtual Reality (VR) and Augmented Reality (AR), offer individuals immersive environments where they can engage with and manipulate both physical and virtual environments in multidimensional ways, presenting considerable opportunities for various sectors like education, training, collaboration, entertainment and health. The technologies that facilitate this immersive experience offer ways to engage and interact with these virtual or augmented environments through hand-held point and click interfaces (e.g., controllers) or data driven AI-based hand tracking. However, these technologies are limited for natural interactions, especially for human-to-human communication since human-to-human interaction is mainly driven by language (verbal communication) and enhanced with gesture-based physical cues. Furthermore, the rapid spread of the COVID-19 pandemic accelerated the progress of digital meeting platforms and digital assistants. As a result, human communication and interaction underwent a notable shift toward the digital medium. The ability to organize events, engage in meetings with colleagues, and conduct training activities became feasible through digital platforms in the midst of the pandemic allowing individuals all around the world to be connected. Moreover, recent advancements in Artificial Intelligence (AI) technologies have further elevated the interaction with machines, enabling users to seamlessly employ natural language in their day-to-day activities. These observations underscore the need for a coordinated strategy in implementing context-aware, multilingual, visually grounded tools to enhance natural communication among humans and between humans and machines on digital platforms such as XR environments.

One of the main objectives of VOXReality is to implement natural language processing models which are pre-trained, publicly available, optimized, multilingual, visually grounded and knowledgeable of the domain-specific needs of applications in XR environments. To this end, the VOXReality consortium invests in research and development activities in WP3 to provide NLP models for automatic speech recognition, machine translation, visual question answering, visual captioning, spatial scene description, and conversation agents for navigation and training assistance in 4 tasks.

“Task 3.1 - VOXReality Audio processing” focuses on the algorithms, models and tools to process audio streams or files for the automatic speech recognition, “Task 3.2 - VOXReality Multilingual translation” is responsible for the implementation, training and fine-tuning of the machine translation models, “Task 3.3 - Vision and language based XR models” implements visual language (VL) models that can answer questions about visual cues, describe an image by summarizing the content of the image (caption), and describe a scene with spatial consideration, and “Task 3.4 - Context aware generative dialogue system” is responsible for the implementation of two generative conversation agents that are adapted to two different tasks: indoor navigation and machine assembly training.

The technical work described in this document is the work performed in all four tasks (T3.1, T3.2, T3.3, and T3.4) of WP3 until the end of the 15th month of the VOXReality project.

2. Related Works

Natural Language Processing (NLP) is the study and development of algorithms and computational models that enable machines to understand, interpret, and generate human language. Early NLP research focused on the development of rule-based systems and symbolic approaches that relied on predefined grammatical rules and linguistic structures to analyse and generate text. However, these approaches faced limitations in handling the complexity and variability of natural language. The shift towards statistical methods, probabilistic models and machine learning techniques enabled the NLP systems to learn from data and adapt to linguistic patterns facilitating more robust language processing capabilities. Recent advancements in machine learning, deep learning, the availability of large datasets and more efficient and powerful hardware setups revolutionized the NLP.

Recent years have seen the rise of transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2018] and GPT (Generative Pre-trained Transformer) [Radford et al., 2018], which have demonstrated remarkable performance in a wide range of NLP tasks. The following sections present the evolution of approaches for the NLP tasks that VOXReality is investing in, namely, automatic speech recognition, machine translation, visual language models, and conversation agents.

2.1. Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) focuses on the development of systems capable of transcribing spoken language into text. The approaches to ASR evolved from early rule-based approaches to the current state-of-the-art methods driven by deep learning and transformers. Early systems relied heavily on handcrafted rules and linguistic models that match acoustic features with phonetic units [Jelinek, 1976]. Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) were utilized to develop early systems that were capable of handling limited vocabulary tasks for specific domains. HMMs were used to handle the temporal dynamics of speech, while GMMs were used to model the probability distribution of acoustic features [Rabiner, 1989]. The first revolutionary shift happened with the advancements in machine learning methodologies. Deep Speech [Hannun et al, 2014] and Deep Speech 2 [Amodie et al, 2016] are two of the most prominent examples of the ASR models introduced in the era of rise of machine learning algorithms in NLP.

Deep Speech employs an end-to-end neural network architecture (Figure 2.1a) with connectionist temporal classification (CTC) loss to train an NLP model, where acoustic features, such as spectrograms, are mapped to a character sequence without the need for phonetic or linguistic modelling and without the need for aligned input-output pairs. The CTC loss, increased number of parameters (model size) and increased amount of data utilized to train the model contributed to the success of the model in achieving the state-of-the-art at the time of publication. Deep Speech 2 builds upon the previous iteration by introducing the Recurrent Neural Network (RNN) approach (Figure 2.1b), specifically Long Short-Term Memory (LSTM), to handle long-term dependencies in the sequential data (speech), and the batch normalization, which normalizes the inputs to a layer within a mini batch, addressing issues like vanishing or exploding gradients. Furthermore, Deep Speech 2 introduces the multilingual capabilities of deep neural network architectures and surpasses its predecessor in achieving the state-of-the-art.

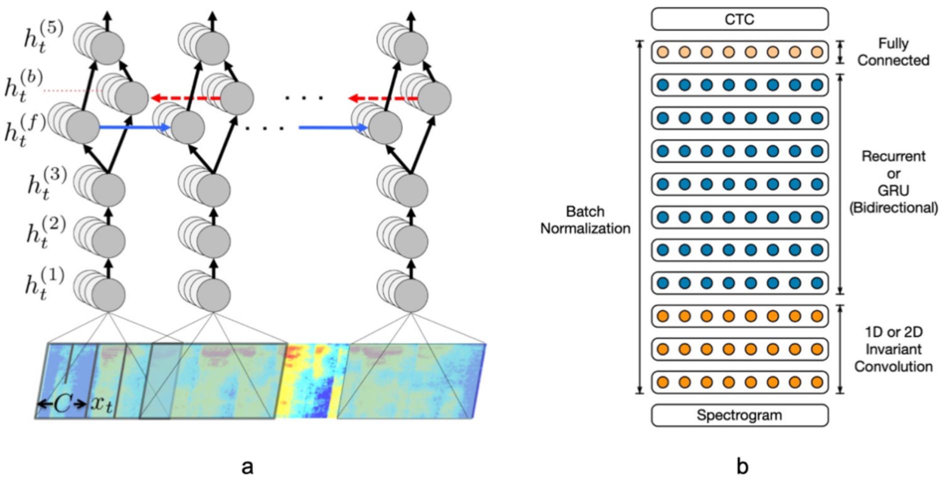


Figure 2.1. The architectures of a) Deep Speech and b) Deep Speech 2

The introduction of attention mechanisms and transformer models [Vaswani et al., 2017] accelerated the NLP research tremendously, especially with the success of models such as BERT. Recent ASR approaches employ speech representation models such as wav2vec [Baevski et al., 2020], XLSR [Conneau et al., 2020] and HuBERT [Hsu et al., 2021], which are three of the most prominent speech models benefited from the methodologies introduced with attention mechanisms, transformers and BERT. All three models are trained with a self-supervised learning approach where a model is trained on a task generated from its own input data that allow the model to learn meaningful representations without relying on externally labelled datasets. wav2vec introduces a framework (Figure 2.2a) that leverages contrastive learning, a self-supervised learning approach where a model is trained to maximize the similarity between augmented views of the same instance and minimize the similarity between views of different instances, while XLSR introduces the functionality of multilingual quantization (Figure 2.2b) to the contrastive learning approach, where the aim is to capture the shared phonetic representations across languages. HuBERT, on the other hand, employs the masked language modeling (MLM) approach (Figure 2.2c), where random tokens in a given language sequence are masked and the model is trained to predict the masked tokens based on the surrounding unmasked tokens, to capture contextual dependencies in the speech signal.

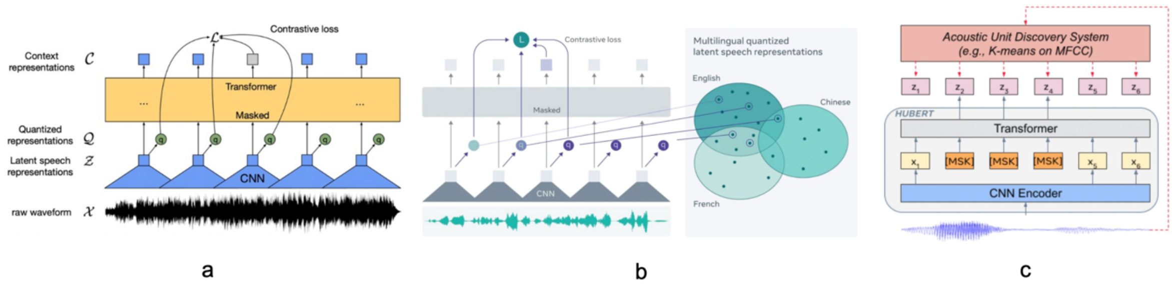


Figure 2.2. The architectures of a) wav2vec, b) XLSR, and c) HuBERT

One of the most recent advancements in ASR is Whisper [Radford et al., 2023], which is an end-to-end model based on transformers and utilizes log-Mel spectrogram of audio inputs. Whisper is trained to predict textual outputs with a multi-task approach where these tasks include language identification, multilingual speech transcription, and to-English speech translation (Figure 2.3). In the pre-training phase, the model is trained on a large dataset using the MLM approach. In the fine-tuning phase, the model is further trained on a smaller labelled dataset using supervised learning to improve its performance on specific tasks.

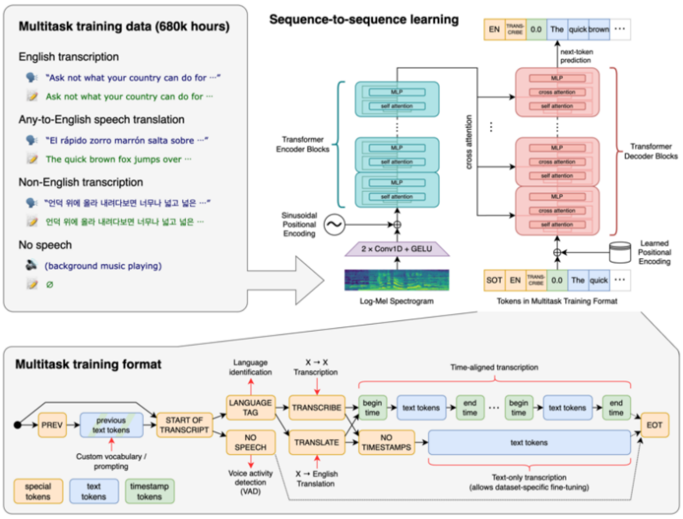


Figure 2.3. OpenAI's Whisper architecture

2.2. Neural Machine Translation (NMT)

Machine Translation (MT) refers to the automatic text or speech translation from one language to another using computational methods. Similar to the ASR evolution, MT transitioned from rule-based methodologies to transformer-based, deep neural network methodologies over the years. Early methods utilized linguistic rules crafted by linguists and dictionaries to translate text [Bar-Hillel, 1951]. The first automated machine translation tools were developed with the introduction of Statistical Machine Translation (SMT) approaches such as Moses [Koehn et al., 2007]. SMT models, especially Moses, utilize statistical alignment models to align words or phrases in source and target corpora and decoding strategies such as beam search or cube pruning.

The first breakthrough from the early statistical methods occurred with the introduction of deep learning techniques and the emergence of the Neural Machine Translation (NMT) field, which is the approach to machine translation that utilizes neural network architectures, particularly sequence-to-sequence models. In the first study that mentions NMT [Sutskever et al., 2014], the model proposed, called Seq2Seq, consists of an RNN encoder that processes the input sequence into a fixed-size vector, and an RNN decoder that generates the output sequence based on this vector. This architecture (Figure 2.4) paved the way for models that assign a sequence of data to a fixed size vector, which revolutionized the MT field.

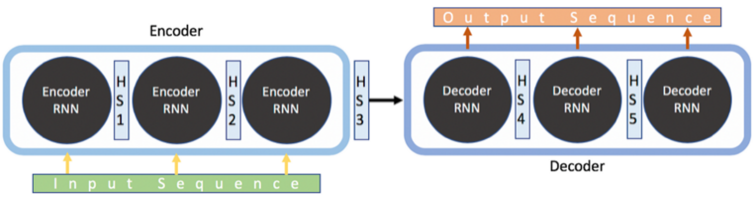
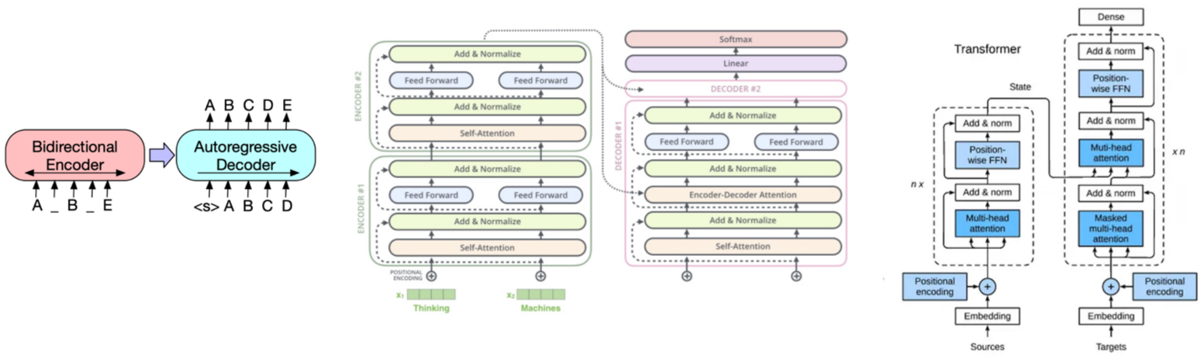


Figure 2.4. Seq2Seq architecture[1]

[1] https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

The latest breakthrough occurred with the introduction of attention mechanisms, transformers and BERT and GPT architectures. Recent models such as BART [Lewis et al., 2019], T5 [Raffel et al., 2020], M2M-100 [Fan et al., 2021], and NLLB [Costa-jussà et al., 2022] all benefited from this breakthrough and they were considered to be the state-of-the-art at the time of their publication. BART (Bidirectional and Auto-Regressive Transformers) is an encoder-decoder model (Figure 2.5a), where the encoder is trained in a bidirectional manner using MLM objective with a denoising approach, meaning the input sequence is randomly masked, the encoder is trained to reconstruct the original sequence from the corrupted one and the decoder is trained using an autoregressive approach, meaning the decoder generates one token at a time and the next token is generated based on the previously generated token. T5 (Text-To-Text Transfer Transformer) is an encoder-decoder structure (Figure 2.5b), where the model is pretrained on a large corpus using unsupervised learning with a denoising autoencoder objective and introduces task-specific prefixes during training and fine-tuning that enable the model to handle a plethora of down-stream tasks such as translation. M2M-100 (Many-to-Many Multilingual) is a many-to-many multilingual transformer-based model (Figure 2.5c) trained on a large multilingual corpus, which includes parallel data for translation tasks in a task-agnostic manner leveraging self-attention mechanisms to capture contextual relationships in the input sequence. NLLB (No Language Left Behind) is a multilingual transformer-based model, which introduces the Sparsely Gated Mixture of Experts approach to the network structure that is constructed based on another model called LASER3 (Language-Agnostic SEntence Representations) [Heffernan et al., 2022]. Sparsely Gated Mixture of Experts models activate a subset of model parameters as opposed to all parameters in traditional dense models.



a b c

Figure 2.5. The architectures of a) BART, b) T5, and c) M2M-100

Various NMT studies focus on specific tasks for machine translation to tackle the challenges of the multilingual general NMT models. VOXReality focuses on three of these major challenges in NMT research: a) Context-aware Machine Translation, which refers to the integration of contextual information, such as surrounding context or terminology, into the process of machine translation to enhance the accuracy of translated text, especially to minimise the errors coming from ambiguities in the source text, b) Robust Machine Translation, which refers to the machine translation systems that can produce accurate and reliable translations across a variety of challenging conditions, including linguistic variations and noisy inputs, and c) Simultaneous Machine Translation, which refers to the machine translation systems that can translate in real-time as the source text is being generated, without waiting for the entire input to be available.

In Context-aware Machine Translation, the state-of-the-art architectures can be broadly categorized into single-encoder and multi-encoder approaches. In single-encoder architectures, the context sentences are concatenated with the current sentence and processed as a long sequence by a single encoder [Sun et al., 2020]. In multi-encoder architectures, the context sentences are processed by a separate encoder than the current sentence and may involve sharing parameters of encoders to reduce the number of parameters needed to be trained [Huo et al., 2020]. In multi-encoder approaches the features obtained from the encoders can be fused in multiple ways, such as by concatenation before passing the information to the decoder or by using cross-attention mechanisms.

In Robust Machine Translation, the methodologies can be categorized into two main approaches: 1) data augmentation and domain adaptation and 2) adversarial training. In data augmentation approaches, the model is exposed to a diverse set of linguistic patterns through techniques such as back-translation, multi-domain training, and fine-tuning on in-domain and/or out-of-domain data [Xie et al., 2020]. In adversarial approaches, the model is trained to produce accurate results even in the presence of adversarial and noisy inputs using techniques such as regularization that penalize sensitivity to input variations or employing adversarial training to reduce the effect of domain shift [Miyato et al., 2016].

In Simultaneous Machine Translation, the methodologies employed can be categorized into fixed-wait, multiple-wait and adaptive-wait approaches. In fixed wait architectures, the models wait for *k* amount of tokens to be received to start the translation process [Ma et al., 2018]. In multiple-wait architectures, the model is trained to support a set of *k* values and can be utilized with one of these *k* values by using methods such as mixture-of-experts to wait for tokens to start translating [Zhang & Feng, 2021]. In the adaptive-wait architectures, the model is trained to decide when to translate and when to wait for more tokens by using methods such as monotonic multihead attention. The models trained with this approach also have the capability to adapt to different wait (*k*) values [Ma et al., 2019].

2.3. Visual Language Models (VL)

Visual language models refer to the neural network models that aim to jointly represent and understand cross-modal information in images and text. Early examples of visual language understanding approaches followed traditional computer vision methods and hand-crafted features and utilized separate traditional models for visual content and the natural language.

The introduction of Convolutional Neural Networks (CNNs), RNNs and large-scale datasets such as Common Objects in Context (COCO) [Lin et al., 2014] paved the way for data-driven approaches such as Show and Tell [Vinyals et al., 2015], which utilizes CNNs for image feature extraction and LSTMs for natural language generation (Figure 2.6).

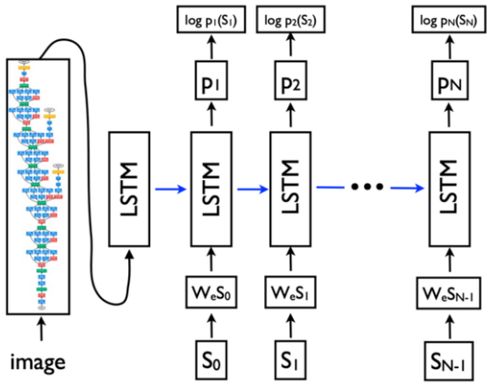


Figure 2.6. Show and Tell architecture

The latest breakthrough occurred with the introduction of attention mechanisms and transformer architectures. The most prominent models that utilize these methods are called LXMERT [Tan & Bansal, 2019], VisualBERT [Li et al., 2019] and UNITER [Chane et al., 2020]. These models are capable of a plethora of visual language tasks, such as image captioning and visual question answering, which are the tasks VOXReality is also investing in.

LXMERT (Language-visual Model for Efficient Representations of Transformers) is a visual language model that utilizes a transformer-based architecture (Figure 2.7a), a vision-answer bi-attention mechanism that focuses on relevant image regions and words in the textual context, and a positional encoding for the vision modality to capture the spatial relationships, and learns joint representations of both modalities during training for better understanding of the relationship between visual and textual context.

VisualBERT is a visual language model that extends the BERT architecture (Figure 2.7b) and it is pre-trained with an MLM objective and a sentence-image prediction objective using the COCO dataset. It also introduces a set of visual embeddings: a visual feature representation of the bounding region, a segment embedding to indicate that the representation is an image embedding. and a position embedding to align words and bounding regions.

UNITER (UNiversal Image-TExt Representation) is a visual language model that employs a transformer-based architecture (Figure 2.7c) and is pre-trained using four objectives: MLM, Masked Region Modeling (MRM), where regions in an image are masked and the model's objective is to predict the masked region, Image-Text Matching, where the objective of the model is to predict if the given text and the image region is a match or not, and Word-Region Alignment, where the aim is to align words and image regions using a predetermined loss function instead of letting the model determine the alignment implicitly. In UNITER, the MLM objective is conditioned on the observation of the full text and the MRM objective is conditioned on the full image instead of the commonly used approach of applying joint random masking to both modalities.

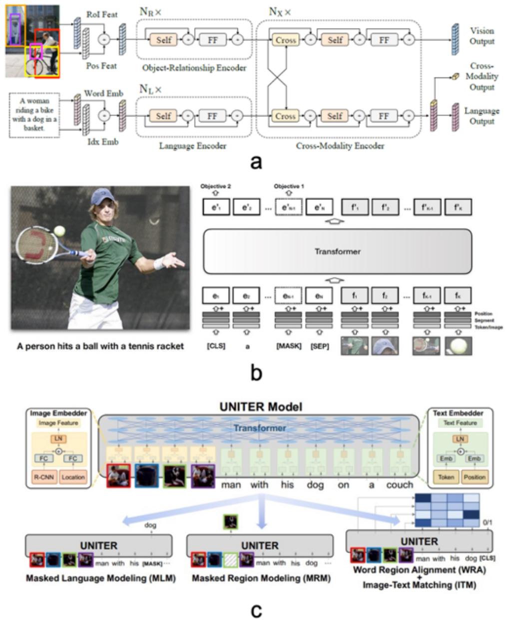


Figure 2.7. The architectures of a) LXMERT, b) VisualBERT, and c) UNITER

2.4. Conversation Agents (CA)

Conversation agents (CAs), commonly known as chatbots, are systems designed to simulate human-like conversations with users, often through text or voice-based interfaces. The history of CA began with the creation of ELIZA [Weizenbaum, 1966], a program that mimicked a Rogerian psychotherapist by using pattern matching and substitution methodology. This was followed by PARRY [Kochen, 1980], another milestone that simulated a person with paranoid schizophrenia.

The advancement of CAs received a considerable boost with the advent of deep learning technologies. Notably, the introduction of Seq2Seq [Sutskever et al., 2014] marked a major breakthrough in this evolution, significantly enhancing the capabilities and complexity of conversational agents. This model was primarily designed for machine translation but was quickly recognized for its potential in creating conversational agents. The significant advancements in CA technology took a big step forward with the introduction of the attention mechanism and transformer-based models [Vaswani et al., 2017]. Transformer models are exceptionally efficient in handling sequential data and are particularly adept at capturing the intricacies of human language, making them ideal for creating highly sophisticated conversational agents.

Meanwhile, OpenAI's GPT (Generative Pre-trained Transformer) [Radford et al., 2018] series, including GPT-3 [Brown et al., 2020], utilized an architecture (Figure 2.8) that is characterized by its use of self-attention mechanisms, enabling the models to weigh the importance of each part of the input data in relation to the rest, a crucial factor in understanding and generating coherent, contextually relevant text. The combination of a transformer backbone with extensive pre-training and fine-tuning underpins the remarkable capabilities of GPT models in generating human-like, context-aware text.

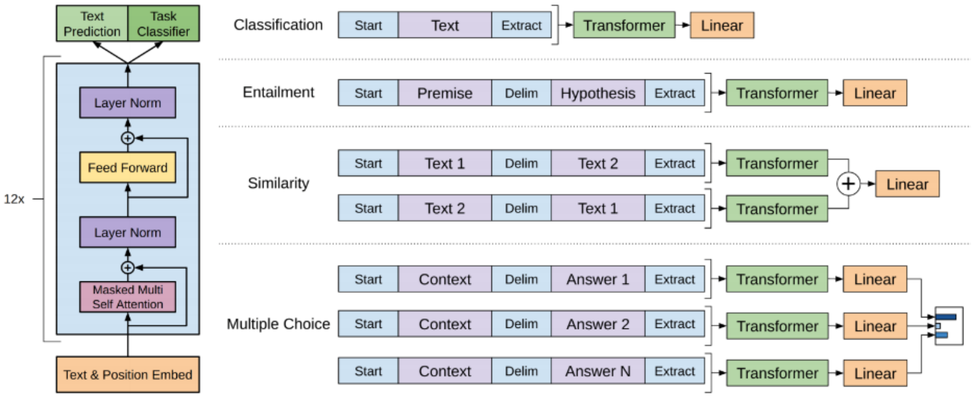


Figure 2.8. The architecture of GPT model [Radford et al., 2018]

Moreover, the Llama-2 [Touvron et al., 2023] model represents a significant advancement in the field with the utilization of the Ghost Attention (GAtt) method, which addresses the loss of context in multi-turn conversations. This method artificially concatenates the instruction to all user messages, leading to context-rich dialogues and improved attention. Furthermore, Llama-2 forms the backbone of instruction-tuned CA Llama2-Chat (Figure 2.9), which applies pretraining on the Llama 2 model using publicly available online sources and a further supervised fine-tuning. The model was then further refined through iterative processes using Reinforcement Learning with Human Feedback (RLHF) methodologies, including rejection sampling and Proximal Policy Optimization (PPO) [Schulman et al., 2017]. Crucially, throughout the RLHF stage, there was a parallel accumulation of iterative, reward modeling data, ensuring the reward models remained within distribution and aligned with the model enhancements. The Llama2-Chat models are available in three versions: 7 billion, 13 billion, and 70 billion parameters, each tailored to different levels of complexity and conversational requirements.

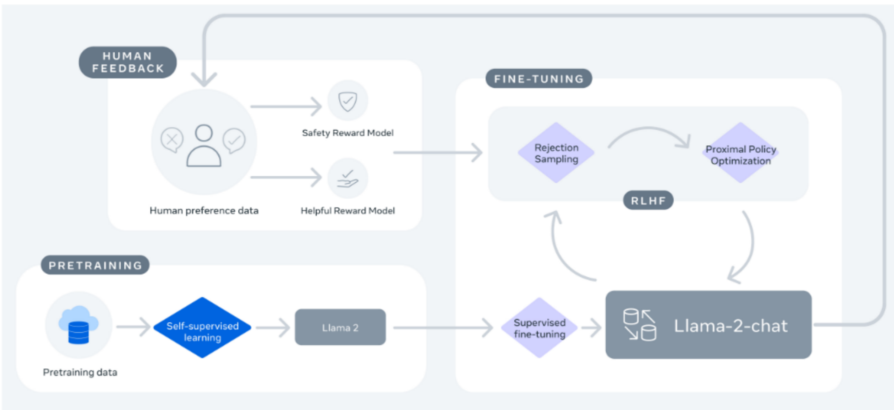


Figure 2.9. Training process of Llama 2-Chat

The conversation agents can be classified into two categories: task-oriented conversational agents, which focus on specific functions, and general conversational agents, designed for more dynamic and wide-ranging interactions. VOXReality focuses on the development of task-oriented conversational agents, specifically tailored to the requirements of the virtual conference and machine assembly training use cases. The agent in virtual conference use case is being developed with a focus on navigation assistance, providing detailed information about the conference program and facilitating interactions within the trade show, and the agent in machine assembly training use case focuses on the conversational training assistant. The task oriented CAs include three main components: Natural Language Understanding (NLU), Dialogue Management (DM), and Natural Language Generation (NLG).

1. Natural Language Understanding (NLU): NLU is the component responsible for extracting the meaning from user inputs in natural language. It serves as the bridge between the user's query and the agent’s ability to comprehend and respond effectively.

2. Dialogue Management (DM): Following NLU's extraction of user intent and context, the extracted information is passed on to the DM, which is the component responsible for orchestrating the conversation and deciding how the agent should respond based on the user's input and the bot's task or purpose. It acts as the decision-maker, determining the appropriate course of action within the conversation flow. Dialogue Management is often split into two sub-components: Dialogue State Tracking and Dialogue Policy.

a) Dialogue State Tracking (DST): DST focuses on keeping track of the current state of the conversation by maintaining a record of all relevant information and context, including user queries, preferences, and any previous interactions. DST employs techniques such as slot filling to extract key pieces of information from user messages.
b) Dialogue Policy (DP): DP is responsible for deciding the agent’s actions and responses based on the dialogue state using strategies and algorithms for selecting the most suitable response or action at each turn of the conversation. DP uses the information provided by DST to determine the next steps in the dialogue. DP ensures that the agent’s responses are coherent, relevant, and aligned with the user's goals.

3. Natural Language Generation (NLG): NLG is responsible for converting structured data and decisions generated by the DM into human-readable and natural-sounding responses by generating text responses that are coherent, contextually relevant, and grammatically correct.

In the context of task-oriented CAs, it's essential to highlight that the utilization of state-of-the-art transformer-based models like T5, BERT and similar ones can be advantageous for all the components. T5 has reimagined NLP tasks by framing them uniformly as text-to-text problems, facilitating a more integrated and coherent approach to a variety of tasks, including translation, summarization, and question answering. T5 has undergone comprehensive pre-training on the “Colossal Clean Crawled Corpus” (C4)[2], which is instrumental in its ability to generate and understand human-like text. The model is available in several scalable variants, each tailored for different computational needs and performance requirements:

- **T5-Small**: Optimized for limited computational resources, ideal for mobile or edge computing.
- **T5-Base**: A balanced choice for general-purpose NLP tasks, offering a good compromise between efficiency and performance.
- **T5-Large**: Geared towards more complex NLP tasks, it delivers higher accuracy due to its larger size.

[2] https://github.com/google-research/text-to-text-transfer-transformer#c4

- **T5-3B and T5-11B**: These are the largest configurations, containing billions of parameters, and are typically used for high-end NLP research and applications with substantial computational resources.

However, DP often takes a different approach; instead of utilizing pre-trained models, DP commonly relies on Reinforcement Learning (RL) methods. PPO is a popular RL algorithm used for DP, which optimizes the policy of the agent by iteratively collecting data and improving the policy to maximize rewards. Another algorithm, Deep Deterministic Policy Gradients (DDPG) [Lillicrap et al., 2017], is used when the action space is continuous since the aim of the algorithm is to learn deterministic policies that map states to specific actions, making it suitable for precise control in task-oriented dialogues.

In recent advancements, the end-to-end approach for task-oriented dialogue agents, such as GPT-3 and Llama-2, has gained significant traction, offering a more streamlined and integrated methodology. This approach treats the entire dialogue process as a single holistic task, which includes understanding user intent, managing the dialogue, and generating responses. Unlike traditional models that handle these components separately, the end-to-end approach process and respond to user inputs directly. GODEL [Peng et al., 2022] is another end-to-end model that addresses the challenges of task-oriented dialogues.

2.5. Evaluation Metrics

There are three evaluation metrics widely adopted across the NLP field: BLEU [Papineni et al., 2002], ROUGE [Lin et al., 2004] and METEOR [Banerjee et al., 2005].

BLEU (Bilingual Evaluation Understudy) is the most widely used metric for evaluating the output of machine translation systems with human-generated reference translations. It operates by comparing n-grams (contiguous sequences of n items, typically words) between the candidate (machine-generated) and reference (human-generated) translations. The precision of n-grams in the candidate translation is measured by how many of them overlap with n-grams in the reference translations. While this concept is sound for long translations, it has a tendency to over-rate short translations. In an effort to overcome this, Modified Precision is introduced, which considers the maximum number of times a particular n-gram appears in any single reference translation. This is done to prevent the metric from being overly optimistic about short and repetitive translations. Another measure to that direction is that it includes a brevity penalty if the length of the candidate translation is less than the length of the reference translations.

BLEU computes precision scores for different n-gram lengths (usually up to 4-grams), and then combines them using a weighted geometric mean. The weights are usually equal, but some variations may assign different weights. The final score is calculated by multiplying the combined precision by the brevity penalty. The brevity penalty is typically the exponential of 1 minus the ratio of the length of the reference translation to the length of the candidate translation. Its values range from 0 to 1, with 1 indicating a perfect match between the candidate and reference translations.

A notable limitation of BLEU is that it does not explicitly consider word order or capture semantic nuances and a high BLEU score doesn't necessarily mean the translation is of high quality in all aspects. It doesn't capture semantic meaning or fluency and might favour translations that are close to the references but not necessarily correct or natural sounding.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric commonly used for evaluating the quality of automatic summarization and machine translation outputs. It assesses the similarity between a generated summary and one or more reference summaries by measuring the overlap of n-grams (contiguous sequences of n items, usually words) and other units of text. There are several variants of ROUGE metrics, such as ROUGE-N, ROUGE-L, ROUGE-L_{sum}, ROUGE-W, ROUGE-S, ROUGE-SU and ROUGE-M. Each variant focuses on different aspects of similarity:

- ROUGE-N (Unigram, Bigram, etc.): Measures overlap of n-grams between the system summary and the reference summary. ROUGE-1 considers unigrams (single words), ROUGE-2 considers bigrams (two consecutive words), and so on.
- ROUGE-L (Longest Common Subsequence): Measures the longest common subsequence of words between the system and reference summaries. It is less sensitive to word order than n-gram-based metrics.
- ROUGE-L_{sum}: Similar to ROUGE-L but considers multiple occurrences of the same n-gram in the reference.
- ROUGE-W (Weighted Overlap): Considers the overlap of weighted words. It assigns higher weights to important words, giving more importance to content words.
- ROUGE-S (Skip-Bigram): Measures the overlap of pairs of words that have one or more words in between them. It helps capture some word order information.
- ROUGE-SU (Skip-Bigram with Unigram): An extension of ROUGE-S that considers unigrams as well. It combines unigrams and skip-bigrams to capture more information.
- ROUGE-M (ROUGE for Multigrams): Computes precision, recall, and F1 score for various lengths of n-grams.

ROUGE scores range from 0 to 1, where a higher score indicates better similarity between the generated and reference summaries. The basic calculations in all variants involve counting the number of overlapping units (n-grams, words, or sub-sequences) between the generated summary and the reference summary, based on the chosen metric (n-grams, longest common subsequence, etc.). The precision, recall, and F1 score are then computed based on these counts:

- Precision: Proportion of overlapping units in the generated summary with respect to the total number of units in the generated summary.
- Recall: Proportion of overlapping units in the generated summary with respect to the total number of units in the reference summary.
- F1 Score: Harmonic mean of precision and recall.

ROUGE, like all metrics described here, has innate limitations that need to be taken into consideration. Firstly, it is case-sensitive, so "cat" and "Cat" would be treated as different units. In order to alleviate this, case folding may need to be applied. Also, punctuation and stop-words (common words like "the," "and," etc.) are often excluded or given less importance. Additionally, ROUGE metrics may be sensitive to summary length. Extremely short or long summaries can affect the evaluation. It's important to ensure that the evaluation considers the length of the summaries appropriately. Finally, it should be noted that it is upon us to choose the correct variant of ROUGE, better suited to our case. For example, ROUGE-N is sensitive to word order, while ROUGE-L is more lenient in this regard. Also, ROUGE can be applied at different levels, including word level, sentence level, or document level. The level chosen depends on the granularity of evaluation desired.

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is another metric used for evaluating the quality of machine translation outputs and was designed to address some of the limitations of BLEU and ROUGE. It incorporates explicit word matching, stemming, synonymy, and word order into its evaluation process. The evaluation process starts by aligning words in the candidate (system-generated) translation with words in the reference translation. It considers exact matches, stemming (reducing words to their root form), and synonymy (using WordNet) to find matches. Then, it computes precision, recall, and F1 score based on the number of matched unigrams, bigrams, and synonym pairs. These measures are calculated separately and then combined using a harmonic mean to get an overall score. METEOR incorporates a penalization mechanism for overgeneration (when the system generates more words than necessary) and undergeneration (when the system generates fewer words than necessary). These penalties are incorporated into the score to encourage more accurate translations. Another feature of it is that it employs a more flexible alignment approach compared to some other metrics. It allows for multiple reference words to be matched to a single candidate word and vice versa. This helps handle variations in expression.

METEOR provides scores for various components, such as unigram precision, unigram recall, precision for stemmed unigrams, recall for stemmed unigrams, synonym precision, synonym recall and penalty terms for overgeneration and undergeneration. It allows for different components to be weighted differently, based on their importance. This enables users to customize the metric according to their priorities. The final score is a combination of the individual component scores, with each component weighted according to user preferences. The flexibility of METEOR in handling stemming, synonymy, and various language-specific nuances makes it suitable for evaluating translations in a wide range of languages. It has shown good correlation with human judgment in various machine translation evaluations, making it a quite reliable metric for automatic assessment. Of course, like any automatic evaluation metric, it might not fully capture the intricacies of human judgment. There are cases where a translation may be technically correct but might not align perfectly with reference translations.

While METEOR is a comprehensive metric for the evaluation of machine translation outputs, it also has some limitations and considerations to be aware of. It can be sensitive to preprocessing choices, such as tokenization and stemming. Small variations in preprocessing can lead to differences in the evaluation results. Also, it relies on WordNet for synonym matching. While WordNet is a valuable resource, its coverage and accuracy may vary across different languages and domains. In some cases, it may not capture all relevant synonyms. Most notably though, METEOR places a strong emphasis on exact matching. While it considers stemming and synonyms, deviations from the reference translations still result in lower scores even if the meaning is preserved, mainly rooted to the fact that it takes word order into account. Lastly, in practice tuning METEOR for optimal performance can be challenging due to its multiple parameters and components. The need for careful parameter tuning makes it less straightforward to use.

3. VOXReality Natural Language Processing Models

3.1. Automatic Speech Recognition (ASR)

Automatic Speech Recognition is crucial to the VOXReality Pipeline. We found that ASR models underperform on Greek language, and we worked on improving Whisper on Greek through finetuning and using Adapter modules.

Whisper [Radford et al., 2023] is a Multilingual Multi-task model that can perform ASR, translation into English and Language Identification. We integrate Whisper into our pipeline because of its robustness to speech variations that comes with training on large amounts of weakly supervised data. Although Whisper performs really well on high resource languages, it underperforms on low or medium resource languages like Greek, but the representations learned during the Multi-task pretraining can still be useful for finetuning a better model for Greek.

We use adapters [Rebuffi et al., 2017] [Houlsby et al., 2019] for efficient finetuning of the Whisper model. Whisper architecture is a normal transformer-based encoder-decoder architecture, therefore, we plugin adapter modules after the feed forward network of both the encoder and the decoder. We insert one adapter module per layer. The adapter architecture [Bapna et al., 2019] we use is composed of layer normalization, a linear layer that lowers the dimension of the input (down projection), and a linear layer that brings the input back to its original dimension (up projection). The two linear layers are separated by a Rectified Linear Unit (RELU). The architecture of the model that utilizes adapter modules is illustrated in Figure 3.1.

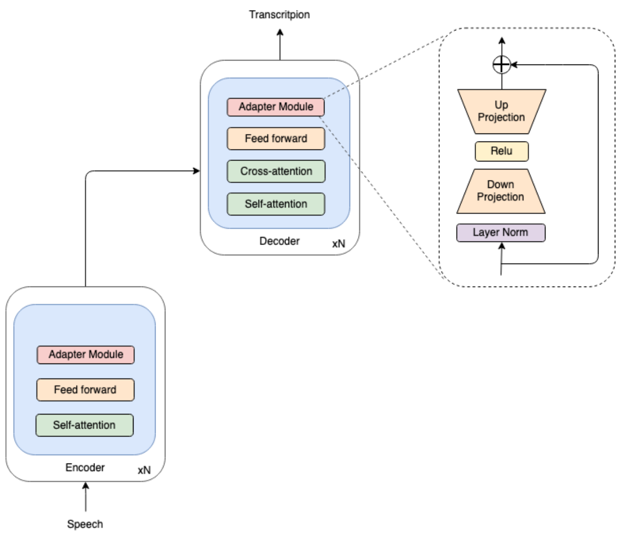


Figure 3.1. Whisper Model with Adapter Modules

Results

Whisper comes in multiple variants that can suit different quality and size requirements. As shown in Table 3.1, these models differ in terms of their number of parameters, and thus their performance is also different. The number of parameters can have implications on the latency and the amount of memory required to run the model for inference. We choose Whisper-small as a good trade-off between quality and size of the model.

Table 3.1. Whisper Size Variants

Model	Layers	Width	Heads	Parameters
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

For training and evaluation, we use the Common Voice dataset [Ardila et al., 2019], which is a multilingual corpus for Speech Recognition and Language Identification. The dataset is collected through crowdsourcing, where contributors can record their own voice reading a sentence in the Common Voice website or app, hence, the dataset contains speech from different speakers and with different levels of noise, and it can be used to evaluate or train robust speech recognition models. We use the Greek Language partition of the dataset, which contains 1.91k rows for training, 1.7k for validation, and 1.7k for test. We used both the training and validation subsets for training to get better results and used the test set for evaluation. For the other languages we only used the test set for reporting the results. We report in Table 3.2 the hyperparameters of the finetuning.

Table 3.2. Hyperparameters of Whisper Finetuning

Hyper-parameter	Value
Encoder Layers	12
Decoder Layers	12
Attention Heads	12
Embed Dim	768
FFN Embed Dim	1536
Dropout	0.
Optimizer	Adam
Learning Rate	1e-5
LR Scheduler	Linear
Batch Size	8

We report the Word Error Rate (WER) metric results in Table 3.3. The original model (Whisper-small) achieves decent performance on all languages except Greek, where the WER is between 17.28 and 24.74 points higher than the other reported languages, a fact which motivated our work on finetuning Whisper for Greek. After finetuning, the performance improves by ~14 points, but it worsened on the other languages due to information forgetting. We found that finetuning is still better than using adapters by ~5 points, but the adapters guarantee a decent performance on the other languages because the model is frozen and it allows us to omit the use of the adapters during inference.

Table 3.3. WER Results of Finetuning vs. Adapter finetuning of Whisper

Model	Greek	Dutch	German	Italian	Spanish	English
Whisper-small	41.93	17.71	17.00	24.65	16.19	19.36
Whisper-small-finetuned	28.11	22.67	21.07	36.04	22.05	25.35
Whisper-small-adapters	33.48	19.87	18.41	27.10	17.05	20.36

Deployment

In the VOXReality pipeline the audio processing is handled through a set of endpoints in a REST API developed using FastAPI[3] framework. The speech related endpoints are designed to generate transcriptions and translations from audio files in “wav” format and utilizes the Whisper-small-adapters model for ASR. There are three endpoints: speech recognition (transcription), speech translation and context-aware speech translation. The details of the API calls are presented in Appendix I.

In speech recognition, the function generates a textual response to a given audio file in the “wav” format. The endpoint does not require a source language to be specified since it is automatically recognized by the model itself. Figure 3.2 presents the FastAPI-powered endpoint that can be requested using the handle “transcribe_audio_files” from the deployed server.

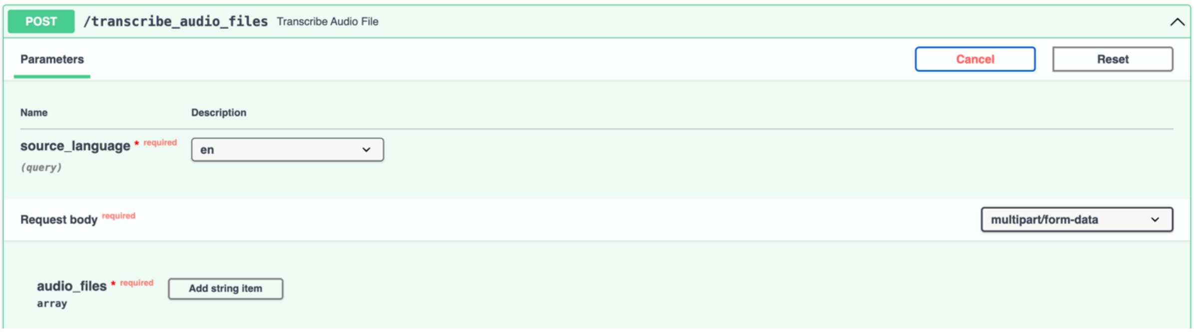


Figure 3.2. The speech transcription endpoint

In speech translation, the function generates a textual response to a given audio file in the “wav” format. The endpoint does not require a source language to be specified since it is automatically recognized by the ASR model itself. This endpoint utilizes the speech recognition model to transcribe the audio file and automatically utilizes the latest text translation model to translate the transcribed text to the given target language. Figure 3.3 presents the FastAPI powered endpoint that can be requested using the handle “translate_audio_files” from the deployed server.

Figure 3.3. The speech translation endpoint

[3] https://fastapi.tiangolo.com/

In context-aware speech translation, the function generates a textual response to a given audio file in the "wav" format. The endpoint does not require a source language to be specified since it is automatically recognized by the ASR model itself. This endpoint utilizes the speech recognition model to transcribe the audio file and automatically utilizes the latest context-aware text translation model to translate the transcribed text to the given target language using the given contextual text. Figure 3.4 presents the FastAPI-powered endpoint that can be requested using the handle "contextual_translate_audio_files" from the deployed server.

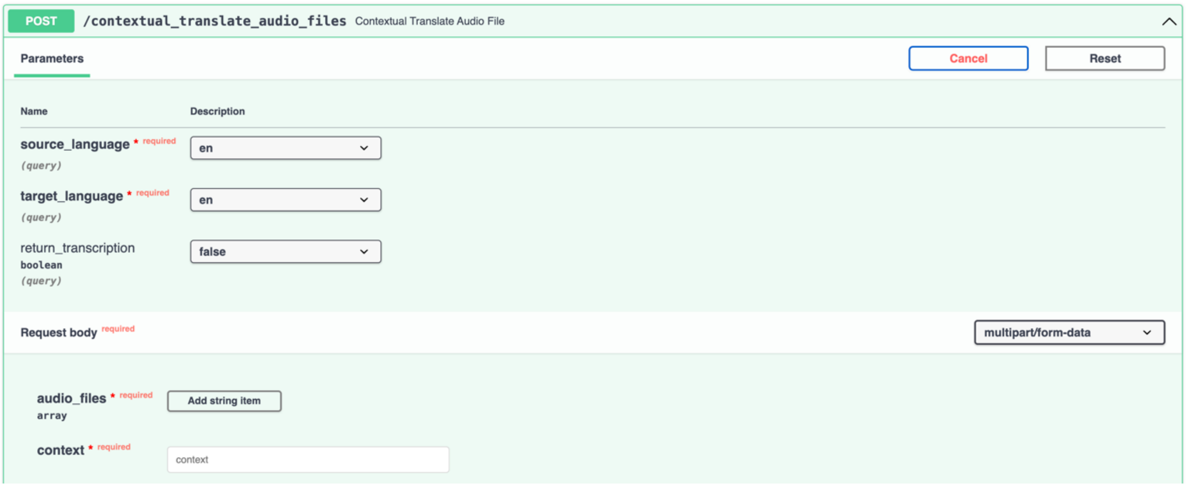


Figure 3.4. The context-aware speech translation endpoint

Next steps

The Massively Multilingual Speech (MMS) Project is a recent effort from Meta to scale Speech models to 1000+ languages [Pratap et al., 2023]. The effort has resulted in pretrained speech representation models, ASR, Language Identification and Speech Synthesis models. The pretrained model is a Wav2Vec 2.0 based model [Baevski et al., 2020], which they finetune on downstream tasks. The ASR finetuning was done on a multilingual dataset of 1107 languages. This has led to MMS achieving better results than Whisper on languages that both models are trained on, as it is shown in Table 3.4. We will invest in the MMS models and apply our approach to the pre-trained MMS models to produce better ASR performance for the consortium languages. Furthermore, we will work on real-time ASR models to facilitate simultaneous speech translation and simultaneous speech transcription.

Table 3.4. MMS and Whisper Comparison Using Average WER on 54 Languages

	#lang	labeled train data (h)	FLEURS-54 dev	FLEURS-54 test
Prior Work				
Whisper medium	99	680K	-	50.1
Whisper large-v2	99	680K	-	44.3
This Work				
MMS	61	3K	20.9	20.7
MMS (LSAH)	61	3K	19.0	19.1
MMS	1,107	45K	24.8	24.8
MMS (LSAH)	1,107	45K	18.7	18.7

3.2. Neural Machine Translation (NMT)

In VOXReality, we introduce three major categories of machine translation models: a) Context-aware Machine Translation, which includes surrounding sentences and/or terminology as context, b) Robust Machine Translation, which utilizes mechanisms to handle translation of noisy inputs, and c) Simultaneous Machine Translation, which translates source text in real-time as the input is being generated.

3.2.1. Context-aware Machine Translation

In VOXReality, we utilize multi-encoder architectures, where all the encoder parameters are shared. This allows caching the hidden representation of the current sentence and reusing it as the hidden representation of the context when translating subsequent sentences, and we refer to this architecture as caching. Furthermore, several techniques have been proposed to shorten the sequence of tokens where the tokens are combined in shortening modules that are added between a specified number of encoder layers. This can lead to the reduction of the computational and memory requirements in the subsequent layers; therefore, a compressed representation of the previously seen sentences should be enough to use as a context.

Consequently, we focused on the application of Sequence Shortening to Context-aware Machine Translation and we introduce two new shortening techniques (Latent Grouping and Latent Selecting), where the network can learn how to group or select tokens to form a shortened sequence. Shortening can be done by average/max pooling of the hidden representation of the tokens [Subramanian et al., 2020] or linear pooling of the concatenated representation of the tokens of the original sequence [Nawrot et al., 2021]. In the pooling-based shortening, the sequence is divided into non-overlapping groups of K neighbouring tokens.

In Latent Grouping, each token is categorized into a group by the feed-forward network (FFN) with the number of outputs equals to the number of groups (K). We obtain the categorization for the ith token to K^{th} group by applying the SoftMax function to the outputs in the dimension of the groups. Subsequently, the groups are constructed as the sum of the hidden representations. The network learns how to softassign each token to the groups. A group representation is computed using the weighted average of tokens, which makes backpropagations into the categorizing network possible. Finally, the attention module is applied on the group representations. Latent Selecting enables the groups to select tokens to aggregate rather than to assign each token to a group and allows the model to ignore tokens entirely rather than to assign them to at least one group. Although Latent Selecting can be achieved by maintaining a categorizing feed-forward network for each group, we utilize the same network as described for Latent Grouping but apply SoftMax in the sequence dimension instead of the token dimension. Figure 3.5 illustrates the model utilized for both latent grouping and latent selecting where the number of groups (K) is set to three.

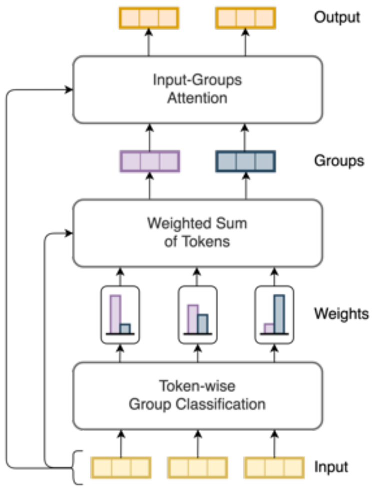


Figure 3.5. The architecture of latent grouping and latent selecting

The architecture we use, illustrated in Figure 3.6, is based on caching the hidden representations produced by the encoder, where the representations of the tokens of the current sentence are stored and can be reused as context when the subsequent sentences are translated. Although this architecture uses only a single encoder, it is different from the single-encoder models because the current sentence and the context sentences are processed separately. While in the standard caching architecture the hidden representation of all the tokens is stored, we introduce a sequence shortening module directly after the encoder, which returns the compressed hidden representation usually containing fewer tokens than the original sequence. The context is integrated in the decoder in the multi-encoder manner by using a separate cross-attention module for context tokens. To allow the decoder to distinguish between context sentences we employ learned segment embeddings [Devlin et al., 2018] and learned positional encoding for the shortened tokens inside context sentences.

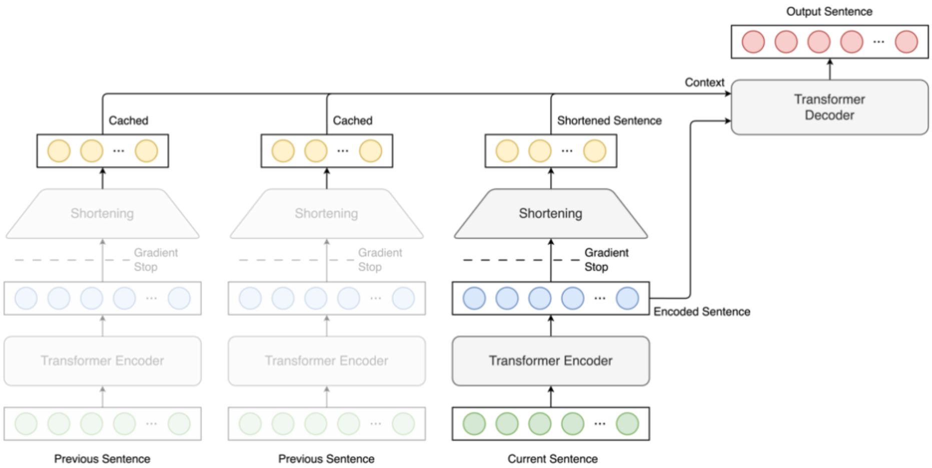


Figure 3.6. The architecture of VOXReality context-aware MT model

During training, caching is not used, meaning that the model receives tokenized context sentences and processes them using the same encoder, where the weights of the encoder receive the backpropagation gradient from multiple sources - the current sentence and each of the context sentences. In order to eliminate difficulties in training that would arise from this, we block the gradient after the encoder and before shortening where applicable. This is done by allowing the gradient information to flow for a specified number of context sentences, after which the gradient is blocked.

Moreover, to allow a machine translation model to use terminology, we adapted the work of [Jon et al., 2021]. The main differences of our work are threefold:

1) we apply this technique to the multi-lingual setting,
2) we train the model capable of context-aware translation by utilizing the previous sentence of the source language,
3) we finetune the pre-trained model (NLLB-600M [Costa-jussà et al., 2022]) instead of training from a random initialization.

The model requires the list of terminology constraints with each item in the form of phrases in all consortium languages (English, German, Italian, Dutch, Spanish and Greek), although the phrase is not required to be specified in all languages. Each pair formed from the two phrases in different languages is treated as a constraint in both directions. We pre-process the terminology list by lemmatizing the phrases using spaCy [Honnibal et al., 2020].

During inference, the source sentence is lemmatized, and the search is conducted for any of the phrases in the terminology list from the same language. If a phrase is found, the corresponding phrase in the target language is appended to the source sentence delimited with the separator token “</s>”. In case when no phrase is found, the token is still added at the end of the source sentence (apart from the usual end-of-sequence token). Additionally, we allow the model to accept the previous sentence from the source language as context. To this end, we prepend the context to the input of the network delimited with the separator token. NLLB-600M model uses the same tokens as a separator and end-of-sequence token and we do not change that in our finetuned model. The architecture is illustrated in Figure 3.7.

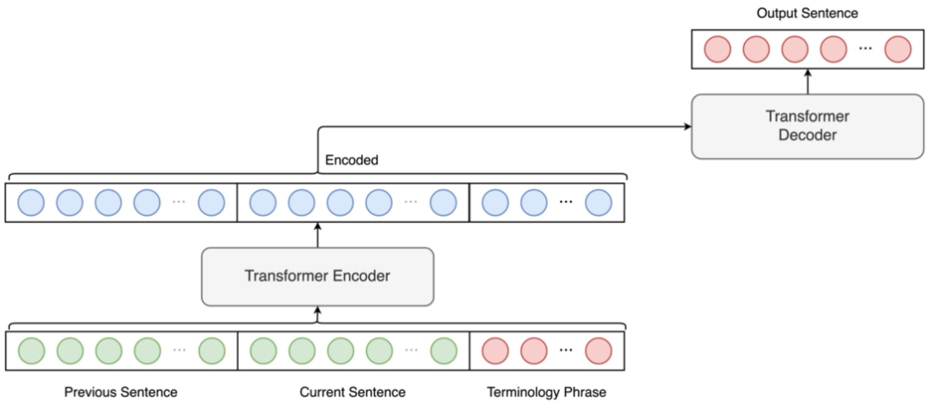


Figure 3.7. The terminology-constrained architecture

Results

It has been argued that the sentence-level metrics do not capture well the ability of the model to use the context. For this reason, to measure how well the model utilizes the contextual information, it is common to use a contrastive dataset such as Contrastive evaluation of Pronoun translation (ContraPro) [Müller et al., 2018] and Large Contrastive Pronoun Test (LCPT) [Lopes et al., 2020]. They are very similar to each other and both target pronoun disambiguation task. They differ in the language pair they use – ContraPro is English-German and LCPT is English-French dataset. The example from the datasets is in the form of context sentences + source sentence in the input and several (usually three) target sentences that differ only in the pronoun. Only one of the target sentences is correct and the others are erroneous. Here is an example from EN-DE dataset:

Context:	*"What's your plan?"*
Source:	*"I forgot to confide it to you."*
Correct Target:	*"Ich vergaß, **es** euch zu vertraun."*
Incorrect Targets:	*"Ich vergaß, **sie** euch zu vertraun."*
	*"Ich vergaß, **ihn** euch zu vertraun."*

The model outputs the probabilities of each target sentence (as a factorized probability calculated by multiplying the probability of each token). When the model scores the correct target as the most probable, model counts it as a correct prediction. Otherwise, it is incorrect.

We trained the following models:
- **Sentence-level Transformer** - where context sentences are ignored
- **Single-encoder Transformer** - where context sentences are prepended to the current sentence and processed by the encoder
- **Multi-encoder Transformer** - with the separate encoder (without weights-sharing) used to encode the context sentences, where the context and the current sentence are concatenated in the decoder (Our experiments revealed that this integration yields better results than with the separate context-attention module)
- **Caching Tokens** - where the encoder representations of the context sentences are stored directly
- **Caching Sentences** - where the representations of the context sentences are averaged and stored
- **Shortening - Mean Pooling** - Sequence shortening with Mean Pooling applied to the outputs of the encoder
- **Shortening - Max Pooling** - shortening with Max Pooling
- **Shortening - Linear Pooling** - shortening with Linear Pooling
- **Shortening - Grouping** - shortening with Latent Grouping (our proposed method)
- **Shortening - Selecting** - shortening with Latent Selecting (our proposed method)

To test the caching and sequence shortening models we trained each model with the same hyper-parameters that are presented in Table 3.5. We used the English to German and English to French directions of the IWSLT 2017 [Cettolo et al., 2017] document-level dataset that is based on the subtitles of TED Talks. We trained each model on both language pairs with varied context sizes (in terms of the number of previous sentences) separately. We measured BLEU [Papineni et al., 2002] using the sacreBleu library, and the accuracy on the contrastive datasets (ContraPro for EN-DE and LCPT for EN-FR).

Table 3.5. The hyper-parameters of all models trained on IWSLT 2017 dataset

Hyper-parameter	Value
Encoder Layers	6
Decoder Layers	6
Attention Heads	8
Embed Dim	512
FFN Embed Dim	2048
Dropout	0.3
Optimizer	Adam
Learning Rate (LR)	5e-4
LR Scheduler	Inverse Sqrt
Batch Max Tokens	4096

The results for EN-DE are presented in Table 3.6. The BLEU score of the context-aware models is generally similar to or slightly higher than the sentence-level Transformer. BLEU does not correlate well with the contrastive accuracy, which is strictly higher for all context-aware models. This confirms that sentence-level metrics do not reflect the context usage of the models. The highest contrastive dataset accuracy was achieved by the Grouping Shortening model for the context size of one, the Max Pooling Shortening model for the context size of two, and the Selecting Shortening model for the context size of three. The highest accuracy averaged over the tested context sizes was reached by the model employing Latent Grouping, followed by the Latent Selecting model. Caching Tokens architecture exhibits comparable BLEU scores to the Single- and Multi-encoder architectures while achieving higher accuracy on the contrastive dataset.

Table 3.6. Results of the models trained on the IWSLT 2017 with EN-DE dataset

Model	BLEU	Accuracy				
Sentence-level	28.11	43.67%				
	Context Size: 1		Context Size: 2		Context Size: 3	
Model	BLEU	Accuracy	BLEU	Accuracy	BLEU	Accuracy
Single-encoder	28.31	47.42%	27.95	48.18%	27.88	48.88%
Multi-encoder	28.67	44.93%	28.50	46.65%	28.26	45.00%
Caching Tokens	28.35	54.06%	28.50	54.13%	29.08	51.23%
Caching Sentence	28.38	45.72%	26.73	45.26%	26.70	44.91%
Shortening – Max Pooling	27.62	51.67%	27.88	55.08%	28.26	50.89%
Shortening – Avg Pooling	28.09	53.37%	27.85	54.81%	28.38	50.54%
Shortening – Linear Pooling	27.62	52.71%	28.03	52.13%	28.18	51.27%
Shortening – Grouping	28.21	56.98%	28.70	54.51%	28.49	51.16%
Shortening – Selecting	28.15	54.48%	28.55	54.21%	28.01	51.95%

The results of the models on the EN-FR datasets can be seen in Table 3.7. The BLEU scores of all models are comparable (apart from the Caching Sentence architecture). Latent Grouping achieved the highest accuracy on the contrastive dataset for the context size of one, and Latent Selecting and Single-encoder architectures for the context sizes of one and three, respectively. In general, Caching Tokens and Shortening models achieved higher accuracies than the Single- and Multi-encoder architectures (with the exception of the Single-encoder on English to French translation with a context size of three).

Table 3.7. Results of the models trained on the IWSLT 2017 with EN-FR dataset

Model	BLEU	Accuracy				
Sentence-level	37.64	75.92%				
	Context Size: 1		Context Size: 2		Context Size: 3	
Model	BLEU	Accuracy	BLEU	Accuracy	BLEU	Accuracy
Single-encoder	37.25	77.27%	37.18	78.98%	37.12	80.87%
Multi-encoder	37.44	75.72%	37.12	77.23%	37.34	75.76%
Caching Tokens	36.88	79.67%	37.29	80.14%	37.73	79.90%
Caching Sentence	36.50	77.33%	34.21	76.25%	34.78	75.71%
Shortening – Max Pooling	37.48	79.51%	36.72	80.59%	37.85	79.71%
Shortening – Avg Pooling	37.13	77.75%	37.12	80.16%	38.18	80.41%
Shortening – Linear Pooling	37.02	80.47%	37.12	79.37%	37.42	79.64%
Shortening – Grouping	37.05	79.91%	37.98	81.13%	37.18	79.54%
Shortening – Selecting	37.38	80.89%	37.83	80.32%	37.81	80.09%

We train the terminology-constrained model on the OpenSubtitles2018 dataset [Lison et al., 2018] for all 30 language pairs formed from the consortium languages in both directions. For each language, we use 50000 examples with the previous sentence as context and 50000 without context. Following [Jon et al., 2021] we augment the dataset with artificial terminology by selecting a random span in the lemmatized target sentence and using it as a target terminology constraint. This augmentation is done to the examples in both sentence-level and context-aware datasets with the probability of 0.3. The word-length of the terminology item is drawn from the geometric distribution with p=0.85 with the maximum of three. The final training dataset is formed by interleaving the examples from the sentence-level and context-aware datasets as well as the language pair-specific datasets. Table 3.7 presents the hyper-parameters.

Table 3.8. The hyper-parameters of the terminology-constrained model

Hyper-parameter	Value
Encoder Layers	12
Decoder Layers	12
Attention Heads	16
Embed Dim	1024
FFN Embed Dim	4096
Dropout	0.1
Optimizer	Adam
Learning Rate	5e-5
LR Scheduler	Linear
Batch Size	12

We evaluated our terminology-constrained model and the baseline (NLLB-600M) on the test subset of the OpenSubtitles2018 dataset for all consortium language pairs. We used our model in four modes of operation:

- No context, no terminology (Figure 3.8 - Ours)
- Context, no terminology (Figure 3.8 – Ours with context)
- No context, terminology (Figure 3.8 – Ours with terminology)
- Context, terminology (Figure 3.8 – Ours with context and terminology)

To obtain the terminology for our model, we used the same procedure as used for training with the same probabilities. The results in terms of BLEU are illustrated in Figure 3.8. Our model outperforms the baseline in all models of operation. Using the terminology leads to the largest improvements in terms of BLEU.

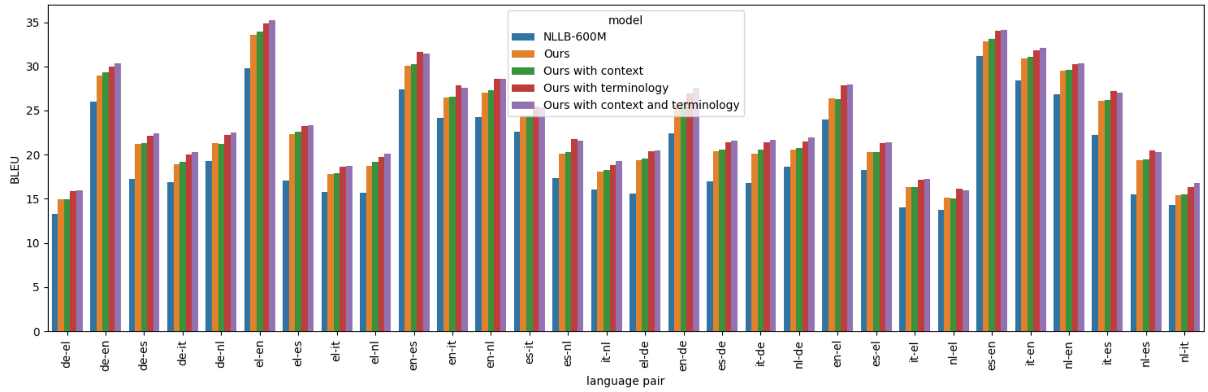


Figure 3.8. BLEU scores on the OpenSubtitles2018 test set in all consortium languages.

Additionally, we tested the usage of the context of our model on the ContraPro (EN-DE) dataset. The results including the base model are presented in Table 3.9. The performance in terms of accuracy on ContraPro dataset of our model is increased by 4% compared to the baseline.

Table 3.9. Accuracy of the models on the ContraPro contrastive dataset

Model	Accuracy
NLLB-600M	46.44%
Ours with context and terminology	50.59%

3.2.2. Robust Machine Translation

Part of our goals at VOXReality is to build robust and reliable models that are ready for production. We research techniques for building a Robust Machine Translation model that is able to produce a correct translation even when the input is noisy. We focus on non-native speaker noise and adapt the model to handle non-native input issues.

We worked on improving a model that underperforms on non-native speakers' noise. We experimented with finetuning and realised that it leads to information forgetting which negatively impacts the performance on clean data. This has led us to the solution of using adapter networks [Rebuffi et al., 2017] [Houlsby et al., 2019]. Adapter networks are small modules that are inserted into the model and can be trained to adapt the model's representations to different tasks or domains. The model can be frozen or trained jointly with the adapters. In our case, we freeze the model to minimize information forgetting which can impact the performance, especially since we are finetuning on noisy data. We use a simple adapter architecture [Bapna et al., 2019] composed of layer normalization, followed by a down projection and up projection with a Rectified Linear Unit (ReLU) in between. The down projection is a linear layer that projects the input into a lower dimension, and the up projection brings the representations back to their original dimension. We insert the adapter network after the feedforward network of the transformer layer. The adapter output is added to the original feedforward network output, which forms a residual connection.

In an encoder-decoder architecture, the encoder takes care of building a representation of the input text that then the decoder can rely on to generate the output, which means that if we can nudge the encoder to output a clean representation even when the input is noisy, the decoder will be able to correctly generate the output. We insert the adapters into the encoder and train them to update the encoder representations while the encoder and decoder parameters are frozen. We train the adapters with two objectives: a translation objective, and a similarity objective where the decoder is motivated to produce representations that have high similarity to clean representations from the same frozen encoder. Figure 3.6 illustrates this architecture. The following equation shows the training objective of our model:

$$L = \lambda.Similarity\left(encoder\left(x_{noisy}\right), encoder'\left(x_{clean}\right)\right) + (1-\lambda).CrossEntropy(y, \hat{y})$$

where λ is a hyperparameter for controlling the impact of the similarity on the model. encoder' refers to the encoder with the adapters. x_{noisy} and x_{clean} are the noisy and clean input respectively. y and $\hat{y}$ are the target and predicted translation respectively.

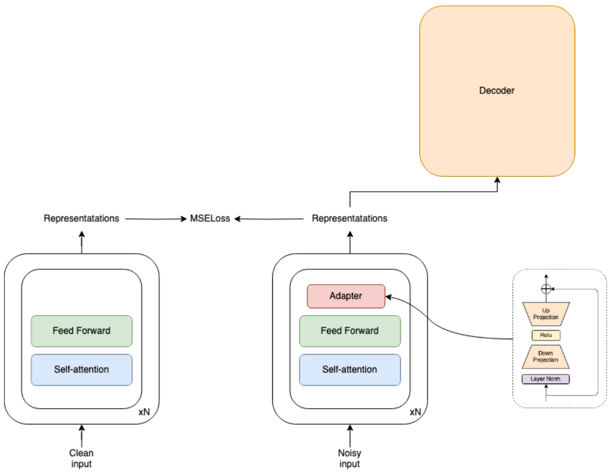


Figure 3.9. Training a Robust Model Using Adapters

Results

In order to properly evaluate our models, we make sure that we evaluate using both clean and noisy input to ensure we don't sacrifice performance on clean data for performance on the noisy one. We use The John Hopkins University (JHU) FLuency-Extended Grammatical/Ungrammatical (GUG) corpus (JFLEG) [Napoles et al., 2017], a dataset of non-native speaker text with corrections. More specifically, we use the JFLEG-es [Anastasopoulos et al., 2019] which is the same dataset paired with Spanish translations. For finetuning the model, we rely on Grammatical Error Correction datasets to acquire clean and noisy input pairs. Then we use the original model to translate the clean inputs to the target language and use the results as reference translations for finetuning. We evaluate the model using three datasets, namely: Lang-8 [Tajiri et al., 2012], First Certificate in English corpus (FCE) [Yannakoudakis et al., 2011], and Building Educational Applications (BEA) [Bryant et al., 2019]. The training dataset size of these datasets are 30543, 1.9 million, and 34309 for FCE, LANG-8, and BEA respectively. We finetune OPUS-mt-en-ROMANCE [Tiedemann & Thottingal, 2020] on each of the three datasets. We report the finetuning hyperparameters in Table 3.10.

Table 3.10. The hyper-parameters of the robust MT model

Hyper-parameter	Value
Encoder Layers	12
Decoder Layers	12
Attention Heads	16
Embed Dim	1024
FFN Embed Dim	4096
Dropout	0.1
Optimizer	Adam
Learning Rate	5e-5
LR Scheduler	Linear
Batch Size	16

We report BLEU on both noisy and clean data in Table 3.11. No finetuning refers to using the original model directly for translation. We can see that there is a gap of more than 2 BLEU points between the performance on clean and noisy data. After finetuning the whole parameters of the model, the performance on both clean and noisy data drops, which can be explained by information forgetting. Finetuning all parameters on noisy data leads the model to forget what it has learned during the initial training. Table 3.11 also reports the results of using adapters with and without the similarity loss and we can see that both techniques improve the results over the original model in both clean and noisy data, while adapters with similarity loss are better on noisy translation when finetuning on Lang-8 and BEA.

Table 3.11. A Comparison of Finetuning and Adapter based finetuning on JFLEG-es

Dataset	No finetuning		Finetuning		Adapters-Similarity Loss		Adapters	
	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
LANG-8	31.92	29.40	26.94	23.67	32.46	31.10	32.32	30.73
FCE	31.92	29.40	29.82	28.54	32.39	31.28	32.55	31.57
BEA	31.92	29.40	30.75	30.06	32.33	31.46	32.41	30.96

3.2.3. Simultaneous Machine Translation

In VOXReality we introduce two approaches for simultaneous machine translation: a) Multi-path training with adapters: a single model that can support multiple fixed wait-k values by using adapters and b) Adaptive strategy: utilizing the probabilities that the model assigns to the most likely token and a predefined probability threshold, we decide on either a “read” (the model continues to read new tokens) or a “write” (the model can continue with translation) action.

In multi-path training with adapters, we insert adapters into the decoder layer, following the implementation of [Bapna et al, 2019] and insert the residual adapter modules after the feed-forward layer. Each adapter is activated for a set of train or test wait-k values. During training, the wait-k values are sampled uniformly following multi-path training [Elbayad et al., 2020]. In the adaptive strategy, we follow the approach of [Zheng et al., 2020] to build an adaptive strategy by using adapters instead of different models for each wait-k value. At each decoding step, we activate one single adapter based on the lagging behind the current generation step. Then, we utilize the probability of the most likely token to decide whether to write or read a new token. If the probability is less than the set threshold, we read a new token, otherwise we write. Furthermore, if wait value (k) is lower than the minimum wait value, we force the model to read and if it is higher than the maximum wait-k, we force the model to write, which means that the choice of minimum and maximum values of wait also impacts latency. The architecture of the model is presented in Figure X.

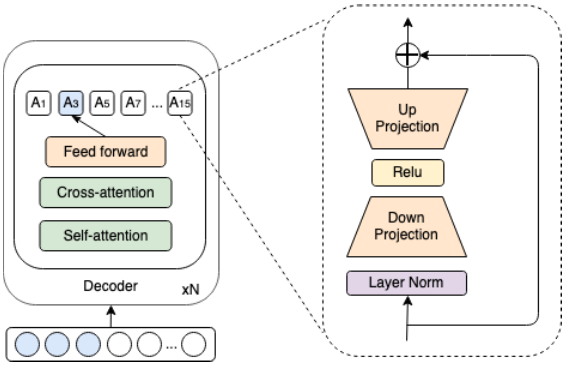


Figure 3.10. The architecture of wait-k adapter model

Results

We utilized our methods in three transformer versions: transformer-small, transformer-base and transformer-big. The hyperparameters for these models are presented in Table 3.12. The evaluation was performed with two public datasets: EN-VI (English - Vietnamese) dataset from IWSLT 2015 Evaluation Campaign [Cettolo et al., 2015] for evaluating transformer-small models, and DE-EN (German - English) dataset for evaluating transformer-base and transformer-big models. The EN-VI dataset consists of 133K pairs and DE-EN dataset consists of 4.5M pairs.

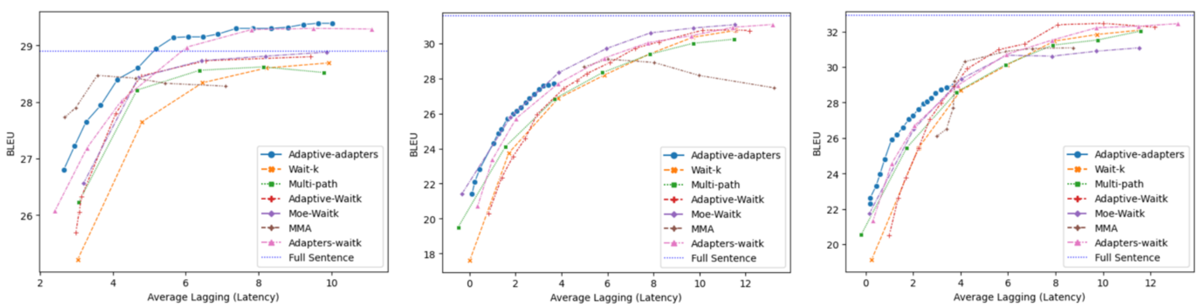
Table 3.12: Hyperparameters of the Simultaneous Machine Translation Models

Hyper-parameter	Transformer-small	Transformer-base	Transformer-big
Encoder Layers	6	6	6
Decoder Layers	6	6	6
Attention Heads	4	8	16
Embed Dim	512	512	1024
FFN Embed Dim	1024	2048	4096
Dropout	0.3	0.3	0.3
Optimizer	Adam	Adam	Adam
Learning Rate	5e-4	5e-4	5e-4
LR Scheduler	Inverse Sqrt	Inverse Sqrt	Inverse Sqrt
Batch Max Tokens	16000	8192	8192
Number of Adapters	8	8	8
Adapter Bottleneck	64	64	64

We trained the following models:

- **Offline Transformer** - where the model takes the full sentence
- **Wait-k** - where the model waits for k source tokens before starting to alternate between writing a target token and reading a source token [Ma et al., 2018]
- **Multipath Wait-k** - where the model supports multiple wait-k policies by randomly sampling k during training and the k value is fixed during inference [Elbayad et al., 2020]
- **Adaptive Wait-k** - where there are multiple models for different wait-k values and during inference the model is selected based on the lagging behind generation step and the decision to write or read is based on the output probabilities. [Zheng et al., 2020]
- **MoE Wait-k** - where the model uses multipath wait-k approach but it utilizes Mixture-of-Experts (MoE) method instead of random [Zhang & Feng, 2021]
- **MMA** - where the model follows the Monotonic Multi-head Attention (MMA) that jointly learns a Bernoulli variable that is used to decide read/write action [Ma et al., 2019]
- **Adapters Wait-k** - multi-path training with adapters (our proposed method)
- **Adaptive Adapters** - the adaptive strategy (our proposed method)

Figure 3.11 presents the results obtained from the comparison with average lagging (k-latency) and BLEU score of our proposed methods with the other models we trained, where Figure 3.11a is EN-VI transformer-small, Figure 3.11b is DE-EN transformer-base, and Figure3.11c is DE-EN transformer-big model results. According to these results, our method improves or competes with other strategies on different latencies while using a single model. MMA, Wait-k, and Adaptive-waitk require the training of multiple models in order to support multiple latencies, while our method is more flexible in this regard. The number and capacity of the adapters can be adjusted depending on the complexity of the language direction and the latencies we are planning to support. Furthermore, using adapters alone (Adapters Wait-k) is competitive with other methods especially on EN-VI, but using an adaptive strategy further improves the results, especially in low latencies. In comparison to Adaptive Wait-k, where wait-k policy models are trained and composed during inference, we find that our method is better in all latencies while being more efficient. Compared to Moe-Waitk, which also aims at minimizing interference introduced by multi-path training, we find that our method is better in all latencies on EN-VI for transformer-small and on DE-EN with transformer-big, while achieving competitive results when using transformer-base.



a b c

Figure 3.11. BLEU-Average lagging results for a) transformer-small, b) transformer-base, and c) transformer-big

3.2.4. Deployment

In the VOXReality pipeline, the translation process is handled through a set of endpoints in a REST API developed using FastAPI framework. The machine translation endpoints are designed to generate translations from textual inputs. There are four endpoints: text translation, context-aware translation, context-aware terminological translation, and terminology configuration. The details of the API calls are presented in Appendix I.

In text translation, the function generates a textual response in the target language to a given text. The endpoint does not require a source language to be specified since it is automatically recognized by the model itself. Figure 3.12 presents the FastAPI powered endpoint that can be requested using the handle “translate_text” from the deployed server.

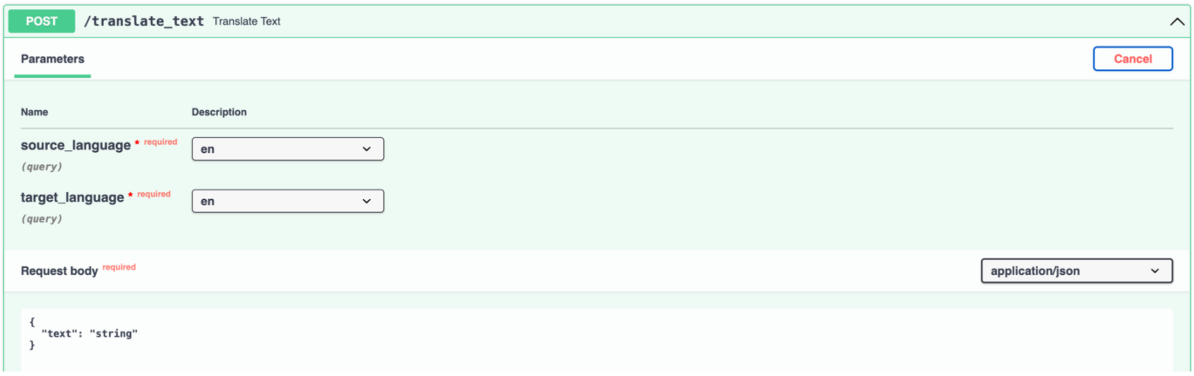


Figure 3.12. The text translation endpoint

In context-aware translation, the function generates a textual response in the target language to a given text and a given context, which is provided as textual information. The endpoint does not require a source language to be specified since it is automatically recognized by the model itself. Figure 3.13 presents the FastAPI powered endpoint that can be requested using the handle “contextual_translate_text” from the deployed server.

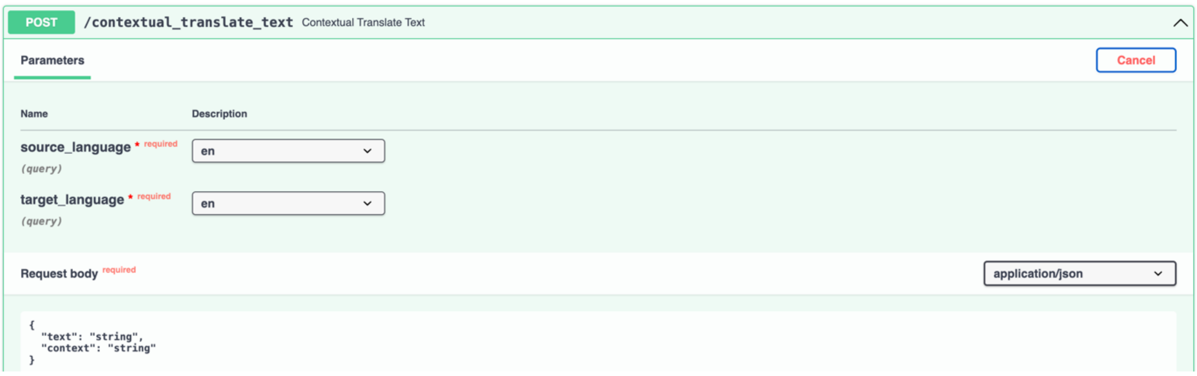


Figure 3.13. The context-aware translation endpoint

In context-aware terminological translation, the function generates a textual response in the target language to a given text and a given context, which is provided as textual information, as in the context-aware translation, but the model in this endpoint also utilizes a file that is uploaded to the server, which contains terminological terms and their translations in other languages. The endpoint does not require a source language to be specified since it is automatically recognized by the model itself. Figure 3.14 presents the FastAPI powered endpoint that can be requested using the handle "contextual_terminology_translate_text" from the deployed server.

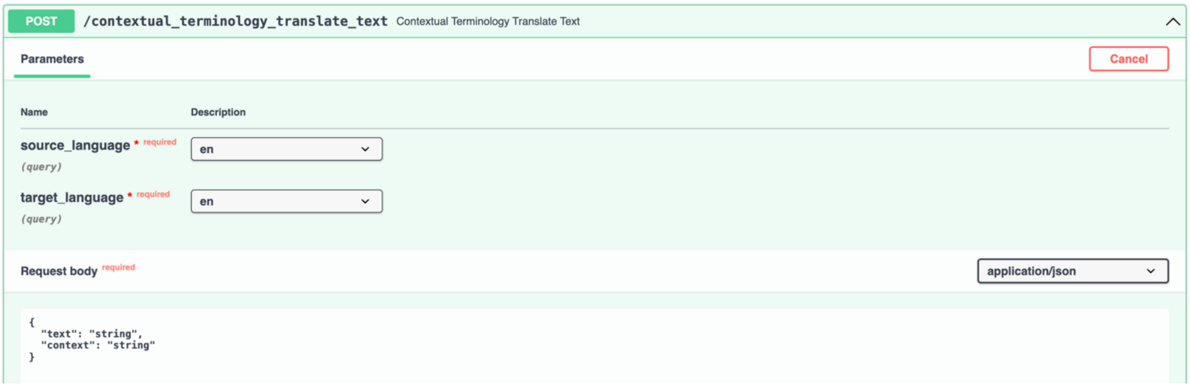


Figure 3.14 The context-aware terminological translation endpoint

In terminology configuration, the function uploads a file that contains terminological terms and their translations in other languages to the server. Figure 3.15 presents the FastAPI powered endpoint that can be requested using the handle "upload_terminology" from the deployed server.

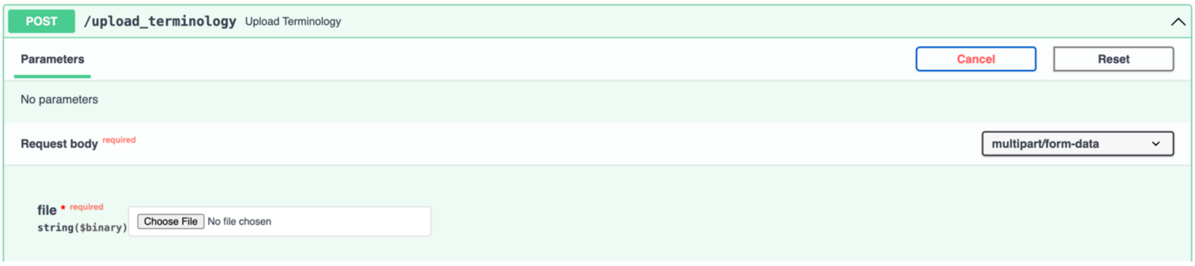


Figure 3.15 The terminology configuration endpoint

3.2.5. Next steps

For the machine translation task, we will implement the following next steps:
1) The integration of the simultaneous machine translation with the API
2) The integration of the robust machine translation with the API
3) The implementation of context-aware terminological robust machine translation
4) The implementation of context-aware terminological robust machine simultaneous machine translation
5) The fine-tuning of the current models with a dataset we are generating using the play Hippolytus by Euripides for better translation of this text from Greek to other consortium languages.
6) Further research and experiments with the proposed models and their variations for achieving better results for the usecases
7) Scientific publication of the proposed methodologies

3.3. Vision-Language Models (VL)

In compliance with the task description in the project's Grant Agreement (GA), we are developing a series of spatially aware Vision-Language (VL) models that act as "visual context" providers to the rest of the VOXReality's pipeline. In this way, various other components like the Neural Machine Translation (NMT), the Automatic Speech Recognition (ASR) and the Conversation Agents (CA) of the use-cases can benefit from additional, current knowledge of the surrounding environment, adapting their behaviour to that.

Core-concept of any VL model is the apparent alignment of Vision and Text modalities, which the network implicitly has to learn in order to perform all the complex VL tasks. All these tasks require foremost the connection between the various entities of the visual media (I.e. image, video) and the words in the accompanying sentence. This is called "vision and language representations alignment". Note that these connections can be so subtle that may refer to sub-word (minimum meaningful parts of words/stems) and sub-object levels (patterns in objects).

In literature, we commonly come across with two prominent VL tasks: the image captioning (IC) and the visual question answering (VQA). In the first task, the model is asked to provide captions of an image, while in the second the model is asked to answer a question about the image. What differentiates our tasks to the typical VL ones is that our models need to understand, describe and answer questions related with the spatial aspects of the word, meaning that generic descriptions about the nature of the scene will not be considered (E.g. the caption "a cat is playing with the ball" is not acceptable. We prefer something similar to "a cat is to the left of the ball"). This requirement puts extra strain on the devised model architecture, since the training and evaluation protocol and the data gathering and selection should all be able to facilitate such an out-of-envelope approach to VL.

So, in the duration of this project we will first explore and provide models for spatial Vision-Language acting on still RGB images, then we will steer our efforts towards 3D-Language modelling, acting directly on raw depth images and lastly, towards the end of the project, we will experiment with Video-Language models trying to harness the additional temporal information.

3.3.1. Early models delivery

In order to facilitate the timely and unobstructed development across the VOXReality pipeline, we deemed necessary to define early in the project a standardized model delivery mechanism and an inference API that, more or less, will remain constant throughout the project's pilots and Open Calls (OC).

Instead of simply defining the templates, we preferred to adapt, package and deliver to the rest of the platform actual state-of-the-art (SOTA) VL models (not spatially aware though), to enable use-case developers to infer and use real working models, in a way similar to what they are about to experience with ours. So, we released two inferable SOTA VL models, one for each task (IC[4] and VQA [Tan & Bansal, 2019]) and an extra composite service that provides a full spatial description of an image exploiting Object Detection (YOLO-NAS[5]), transformers-based monocular Depth Estimation [Ranftl et al., 2021] and our in-house heuristics for sentence creation.

We decided that all our services provided to the project will be in the form of Docker images, GPU-accelerated when needed, exposing RESTful API[6] endpoints for inference. Details on the exact message formatting can be found in Appendix II. Additionally, the services will also be providing a web interface built around FastAPI[7] enabling GUI-based interactive inference, ideal for demonstration and debugging purposes. Figure 3.16 presents the endpoints for the early image captioning model, early visual question answering model, and composite spatial scene description (scenegraph) service used for the creation of our COCO extension dataset.

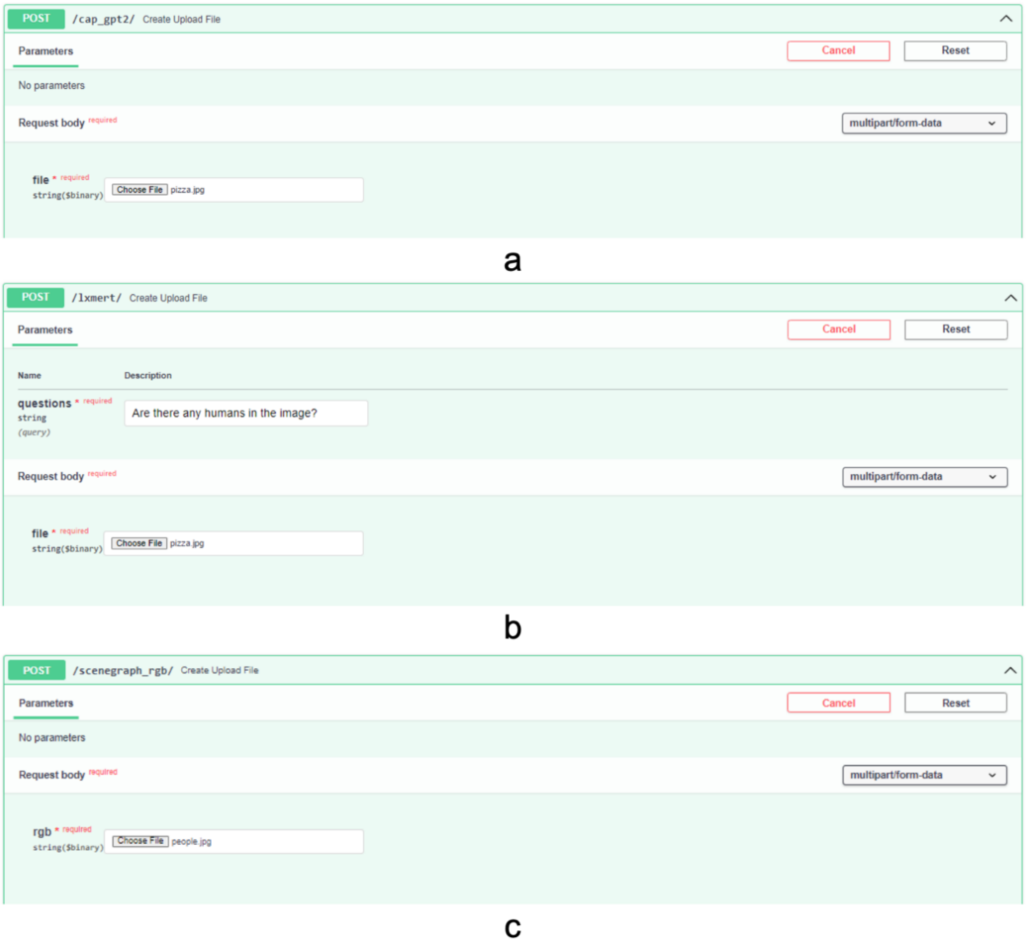


Figure 3.16. Early endpoints for a) image captioning, b) visual question answering and c) spatial scene description (scenegraph)

[4] https://qi-xin.github.io/image%20caption%20generation.pdf
[5] https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md
[6] https://docs.github.com/en/rest?apiVersion=2022-11-28
[7] https://fastapi.tiangolo.com/

3.3.2. Metrics and limitations

A general problem in evaluating NLP, and consequently VL, model predictions is that in human language it is common to find sentences that have almost the same meaning by utilizing synonyms or have the exact same meaning despite having different word ordering. On the other hand, it is also common having sentences that although contain the exact same words, word order changes can alter their meaning significantly. Moreover, sentences that have all words equal and in the same order, with the absence of even one word may totally lose their meaning or natural sound (E.g. the removal of one "not"). Others, with the addition of one redundant word can still have the exact same meaning. All these phenomena, in most cases, render the use of naive similarity or entropy metrics at least questionable for the task of fairly evaluating the output of an NLP model and even though they frequently participate in training loss functions, rarely should be used for actual evaluation.

There are multiple evaluation metrics proposed in literature, each one trying to alleviate different pitfalls of the aforementioned linguistic behaviours. So, it is safe to assume that there is no metric adequate to cover meticulously every aspect of the performance envelope and using multiple ones may usually be the preferred approach. Importantly, since our intended use is for the evaluation of spatiality-containing language, even the most widely used NLP-targeted metrics may not be able to capture our intentions. For this specific reason, we devised our own metric which in its simplicity we believe captures better the spatial accuracy of a model and is most relevant to our case. In the following paragraphs, we explain more about our devised evaluation metric.

When planning for our developments, we soon realized that all these of the well-established metrics might not be suitable for judging a produced sentence for its spatial accuracy. The reason is that the correctness of a sentence we care about does not lie in the top-level semantics of the words contained (their individual meaning) but in the existence of the objects described (correct object identification) and the correct description of the relationships between them (their relative positions). BLEU, ROUGE and METEOR may capture these relationships to some extent, but only incidentally. There has been works like [Kritharoula et al., 2023] and [Liu et al., 2022] that face similar challenges and develop their own evaluation procedures, trying to attack the problem each one from another angle. Our devised approach is described below:

Concurring with the general consensus, we believe that a single number cannot capture all aspects of performance in this challenging task. So, we devised a parametrizable metric that when calculated for the whole dataset intends to answer 2 questions; A: "What is the probability of a caption to be correct" (to describe a valid relationship) and B: "What is the probability of an image to be captioned correctly?" (to have a caption that describes a valid relationship between object in it). We need to remind here that our models generate a small paragraph describing an image and not a single sentence. Consequently, many captions may refer to the same image. Additionally, a different number of captions may be generated for each image. Let us call our metric "$OURS_{h\text{-}\{A,B\}}$". For each variation, we define h to be the number of sentences we wish to consider for each image (the first $h[1,MAX]$ captions generated). We decided to consider only *1,2,3* and the *MAX*imum number of captions the models can generate for each image. *A* and *B* indicate the question answered we mentioned earlier. Note that, we omit the *{1,2,3}-B* cases because, obviously, they are equal to their respective *A*-cases, as difference between *A* and *B* scores arises only when a different number of captions is considered between images (the case of *MAX-B*). In short notation, each of the variations can be defined as:

- $OURS_{\{1,2,3,MAX\}\text{-}A} = \frac{nTrue}{nTrue+nFalse}$

  where *nTrue* is the total number of captions that can be exact matched in their respective images' ground truth paragraphs and *nTrue+nFalse* is the total number of generated captions (in the case of *h=1*, *nTrue+nFalse=nImages*).

- $OURS_{MAX\text{-}B} = \frac{\sum_{i=1}^{nImages}\frac{nTrue_i}{MAX_i}}{nImages}$

  where $nTrue_i$ is the number of exact matched captions of image *i* in its ground truth paragraph and MAX_i is the maximum number of captions the model can generate for image *i* and can vary between images.

A question that naturally arises is "Why the exact matching between a generated caption and the respective image's ground truth is a robust and fair way to extract truthfulness?". Anecdotally, we realised that in our use case the models are trained on a very basic vocabulary that includes solely the object labels ("the cat", "the car", "the human", etc.) and a limited number of phrases that indicate relative positioning ("is in front of", "is behind", "is above", "is below", "is to the left of", "is to the right of"). In that context, the models negligibly fail to produce a grammatically valid sentence and any errors are almost always due to object misidentifications and/or wrong identification of their relative positioning, which is exactly what we look for. So, in fair terms, we can assume that exact searching of the generated captions inside the ground truth paragraph is reliable.

Our metric, of course, comes with its own limitations and considerations. The most notable one is being dependent on the correct identification of the described objects. Consider the example in Figure 3.17 with a flower in a vase and a kitten next to it.

Figure 3.17. The example image with a flower in a vase and a kitten next to it

Prediction:	"The cat is to the right of the vase?"
Ground Truth:	"The kitten is to the right of the vase."

The problem is that even if the relative position between the identified objects is correct and a human would totally make sense of the slight misidentification of the cat to be the kitten (and the kitten is also a cat), our metric would evaluate it being wrong. A more challenging example would be to misidentify the kitten as a leopard. In the context of only two objects being present and the feline nature of the leopard, a human would guess that it probably meant the kitten rather than the vase, and potentially evaluate the relationship as correct. As a countermeasure to all these, we could try to add a pre-processing step similar to METEOR's that use an external source to identify stems and synonyms, in an effort to minimize these effects; but still, it is obvious that there is no clear answer to these challenges.

3.3.3. The Vision (RGB)-Language case

Dataset

As we described in previous chapters, we aim to develop models for 3 cases: Vision (RGB)-Language, 3D (depth)-Language and Video-Language. To achieve spatial understanding in any case, having good amounts of data spatially annotated is of primal importance. We soon realized that there were not big repositories with such data openly available, at least in the form we imagined having them. So, we deemed necessary to extend a widely used VL dataset in the spatial domain to have the large amounts of data we needed to initiate the development of the models. COCO 2014-2015 dataset is considered one of the golden standards of the field, comprising about 164.000 RGB images paired with generic captions. In order to extend this massive dataset to the spatial domain, we devised a composite mechanism able to produce spatial captions, questions & answers and depth images for the whole dataset automatically. Note that, the depth images produced are needed internally by this mechanism to create the spatial captions and are not used directly anywhere by the models developed for the Vision-Language case. In the future, the 3D-Language models we are about to develop will probably process them, to profoundly understand 3D space. The core part of this mechanism, the "spatial scene description (scene-graph)" generator that produces for a given image a paragraph describing all the 1-1 relationships between the detected objects of the scene, is dockerized and released for use together with our early SOTA models. The algorithm we devised to automatically produce the spatial captions for all images is the following:

We take for granted that all positioning will be relative to the viewer's point of view and not according to the perceived orientation of the respective objects. So, when we say "to the left of the gorilla" we mean relative to our view, not the gorilla's view. We chose to evaluate all 1-1 relationships between the identified objects in a scene in 3-axes; lateral (left-right), longitudinal (in front of-behind) and height (above-below). We evaluate lateral and height relationships on the RGB images and longitudinal ones in their respective depth images. We produce the depth images using a SOTA transformers-based depth-estimation model [Ranftl et al., 2021] and since we are interested only in the objects' depth-ordering and not their exact depth, we can safely assume that this model is powerful enough to produce almost perfect results. We randomly sampled and verified manually the truthness of the inferred depth-ordering in about 200 out of the 164.000 created images. In any case, as we will describe in a moment, objects with close depth estimations are not even evaluated in this axis (remains inconclusive what is in front of what), so they cannot teach the model erroneously.

For each RGB image, the procedure initiates by detecting all the objects in the scene. This is done using the latest iteration of the SOTA Object Detector YOLO-NAS and results in a list containing the label, 2D-bounding box (y_{min}, y_{max}, x_{min}, x_{max}) and centre (c_x, c_y) coordinates of all detected entities. Then, for each item in this list we evaluate its relative position against all other items across the 3 aforementioned axes. To form the caption sentences we arrange and concatenate the objects labels with the definite article "the", together with a fixed set of phrases indicative of the spatial relationship between the involved objects, using a heuristic set of rules.

Using as example the lateral axis (x) and 2 objects A and B, we evaluate whether B's left-most bounding box boundary (Bx_{min}) is to the right of the right-most boundary of A's (Ax_{max}). If it is, then we form the sentences "The <B> is to the right of the <A>" and equally "The <A> is to the left of the <B>". If it is not, then we check whether their existing overlap is insignificant enough compared to the scale of the items, that it can still be considered that one item is to the right/left of another. Algorithmically, we evaluate if ($Bc_x>Ac_x$) AND ((Bc_x-Ac_x) / ($Ax_{max}-Bx_{min}$) > 3). If yes, then we still form the aforementioned sentences. If not, we discard this relationship and do not create any sentence. To continue, we repeat the exact same process for the height axis (y) evaluating By_{min}, Ay_{max}, Bc_y, Ac_y respectively, forming sentences using the words "above" and "below" this time. As we mentioned previously, the longitudinal (depth) axis is treated differently. Given that both the RGB and the depth images have the same resolution, and we know the placement of the various objects in the RGB image, we use the same coordinates to sample their depth from the depth image. For simplicity, we consider the bounding box's central pixel's depth value representative of the whole object's depth. To be on the safe side, we have set a threshold of at least 40 depth units [0-255] that two objects must differ in order to consider one in front of the other. So, we check if depth (Ac_x,Ac_y) - depth(Bc_x,Bc_y) > 40 to create the sentences "The <A> is in front of the <B>" and equally "The <B> is behind the <A>".

Apart from the image captioning task, in order to also enable the visual question answering task we need to create pairs of space-related questions & answers. An automated way to do that is to decompose each one of the captions into 2 self-evident questions & answers about the "What" and the "Where" of the described relationship. To give an example, solely from the caption "The <B> is to the right of the <A>" we can safely produce: Question 1 "Where is the <B>?" Answer 1 "To the right of the <A>" and Question 2 "What is to the right of the <A>?" Answer 2 "The <B>". In this manner, we can create a huge amount of Q&As. To summarize, for both tasks, all the possible sentences that the mechanism can automatically produce for a pair of objects A and B are the following 30, and together with all the possible object labels comprise the total of the training vocabulary:

The <B> is to the right of the <A>.	
The <A> is to the left of the <B>.	
The <B> is above the <A>.	
The <A> is below the <B>.	
The <B> is behind the <A>.	
The <A> is in front of the <B>.	
Question: Where is the <B>?	Answer: To the right of the <A>.
Question: What is to the right of the <A>?	Answer: The <B>.
Question: Where is the <A>?	Answer: To the left of the <B>.
Question: What is to the left of the <B>?	Answer: The <A>.
Question: Where is the <B>?	Answer: Above the <A>.

Question: What is above the <A>?	Answer: The <B>.
Question: Where is the <A>?	Answer: Below the <B>.
Question: What is below the <B>?	Answer: The <A>.
Question: Where is the <B>?	Answer: Behind the <A>.
Question: What is behind the <A>?	Answer: The <B>.
Question: Where is the <A>?	Answer: In front of the <B>.
Question: What is in front of the <B>?	Answer: The <A>.

Using the objects' labels, their bounding boxes, centres and the corresponding depth image, the automated mechanism can populate the above 30 sentence templates to create the set of captions and questions & answers we require. For example, using the RGB image and the depth image presented in Figure 3.18, descriptions such as "The person is to the right of the fire hydrant" and "The fire hydrant is in front of the person" can be generated.

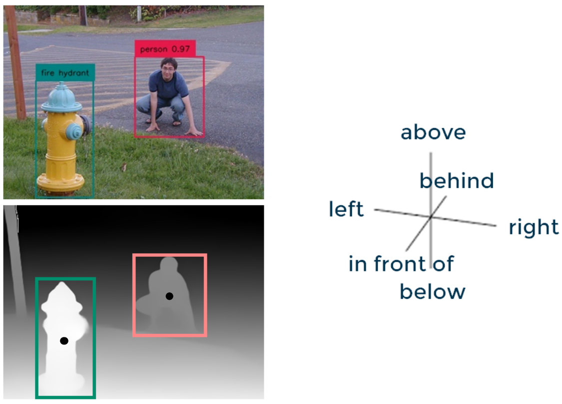


Figure 3.18. Example RGB and Depth image that includes a person and a fire hydrant

The maximum number of captions the mechanism can produce for an image is *nObjects x 6 x (nObjects-1) and* consequently twice that many questions & answers. To have an idea of the final dataset's extend, it contains 164.000 RGB images, equal number of depth images, each image contains on average 6 detectable objects leading to around 20.000.000 spatially aware captions and 40.000.000 sets of spatial questions & answers, all packed in about 70 GB of storage.

Model architecture

Stepping on the theoretical foundations described in the project's GA, we first looked for potential solutions into a couple-years-old prominent family of VL pre-training encoders. We studied and experimented with works like LXMERT [Tan and Bansal, 2019], VisualBERT [Li et al., 2019] and UNITER [Chane et al., 2020]. Although these works have shown, for both IC and VQA tasks, remarkable performance in various benchmarks, the language output they generate is limited and shallow. It was soon realized that in order to generate more natural and richer textual representations we needed to migrate to a Vision Encoder-Decoder scheme. This versatile family of solutions can generate sophisticated and natural-sounding language that is currently considered the state-of-the-art of VL modelling.

Building on the Hugging Face's vision encoder-decoder model[8], we experimented with various combinations of pre-trained transformer-based vision models as the encoder (i.e. ViT [Dosovitskiy et al., 2020], BEiT [Bao et al., 2021], DeiT [Touvron et al., 2020], and Swin [Liu et al., 2021]) and pre-trained transformer-based language models as the decoder (i.e. RoBERTa [Liu et al., 2019], GPT2 [Radford et al., 2019]). The effectiveness of initializing image-to-text-sequence models with pretrained checkpoints has been shown in various works like [Li et al., 2021]. Our SOTA experiments were conducted in order to enhance our understanding of the Hugging Face's framework and decide which combination of vision encoder and language decoder we should train to cover our needs. We need to mention here that the SOTA pre-trained models we tested were mostly trained on the standard COCO dataset and thus showcase no notion of spatiality in their generated captions and question answers. So, we chose to train our selected encoder-decoder combination on our spatially aware extension of the COCO dataset, from scratch (random initial model weights). Of course, when our pre-training is done, we will finetune our model to our specific use cases in Transfer-Learning manner [Zhuang et al., 2019]. Judging from the pre-trained versions' performance and weighting in our hardware limitations, mainly in GPU memory, we opted to develop a combination of ViT for the vision encoder and GPT2 for the language decoder.

Beginning from the encoder, ViT processes the input image by dividing it into fixed-size (16x16) non-overlapping patches (input image dimensions: 224x224). In ViT's context, the patch is treated as the unit for further processing. Each patch is linearly embedded to obtain a patch-embedding. The idea is to capture local information within each one of these, while enabling interactions between them in the subsequent transformer layers. To achieve that, the model needs to introduce positional embeddings to capture the spatial relationships between the different patches of the image. Unlike sequence data where the order of tokens is inherent, images lack a natural ordering of pixels, and therefore, positional information needs to be injected explicitly. In ViT, the patch's position is represented by a unique positional embedding vector created using a combination of trigonometric functions (sine and cosine), ensuring that the positional embeddings have a smooth and continuous pattern. These positional embeddings, added to the patch embeddings, form the input sequence for the transformer encoder. Then, the transformer-based encoder attends to the relationships between the different patches, capturing both local and global contextual information. The final output of the encoder is a set of contextualized embeddings for each patch, representing a deep and rich understanding of the image content.

As regards the decoder, the GPT-2 is a language model designed for generating coherent and contextually relevant sequences of language tokens. In this case, it takes as input the output embeddings from the ViT encoder and fuses them with its own contextual understanding of language during the decoding process in an autoregressive manner, generating one token at a time conditioned on the previously generated tokens. It uses a transformer-based architecture, similar to the original GPT-2 language model, consisting of multiple layers of transformer decoder blocks. Each block contains a multi-head self-attention mechanism and a feedforward neural network. The self-attention mechanism allows the model to attend to different parts of the input sequence, capturing dependencies and context. Similar to the ViT encoder, the GPT-2 decoder utilizes positional embeddings to provide information

[8] https://huggingface.co/docs/transformers/v4.36.1/en/model_doc/vision-encoder-decoder#transformers.VisionEncoderDecoderModel

about the sequential order of tokens in the generated output. While the ViT uses sinusoidal positional embeddings, the GPT-2 decoder uses learned positional embeddings. During the decoding process, the GPT-2 decoder attends to the encoded visual information provided by the ViT encoder. The attention mechanism allows the model to focus on relevant parts of the image content when generating each token in the output sequence.

Note that the architectural analysis given above is indicative to the image captioning task. Slight adaptations in architecture and, of course, training it with different data (pairs of questions and answers) is needed to adapt our model to the visual question answering task. For VQA, the model takes both an image and a textual question as input. The image is processed by the ViT encoder to obtain visual embeddings and the textual question is typically encoded separately, either using a transformer-based architecture or other encoding mechanisms. The ViT-encoded visual embeddings and the encoded question must be combined to form a joint representation. This integration may involve concatenation, attention mechanisms or other fusion techniques to effectively merge the two modalities. This joint representation is then used as input to the GPT-2 decoder. We will adapt our model for use in the VQA task in the near future. To give a perspective of the developed model's size, it incorporates about 239.000.000 parameters. Figure 3.19 illustrates our architecture.

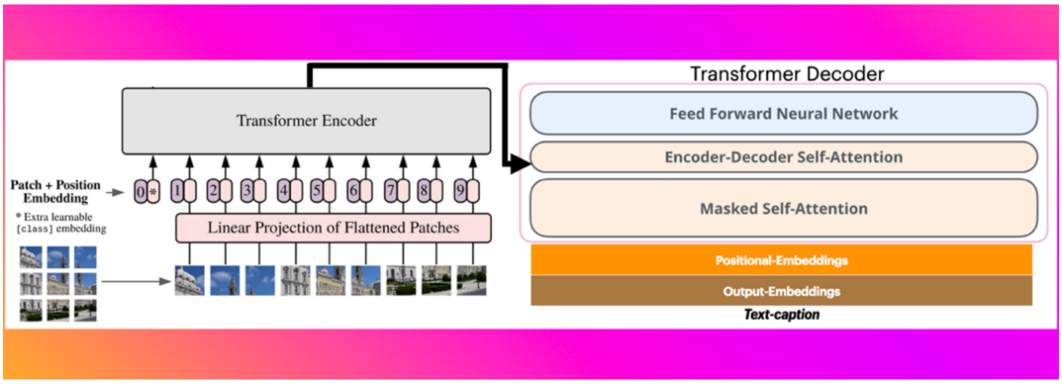


Figure 3.19. The architecture with ViT encoder and GPT2 decoder for the IC task

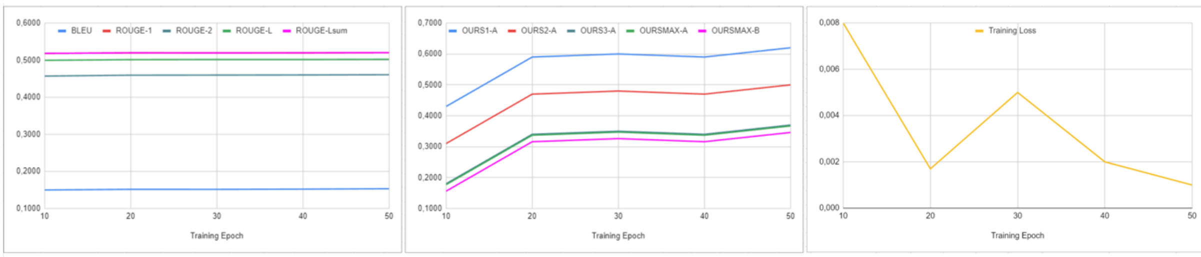
Training and results

Objective of the training process is to minimize the difference between the predicted caption generated by the GPT-2 decoder and the ground truth caption associated with the input image. This involves using a suitable loss function, in this case the cross-entropy loss, to measure the dissimilarity between the predicted and actual sequences of tokens. We need to mention here that, like in many other cases, this loss function (albeit differentiable) may not be very good at capturing the task's nuances and correlate highly with the better-suited metrics (like OURS). The optimization of the loss function is being handled by AdamW [Loshchilov & Hutter, 2017] with a linearly diminishing Learning Rate (LR) as the epochs progress. For training, we used 64.172 samples and for testing 31.156. Periodically, every 10 epochs, we evaluate our model on 10 metrics. The standard BLEU, ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-L_{sum}, and our own devised metric in all its variations $OURS_{1\text{-}A}$, $OURS_{2\text{-}A}$, $OURS_{3\text{-}A}$, $OURS_{MAX\text{-}A}$ and $OURS_{MAX\text{-}B}$. In Table 3.13, we present in detail the results of our model's evaluation study across a training session of 50 epochs, that took around 100 hours of processing on 2x Nvidia RTX 3090 24GB GPUs.

Table 3.13: Evaluation scores on the test-set and training loss across training epochs

Evaluation Metric	Training Epoch				
	10	20	30	40	50
BLEU	0,1499	0,1516	0,1515	0,1521	0,1533
ROUGE-1	0,5183	0,5196	0,5195	0,5196	0,5202
ROUGE-2	0,4570	0,4594	0,4597	0,4601	0,4607
ROUGE-L	0,4996	0,5014	0,5018	0,5017	0,5022
ROUGE-Lsum	0,5183	0,5196	0,5194	0,5196	0,5202
OURS1-A	0,4300	0,5900	0,6000	0,5900	0,6200
OURS2-A	0,3100	0,4700	0,4800	0,4700	0,5000
OURS3-A	0,1800	0,3400	0,3500	0,3400	0,3700
OURSMAX-A	0,1770	0,3370	0,3470	0,3370	0,3670
OURSMAX-B	0,1560	0,3160	0,3260	0,3160	0,3460
Training Loss	0,008	0,0017	0,005	0,002	0,001

As it is evident from Figure 3.20, the model keeps improving until the designated completion of the experiment. Note that, this training session's length is chosen for the purposes of this deliverable document. The actual model to be released will be left to train for a couple-hundreds of epochs. Delving into the details, we observe the typical phenomenon where the model showcases a rapid initial improvement until around 20 epochs (both regarding the train-set loss and the test-set evaluation metrics), and then the progress slows-down, partially due the gradual decrease of the Learning Rate by the linear scheduler. Interesting is that, although the loss value fluctuates quite a lot between 20 and 40 epochs into the training, the evaluation metrics do not follow this trend, remaining almost constant for these 20 epochs and then improving further, together with the further decrease of the loss (possibly escaping out of a local minimum), hinting at the healthy behaviour of the session.



a b c

Figure 3.20. Progression across training epochs of a) standard evaluation metrics, b) variants of OURS metric and c) the respective training loss

Decomposing the evaluation metrics' behaviour, we observe that all the 5 standard and the 5 variants of OURS improve across training, though to different extend. The BLEU, ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-L_{sum} metrics improve marginally, while $OURS_{1\text{-}A}$, $OURS_{2\text{-}A}$, $OURS_{3\text{-}A}$, $OURS_{MAX\text{-}A}$ and $OURS_{MAX\text{-}B}$ improve at a healthier rate. This is particularly aligned with our manual observations of the generated captions that, indeed, seem to improve. As regards the behaviour of the different variants of OURS metric, we conclude that they follow their theorized trend. As it was expected, the $OURS_{1\text{-}A}$ variant achieves the highest accuracy of all (62% at epoch 50), since it evaluates only the first-generated caption of the model for each image. It is reasonable to expect that the more words we pump out of the model the more it will fail and, consequently, all other variants of OURS may showcase lower accuracy. This can be attributed to the autoregressive nature of GPT-2 where each new word is conditioned on the previous ones, and thus any errors accumulate leading to ever-increasing chance of generating nonsense. Indeed, as the results suggest, each other variant of OURS that considers more sentences (2, 3 and the maximum the model can produce for an image - the MAX_i) shows lower accuracy. Interestingly, we observe that past the first 3 generated captions per image considered, the accuracy of the model seems to not drop further, rather stays constant.

Furthermore, to study the difference between our proposed metric and the conventional metrics of the field and the potential usefulness of our metric, we conducted as small study examining all of its variants' correlation with the rest of the metrics, as well as the correlation between all of them for the sake of completeness. What can immediately be noticed is the highly positive correlation of all the standard metrics with each other. We can safely assume that, at least in our specialized dataset, good score in either of the BLEU, ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-L_{sum} metrics would possibly translate to a good evaluation score in any of them. This behaviour is supported in literature, and is reasonable, given that they were all developed to evaluate similar linguistic scenarios. At the same time, all the variants of OURS metric showcase strong positive correlation with each other as well, behaviour equally reasonable given their similar workings. Reaching to the very core of our study, the most valuable finding and primal motive behind its conduction is that the OURS metric, in all of its variants, behaves independently to the rest of the existing established metrics. This fact strongly supports that, at least in the context of our specialized needs, the OURS metric succeeds in capturing a facet of the performance envelop which evades the rest of the well-known metrics; ultimately justifying its creation. Table 3.14 presents these results, where green coloured cells indicate positive correlation, red coloured cells indicate negative, and the white tones in-between suggest independence between the examined metrics.

Table 3.14. Pearson correlation coefficients between all evaluation metrics.

	BLEU	ROUGE -1	ROUGE -2	ROUGE -L	ROUGE -Lsum	OURS 1-A	OURS 2-A	OURS 3-A	OURS MAX-A	OURS MAX-B
BLEU		0,970	0,953	0,916	0,976	0,019	0,375	0,384	0,327	0,342
ROUGE-1	0,970		0,979	0,969	0,998	-0,081	0,210	0,185	0,227	0,305
ROUGE-2	0,953	0,979		0,989	0,971	-0,242	0,201	0,165	0,106	0,248
ROUGE-L	0,916	0,969	0,989		0,954	-0,252	0,182	0,113	0,112	0,305
ROUGE-Lsum	0,976	0,998	0,971	0,954		-0,063	0,204	0,194	0,228	0,276
OURS1-A	0,019	-0,081	-0,242	-0,252	-0,063		0,593	0,628	0,897	0,618
OURS2-A	0,375	0,210	0,201	0,182	0,204	0,593		0,970	0,829	0,823
OURS3-A	0,384	0,185	0,165	0,113	0,194	0,628	0,970		0,802	0,689
OURSMAX-A	0,327	0,227	0,106	0,112	0,228	0,897	0,829	0,802		0,875
OURSMAX-B	0,342	0,305	0,248	0,305	0,276	0,618	0,823	0,875	0,875	

3.3.4. Next Steps

In the prospect of the upcoming pilots, the Open Calls and the maturation of the project towards its completion, we have set a number of actions we plan to take in order to progress successfully.

1. We will let our model train further to some hundreds of epochs and then pack it into a new Dockerized service, updating the project's toolchain. Furthermore, we will release it on the Hugging Face project repository, creating the first publicly available entry of our VL models. The new endpoint to be added to the project's API for inferencing this, will be "/rgb-language_cap".
2. Apart from the IC task, we will extend the use of our Vision (RGB)-Language model to the VQA task. This will happen with training on our created Q&A dataset and some minor architectural changes. The new endpoint to be added to the project's API when the new model is ready, will be "/rgb-language_vqa".
3. In collaboration with 2 of our use cases' developers (AR Theatre, Virtual Conferences), we will capture and manually annotate 2 datasets with realistic views and actions from their scenarios, to finetune our models to the downstream tasks.
4. We will begin the development of our 3D-Language and possibly Video-Language models, utilizing the depth dataset we already created and new datasets that we will create with material from the project's pilots. The new endpoints to be added to the project's API upon completion of the new services, will be: "/3d-language_cap", "/3d-language_vqa", "/video-language_cap" and "/video-language_vqa".
5. Our current notion of the spatial relationships between the objects is coarse and relative in nature; hence we describe them only with "left", "right", "above", "below", "in front of" and "behind". Our aim is to explore the potential generation of "metrically accurate" descriptions of the scene by training the models on captions that contain textually the distances between the objects (I.e. "The cat is three meters to the left of the bus"). In order to acquire the large amounts of metrically annotated data we need to train our models on, we will investigate the creation of a synthetic dataset in which we can accurately manipulate the placement of the objects to create metrically accurate captions for a limitless number of arrangements.
6. In case we succeed with the creation of our metrically accurate VL models, we will need to extend our metric, or even devise a new one, in order to capture correctly the newly added feature of the predictions.
7. We will release our spatially aware extension of the COCO dataset to the public.

3.4. Conversation Agents (CA)

In VOXReality, we aim to develop two conversation agents with two different tasks: 1) VR Conference Conversation Agent and 2) AR Training Assistant.

3.4.1. VR Conference Conversation Agent

The VOXReality VR Conference Conversation Agent is specifically designed to enhance the user experience at the conference, serving a multifaceted role. It is tasked with guiding users through the conference venue, providing essential information about the conference program, and offering insights into the trade show.

In this section we present the methodology followed for the implementation of this conversation agent. We explain in this section the creation of a robust Natural Language Understanding (NLU) model, creation of a dataset and extending it to the fine-tuning process of the model to ensure its effectiveness and accuracy in understanding and processing user queries, the mechanisms behind the conversation agent's workflow and the fine-tuning process applied specifically to the conference agent to provide accurate navigational assistance within the VR conference environment.

Natural Language Understanding (NLU) Model

The primary objective of the NLU model is to accurately determine whether a user's query is related to seeking navigation guidance within the conference venue, inquiring about the conference schedule, or requesting insights about the trade show. It's important to note that while this model plays a crucial role in the research phase, it is not directly integrated into the final workflow of the conference's conversation agent. However, it is planned to be made available for future open calls, offering broader applicability and utility.

Model Selection

In the dynamic landscape of NLU, the selection of an appropriate model is a critical decision that significantly impacts the success of language processing tasks. The T5 model, known for its exceptional adaptability and flexibility, has emerged as a leading choice for handling a wide array of NLU challenges. Its ability to seamlessly adjust to various linguistic tasks, coupled with a unified methodology, greatly eases the process of integrating it into different systems. This makes the T5 model not only highly efficient but also remarkably adaptable in addressing the diverse needs of NLU applications.

For the navigation assistant task, the model that was selected is the T5-Small. This choice was guided by the nature of our NLU task, which does not necessitate the advanced capabilities of more substantial models. The T5-Small perfectly meets our requirements, striking an optimal balance between performance and computational efficiency. It adeptly handles the language processing needs with the right level of complexity, ensuring both accuracy and resource economy. Thus, the T5-Small emerges as the most fitting solution for our NLU objectives, providing just the right mix of efficiency, affordability, and computational effectiveness.

Dataset Description

To develop a stable NLU model using T5-Small, the creation of a dataset tailored to specific use cases is essential. Following the dataset structure utilized by Convlab-3 [Zhu et al., 2022] for training, the Multiwoz [Budzianowski et al., 2018] dataset was beneficial for ensuring well-structured and appropriately formatted data. Each dataset sample starts with a user query, which is a question directed at the model. For example, a query might be: "*Can you find an Indian restaurant for me that is also in the town center?*" The corresponding label for this query is "*[inform][restaurant]([area][center], [food][Indian])*". This labelling system denotes the general user intent, such as seeking information about a restaurant, and specific details like desired location and food preferences. A sample data from the NLU dataset is presented below:

```
{"context": "How can I go to the conference room?", "dialogue_acts_seq": "[request][direction]([destination][conference room])"},
{"context": "Is there a company that is offering solutions for automation?", "dialogue_acts_seq": "[request][trade_show]([product_info][Automation])"},
{"context": "What time does the 'Socializing' event ends?", "dialogue_acts_seq": "[request][program_info]([end_time][Socializing])"},
{"context": "I need to get there. Can you guide me?, "dialogue_acts_seq": "[request][direction]([destination][unspecified]"},
{"context": "What event will be held at 11:30?, "dialogue_acts_seq": "[request][program_info]([event_at_time][11:30]"},
{"context": "Which company is present at booth B02?", "dialogue_acts_seq": "[request][trade_show]([exhibitor][B02]"}
```

Inspired by the structure of Convlab-3's dataset, a new dataset is created with a similar format. This dataset is designed to train a model specifically for a conference assistance application, where it is expected to deliver navigational guidance within the conference venue, details about the conference program, and offer insights about the trade show, including information on the participants and their respective products. Each sample begins with the "context" field, containing the user's query. Following this is the "*dialogue_acts_seq*" field, which holds the corresponding label for the query. This label is crucial for the model to discern the user's initial intent, such as seeking directions, program details, or trade show information.

The structure of the rest of the label varies depending on this intent. For navigation-related queries, the label is designed to identify the user's desired destination. If the destination is not specified in the user's query, the label marks the destination as "*unspecified*". At this point, it should be mentioned that the destinations are unrelated to the rooms of the conference venue of our use case. This approach is adopted to provide the model with a generalized capability, ensuring it does not only learn to identify specific rooms but can adapt to a variety of scenarios. Similarly, for program information queries, the label is structured to extract all related details from the user's request. For instance, as shown in the example above under the program information category, if a user asks about when an event ends, the label includes "end_time" to indicate the user's interest in the event's end time and the event's name. The trade show examples are set up in a similar way to the program information, organizing labels to gather detailed information from questions about the trade show.

The dataset, comprising 498 human-annotated samples, is evenly distributed across three categories: navigation, program information, and trade show, with each category containing 166 distinct samples. This dataset is divided into training, validation, and test sets. For the training set, 100 samples from each category are selected, totaling 300 samples, while the test and validation set includes 99 samples. This approach not only ensures a balanced representation of each category within the dataset but also maintains the quality and relevance of the annotations. Detailed information of the data distribution is presented in Figure 3.21.

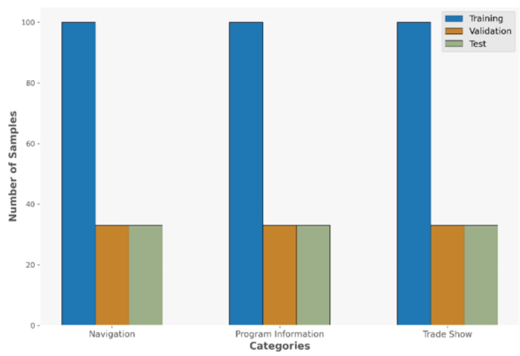


Figure 3.21. Data distribution across categories

Model Fine-tuning Process

The fine-tuning of the T5-small NLU model is carried out using the Convlab-3 platform, specifically designed to adapt different sizes of the T5 model to custom datasets. Convlab-3's robust framework provides an integrated solution for the entire fine-tuning process, ensuring that the model is optimally trained for the task.

The initial step in the model training process involves a comprehensive pre-processing phase, crucial for adapting the dataset for effective use with the T5 model. This phase includes tokenizing both the input queries and the target responses, vital for converting natural language into a format suitable for the T5 model's processing. In addition to tokenization, the Convlab-3 platform efficiently manages the challenges of padding and truncation. Moreover, the platform also includes an evaluation process as part of the training pipeline, involving testing the model on unseen data to measure metrics such as the F1 score, which provides insights into the precision and recall of the model's predictions.

Experimental Results

To guarantee a structured and efficient training procedure, a step-by-step method is implemented. The model is first fine-tuned and evaluated using only the navigation category samples. This focused approach allows for detailed monitoring of the model's performance in a specific context. For the navigation category, 100 samples are used for training, with 33 samples each allocated to testing and validation. The results from this initial phase were highly encouraging, with the model achieving an F1 score of 90.4%. Following the success in the navigation category, the same fine-tuning process was applied to the program information and trade show categories. The model continued to exhibit strong performance, achieving F1 scores of 91.8% and 95% in these categories, respectively. This demonstrates the model's adaptability and effectiveness across different types of user queries.

In the final phase of training, samples from all categories are combined to provide a comprehensive dataset. This allows the model to be trained across a broader spectrum of queries, simulating more closely the variety it would encounter in real-world applications. The overall F1 score for the model, after being trained on this combined dataset is 95.2%. The detailed results of these training phases, illustrating the model's performance across different categories and in the comprehensive dataset, are presented in the Table 3.15.

Table 3.15. Experimental results of the NLU model

Category	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Navigation	89.9	90.4	90.4	90.4
Program information	88.4	90.3	93.3	91.8
Trade show	95	95	95	95
All Categories	94	94.3	96.1	95.2

Conversation Agent

For the Conference Dialogue Agent, the Llama2-Chat 7B model was selected. This model, is specifically designed for chat and dialogue scenarios, making it a natural fit for an agent focused on conversational interactions in a conference setting. The 7B version of the Llama2-Chat model stands out for its efficiency and effectiveness. It is particularly well-suited for environments where computational resources might be limited or for applications that need to be cost-effective. This is a crucial consideration for deployment in varied settings, where the availability of resources can significantly impact the functionality of the agent.

Another key aspect of the 7B model is its ability to provide fast response times. In a conference environment, where interactions are real-time and often require quick exchanges, the speed of the model is essential. The model is designed to maintain a fluid and natural conversation flow, a critical factor in ensuring user engagement and satisfaction. Despite its relatively smaller size compared to the other models in the Llama2-Chat range, the 7B version does not compromise on performance. It can handle the spectrum of queries and interactions, from basic informational requests to more complex dialogue. This model can efficiently manage basic to moderately complex dialogues, ensuring effective communication without the need for the computational resources demanded by larger models.

In summary, the Llama2-Chat 7B model is a well-rounded choice for the Conference Dialogue Agent. It strikes a balance between efficiency, speed, and performance, aligning well with the practical requirements of a conference environment. This model's capabilities ensure that the agent can handle a variety of interactions effectively, making it a suitable tool for enhancing the conference experience through intelligent and responsive communication.

Workflow Description

Designing the conference agent's workflow presented a significant challenge due to its complex, multi-stage response generation process. To address this, the LangChain [9] framework is utilized. LangChain is an innovative, open-source Python library that simplifies building applications with language models like GPT-3 from OpenAI. It employs "chains," which are sequences of modular components, enabling flexible and efficient construction of language-based applications. This framework stands out for its versatility, allowing for the integration of different processing stages, such as context management, response generation, and post-processing. Such modular design makes LangChain particularly advantageous for creating intricate language interactions and tailoring them to specific needs.

The workflow for the conference agent (Figure 3.22), is a multi-stage process. It consistently employs a single model, the Llama2, across its chains. Initially, a brief overview of each component is presented, followed by a more detailed explanation in the following subsections.

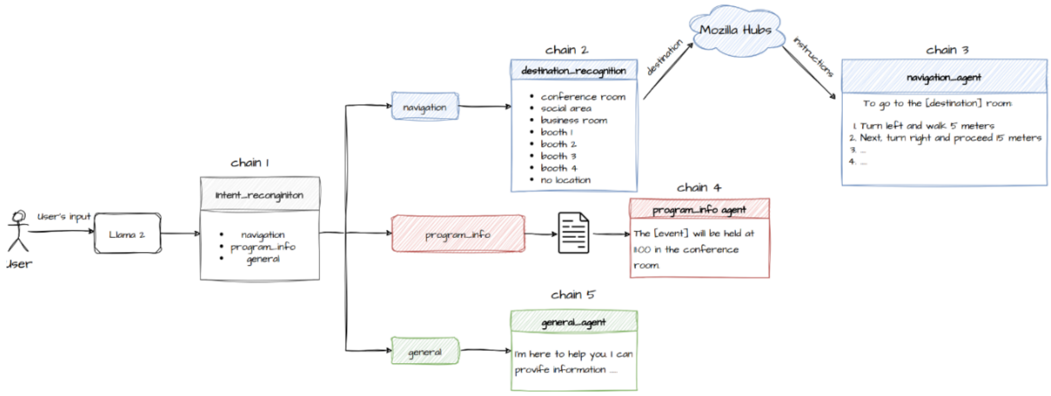


Figure 3.22. Conference agent workflow

[9] https://github.com/langchain-ai/langchain

The first chain employs the model to analyse user input, categorizing it into three distinct types: navigation, program information, and general queries. The 'navigation' type refers to user requests for directions to a specific room. In case that the model intent is navigation, the second chain follows that determines the user's intended destination. This step is crucial for integrating with Mozilla Hubs to identify the shortest path to the desired location. Once the shortest route is established, the third chain takes the navigation instructions and delivers the response to the user. In the second scenario, the 'program information' category involves queries about the conference schedule. In this case, Llama2 relates to a pdf file where the pdf file includes the program of the conference. In this case, chain 4 is activated by combining user’s input and the information of the pdf file response to the user. Lastly, the 'general' category handles all other user inquiries not related to directions or program details.

Intent Recognition

When a user poses a question, the initial and crucial step for the agent is to accurately identify the user's input. This initial step is the key to ensuring that the subsequent process and response align correctly with the user's query. If the model misinterprets the user's intent, the resulting answer will likely be unrelated to the original question. To successfully build an intent recognition model using Llama 2, the capabilities offered by LangChain are utilized. LangChain provides a specialized feature for developing prompt templates based on few-shot examples. This process involves creating several task-specific examples. These examples, coupled with a system prompt, enable the model to adeptly handle the desired task. In essence, this feature is providing by the *FewShotPromptTemplate* function, and it is a powerful tool for creating structured, example-based prompts that guide the model's responses in a specific and controlled manner, making it highly useful for complex querying and data processing tasks.

For the conference agent's intent recognition task, the model is designed to face a classification problem. Based on the user's input, it must categorize the query into one of three labels: 'navigation', 'program information', or 'general'. To assist the model in this process, specific examples were initially created to guide its comprehension and enhance its ability to accurately classify queries. A few examples for intent recognition task are presented below:

```
{“input”: “Where is the main auditorium located?”, “output”: “navigation”},
{“input”: “What time does the keynote speech start?”, “output”: “program_information”},
{“input”: “Hello how are you?”, “output”: “general”},
{“input”: “How do I get to the registration desk?”, “output”: “navigation”},
{“input”: “Who is the speaker for the afternoon session?”, “output”: “program_information”},
{“input”: “Can you help me find information about the conference?”, “output”: “general”}
```

In addition to these examples, a carefully selected system prompt is employed to further guide the model's focus. This prompt was finalized after experimenting with various combinations to determine the most effective one and outlines the task for the model, ensuring that it focuses on categorizing the query into the correct label without providing extraneous details.

Destination Recognition

The approach used for intent recognition was similarly applied to destination recognition. In this scenario, the model's objective is to pinpoint the user's desired destination based on their input but also to have the ability to recognize if the requested destination is not part of the conference venue. Utilizing few-shot prompting examples, user inputs are paired with their corresponding correct outputs that the model is expected to generate. A crucial part in this process is the system prompt. Through a meticulous prompt engineering process aimed at identifying the most effective system prompt for this specific use case. A few examples for the destination recognition task are presented below:

```
{"input": "Can you guide me to the booth 1?", "output": "booth 1"},
{"input": "I want to go to the social area.", "output": "social area"},
{"input": "I want to leave. Where is the exit?", "output": "exit"},
{"input": "Where is the business room located?", "output": "business room"},
{"input": "How do I get to the conference room?", "output": "conference room"},
{"input": "How do I get to the gym?", "output": "no location"}
```

In the prompt, the labels represent all the available locations within VOXReality's conference venue. It is designed to be adaptable, meaning that the prompt can be modified to suit different conference venues with their respective unique rooms. Additionally, the prompt specifies that if the user requests a destination not listed in the venue's specific configuration, the response should be classified as "no location." This feature ensures that the model remains adaptable and applicable across various settings, providing accurate and relevant responses based on the particular layout and offerings of each conference venue.

Navigation Agent

The navigation agent is a key component within the workflow of the conference dialogue agent. The agent handles two types of input: user queries and Mozilla Hubs' navigation instructions. To formulate an appropriate response, the model integrates these two inputs. Mozilla Hubs provides navigation instructions that are necessary for guiding users effectively. These instructions form the core of the navigation process, ensuring users reach their desired destinations accurately. The types of instructions are:

- **Turn around**: Typically, the initial instruction, this directive assists users in aligning themselves correctly at the onset of their journey. It ensures that the user starts from the correct orientation, which is crucial for the accuracy of subsequent directions.
- **Move [number] meters**: This command specifies a precise distance for the user to travel in a straight line. It provides clarity on how far to proceed before the next action, aiding in maintaining a clear and measured progression along the route.
- **Turn right/ left**: These instructions guide the user in making accurate turns. Whether a right or left turn, they play a critical role in keeping the user on the intended path and preventing deviations that could lead to confusion or longer routes.
- **Stairs up/ down**: These commands are particularly important for vertical navigation. They direct users when ascending or descending stairs, ensuring safe and efficient travel between different floors or levels within the environment.

The model's ability to seamlessly blend user queries with these structured navigation cues is what makes the navigation agent essential to the overall efficiency and effectiveness of the conference dialogue agent in a Mozilla Hubs environment.

To optimize a user-friendly agent's behaviour in handling navigation inputs, a specialized system prompt is implemented. This process, known as prompt engineering, involves rigorously testing various system prompts to ensure that the model generates the most effective responses. The chosen prompt aids the model in addressing situations where user instructions are either uncertain or involve requests for non-existent locations within the conference venue. It is especially helpful in guiding the model during scenarios where the user either fails to specify a destination or selects an unavailable one. In such instances, the prompt instructs the model to inform the user about the available rooms at the venue.

The navigation agent is a crucial component of the conference dialogue system, adeptly handling user queries and Mozilla Hubs' navigation instructions. Its advanced integration and prompt engineering process enable it to adapt to various conference venues, providing precise and relevant navigational assistance. This adaptability marks a significant advancement in virtual assistance, enhancing user experiences in complex virtual environments like Mozilla Hubs.

Program Information Agent

The program information agent functions as a part of a larger system designed to interpret and respond to user queries about conference program details. Initially, if the user's input is identified as program information from the intent recognition component, the program information agent activates.

Specifically, the agent is using the *Retrieval QA* function provided from Langchain. The agent is connected to a comprehensive database, typically a PDF file, which contains detailed information about the conference program. Using advanced search and retrieval techniques, the agent scans this document to locate the exact information relevant to the user's question. The strength of this system lies in its ability to parse large volumes of text data and extract specific pieces of information. The agent, by understanding the context in which they appear, ensures that the responses are accurate and relevant to the user's query.

In line with the system's guiding prompt, the agent is directed to be a helpful navigator, focusing on delivering accurate, concise, and reliable information. This prompt ensures that responses are direct and contain only the necessary details asked by the users. The agent's primary mission is to assist users by providing detailed information about the schedule, speakers, events, locations, and other specific details included in the conference program.
This approach ensures that users receive quick, accurate, and contextually relevant answers to their questions about the conference program, enhancing their experience and making information access more efficient.

General Agent

The final element in the system's workflow is the general agent, a component designed to address a wide range of user inquiries that fall outside the specialized functions of other agents, such as navigation or specific conference program details. This agent comes into play when the intent recognition component classifies an inquiry as 'general'.

In its operation, guided by a carefully structured system prompt, the general agent's primary objective is to enhance the overall user experience. It achieves this by delivering clear, precise guidance, informing users about the types of assistance it can offer. This approach is strategically focused on ensuring that users receive assistance that is not only relevant but also precisely aligned with their immediate queries and requirements.

The essence of the general agent's role is to facilitate an interaction that is both efficient and user-friendly. By providing succinct, relevant responses, it helps users navigate their queries with ease, making the process of obtaining information seamless and straightforward. The general agent, with its broad scope and user-centric design, is therefore an indispensable part of the system, ensuring that users' general inquiries are addressed promptly and effectively.

Fine tuning of the navigation agent

While the model initially performed well in delivering navigation instructions, some problems appeared. Firstly, one of the primary concerns was the variability in answer format. This inconsistency not only creates a lack of uniformity across different responses but also potentially leads to confusion for users. For example, some answers may be presented in a detailed, step-by-step manner with clear numerical ordering, while others might be more narrative in style, lacking clear separation of steps.

Secondly, it included unnecessary navigation details that potentially misinformed the users. In the context of the example presented below**Error! Reference source not found.**, phrases like “Face the direction you came from,” “until you reach the next landmark,” and “The height difference is approximately 5 meters” were extraneous and not pertinent to the conference venue, adding confusion. Lastly, the incorporation of too much non-essential information resulted in overly lengthy responses, which could be incomprehensible for users.

```
{
“input”: “How do I get to the tech booth?”
“navigation_intructions”: “turn around, move 8 meters, turn left, move 12 meters, stairs up, turn right, move 10 meters, turn left, move 3 meters",
“output”:  “Great! Here are the step-by-step instructions on how to get to the tech booth based on the information provided:\n\n### Instruction 1: Turn Around (8 meters away)\n* Face the direction you came from and turn around to face the venue entrance again.\n\n### Instruction 2: Move 8 Meters Forward\n* Walk straight ahead for approximately 8 meters until you reach the next landmark.\n\n### Instruction 3: Turn Left (12 meters away)\n* Once you have reached the end of the hallway, turn left and walk down the new corridor.\n\n### Instruction 4: Climb the Stairs (up, 5 meters high)\n* Locate the staircase located nearby and climb the stairs to go up. The height difference is approximately 5 meters.\n\n### Instruction 5: Turn Right (10 meters away)\n* After reaching the top of the stairs, turn right and continue walking along the corridor.\n\n### Instruction 6: Move 10 Meters Forward\n* Walk straight ahead for approximately 10 meters until you reach the next landmark.\n\n### Instruction 7: Turn Left (3 meters away)\n* Once you have walked 10 meters forward, turn left and walk towards the tech booth.\n\n Congratulations! You have now arrived at the tech booth.”
}
```

Considering these challenges, and after various system prompts were tested without achieving the desired format in navigation responses, a comprehensive fine-tuning process was initiated. The fine-tuning process was implemented not only to address the specific formatting issues identified in the responses but also to enhance the overall clarity and usability of the guidance provided by the agent. This step was essential for the system's evolution, ensuring that the navigation instructions provided were both user-friendly and in alignment with the intended navigational objectives.

The following subsections will delve into the specifics of the fine-tuning process, describing from the dataset creation process to the strategies and methodologies employed to refine the system's capabilities and achieve the desired level of performance and user experience.

Dataset Description

The creation of an extensive dataset was a fundamental step in the development of the navigation agent. This carefully structured dataset is comprised of three main components, each playing a vital role in ensuring the agent's effective response to user queries with accurate and useful navigation instructions.

Firstly, the dataset starts with the "input" field, which includes a wide variety of user questions. These queries, ranging from specific requests like "Guide me to the gym" to more general ones such as "I am completely lost, how do I reach the parking lot?", are designed to replicate a wide range of real-world scenarios. The inclusion of such a varied set of queries is crucial for training the agent to manage numerous types of interactions within a virtual environment. A key feature of the dataset is that it contains queries about a variety of rooms and places, not limited to those in the VOXReality conference venue. This strategy is intended to provide the model with a level of generalizability, equipping it to handle a diverse range of queries that might arise in different virtual environments.

The second component, the "knowledge" field, consists of real navigation instructions directly sourced from Mozilla Hubs. This essential part of the dataset includes actual navigation data, formatted as they appear in Mozilla Hubs, as mentioned in the previous subsection. Incorporating these real navigation instructions into the dataset is crucial to ensure that the agent is fine-tuned with accurate and authentic navigation patterns for effective and precise guidance.

The third field is the model's output, which is the agent's navigational guidance in response to the user's query. This output is designed to be clear and concise, prioritizing straightforward navigation instructions and avoiding any unnecessary or confusing details. The primary aim here is to ensure that the agent's responses are user-friendly, helping users efficiently reach their intended destinations.

The dataset prepared for the navigation agent consists of 100 carefully selected samples for training purposes and an additional 40 samples specifically designated for testing model's performance. Each of these samples has undergone meticulous annotation by human experts, a process that is essential for the dataset's quality. The careful annotation process also serves to validate the relevance and applicability of each navigation instruction within the given "knowledge" field. The following examples showcases two examples from the dataset, providing a clear representation of how each sample is structured and annotated.

```
{
"input": "How do I reach the parking lot?,
"navigation_intructions": "move forward, move 10 meters, turn right, move 20 meters, turn left, move 30 meters, stairs down, turn right, move 10 meters",
"output":  "To find the parking lot:\n- Begin by moving forward 10 meters.\n- Turn right and continue 20 meters.\n- Then turn left and walk 30 meters.\n-
Head downstairs.\n- Lastly, turn right and go an additional 10 meters.\n\n This route will take you to the parking lot."
},
{
"input": Can you show the way to the exit?,
"navigation_intructions": "turn around, move 11 meters, turn left, move 5 meters, turn right, move 9 meters, turn left, move 4 meters",
"output":  "To find the way to the exit:\n- Initially, turn around.\n- Go straight for 11 meters.\n- Make a left and travel 5 meters.\n- Turn right and
proceed 9 meters.\n- Finally, turn left and move ahead 4 meters.\n\n You will be at the exit."
}
```

In summary, the well-organized structure and intensive annotation of the dataset are instrumental in the fine-tuning process of the navigation agent. The objective is to enhance the agent's ability to process diverse user inputs, combine them with structured navigation instructions, and generate clear and effective responses. Such fine-tuning process is necessary for improving user experience in virtual space navigation, ensuring the agent offers precise and user-friendly assistance.

Fine-tuning process

In the fine-tuning process of the navigation agent, which utilizes the Llama-2 model, the Low-Rank Adaptation (LoRA) [Hu et al., 2021] method was used. The choice of LoRA for fine-tuning was based on its effectiveness and efficiency in adapting large-scale neural networks like Llama-2 to specific tasks. LoRA is particularly suited for this purpose due to its core principle of matrix factorization and low-rank approximations. This approach allows for the simplification of complex, highly parameterized layers in the neural network into more manageable structures. Figure 3.23 demonstrates LoRA during and after training, showcasing how this approach simplifies complex, highly parameterized layers in the neural network into more manageable structures. Such simplification is crucial for isolating and modifying the most influential features of the model without the need for extensive retraining.

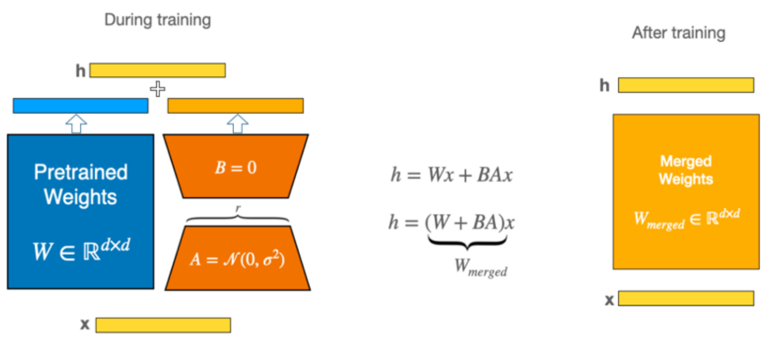


Figure 3.23. Illustration of LoRA during and after training

The use of the LoRA technique in refining the navigation agent is based on its effectiveness in tailoring neural networks for new challenges. LoRA accomplishes this by adding task-specific parameters but keeping the overall size manageable. This method is especially beneficial for the navigation agent because it allows for flexibility and efficiency in processing, which is vital for training substantial models like Llama-2 without excessive computational requirements.

After conducting experiments with various combinations of model parameters, the final selection for fine-tuning the model involved using 24% of all the model's parameters. This specific proportion of parameters was found to be the most effective in striking the right balance between model performance and computational efficiency. The fine-tuning process of the navigation agent yielded promising results (Figure 3.24). During the training epochs, both the training and validation loss showed a steady downward trend, signaling enhancements in the model's accuracy and its ability to generalize to new data. Specifically, the training loss exhibited a steep decrease, suggesting that the model was effectively learning the navigation tasks. Meanwhile, the validation loss also decreased, at a slower rate, which is a positive sign of the model's ability to adapt to new, unseen data. This reduction in loss across epochs demonstrates the effectiveness of the fine-tuning process, enabling the model to retain previously learned information while successfully acquiring new navigational knowledge.

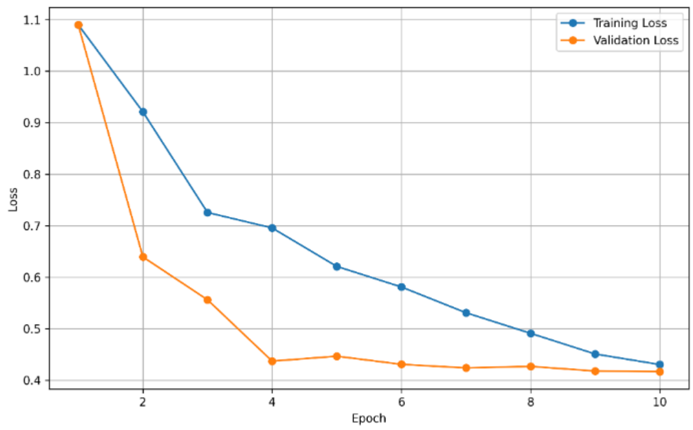


Figure 3.24. Training and validation loss per epoch

Evaluation Process

The evaluation of the navigation agent posed a complicate challenge, given the complex nature of the task. A comprehensive evaluation required careful consideration of two key aspects. Firstly, it was essential to confirm that the model accurately replicated each navigation instruction from Mozilla Hubs without omissions. This step is crucial to confirm the model's reliability in conveying complete navigational guidance. Secondly, the model's output needed to be compared with the ground truth in the test set to ascertain its accuracy. The proximity of the model's responses to this ground truth serves as a measure of its effectiveness in providing precise navigational directions.

So, the first evaluation process was to ensure that the model generated all the navigation instructions. To evaluate the performance of the model the semantic knowledge similarity (SKS) between the provided knowledge and the output of the model is compared. Specifically, the “all-MiniLM-L6-v2” sentence transformer was utilized. This model developed by the Sentence Transformers project [Reimers & Gurevych, 2019], which specializes in providing dense vector representations for sentences and paragraphs. These representations can be used in various NLP tasks such as clustering, semantic search, and information retrieval.

This model is based on the MiniLM architecture, which is known for its small size and efficient performance. The "L6" in the name indicates that the model has 6 layers, making it smaller and faster than larger models with similar capabilities. The model has been fine-tuned on a diverse and extensive dataset, consisting of over 1 billion sentence pairs, which contributes to its ability to understand and encode a wide range of sentences effectively. The model allows to encode batches of sentences, effectively converting textual information into numerical data that capture the semantic essence of the sentences.

To enhance the model's accuracy for the designated task, a fine-tuning process was implemented of the “all-MiniLM-L6-v2” model. This involved developing a specialized dataset comprised of three distinct sections: "anchor," "positive," and "negative." The "anchor" field contains the navigation instructions as issued by Mozilla Hubs. The "positive" field includes the correct rendition of the navigation instructions, ensuring completeness without omitting any details. Conversely, the "negative" field contains outputs that are similar with the "positive" instructions but with certain elements either missing or modified. For instance, in the example below, one "negative" field might involve misdirections such as “Turn left” with “Turn right” and changing the moving distance from “10 meters” to “8 meters.” Another example of the figure shows a complete omission of a step. The construction of this dataset improves the capability of the model to recognize the errors, such as missing or incorrect navigation instructions.

```
{
“anchor”: “move 8 meters, turn right, move 10 meters, turn around, move 15 meters”,
“positive”: “To get to the booth 2, follow these steps:\n- Move forward 8 meters.\n- Turn right and move 10 meters.\n- Turn around and proceed for 15 meters.\n\n You should now be in the booth 2.”,
“negative”:  “To get to the booth 2, follow these steps:\n- Move forward 8 meters.\n- Turn left and move 10 meters.\n- Turn around and proceed for 15 meters.\n\n You should now be in the booth 2.”
},
{
“anchor”: “turn around, move 15 meters, turn left, move 20 meters, turn right, move 10 meters”,
“positive”: “To reach the exit, follow these steps:\n- Start by turning around.\n- Move forward 15 meters.\n- Turn left and proceed for 20 meters.\n- Turn right and move 10 meters forward.\n\n You
should now be at the exit.”,
“negative”: “To reach the exit, follow these steps:\n- Start by turning around.\n- Move forward 15 meters.\n- Turn left and proceed for 20 meters.\n- Turn right.\n\n You should now be at the exit.
}
```

For the training and testing of this fine-tuning process, the dataset comprised 80 training samples and 30 test samples. This number of samples was chosen to provide a balance between comprehensive coverage of different scenarios and maintaining a manageable dataset size for efficient training and evaluation.

The effectiveness of the fine-tuning process was evaluated using three distinct metrics to measure accuracy, each providing a unique perspective on the model’s ability to generate semantically similar sentence embeddings. The first metric employed was the cosine distance, which calculates the cosine of the angle between two vectors in a multi-dimensional space. In addition to cosine distance, the Manhattan and Euclidean distances were utilized. The Manhattan distance measures similarity by summing the absolute differences between coordinates of the vectors. On the other hand, the Euclidean distance measures the direct straight-line distance between two points in Euclidean space, giving a sense of the "ordinary" distance one would intuitively consider between points. The purpose of these evaluations is to understand how well the model is performing in terms of sentence similarity using different measures of distance. The usage of all three metrics helps to get a comprehensive view of the model's performance, as each distance measure captures different aspects of similarity. The evaluation results on the test dataset are presented in Table 3.16.

Table 3.16. Evaluation of sentence-transformers fine-tuning process with sentence similarity

Cosine Similarity (%)	Manhattan Distance (%)	Euclidean Distance (%)
90	89.9	90

The evaluation of the navigation agent incorporated the fine-tuned sentence transformer model, specifically aimed at ensuring the agent's capability to generate all required navigation instructions comprehensively, without any omissions. For this purpose, the evaluation process involved a comparison of the provided navigation instructions with the outputs generated by the agent, followed by the calculation of their cosine similarity. Additionally, semantic textual similarity (STS) was evaluated using the original "all-MiniLM-L6-v2" model, by comparing the ground truth responses with those produced by the fine-tuned model. Moreover, the Rouge score was also employed as a metric in this evaluation process. The inclusion of the Rouge score provided an additional layer of assessment regarding the preciseness and completeness of the generated instructions. It helped in estimating how well the model's outputs aligned with the key information presented in the original navigation instructions.

To benchmark the effectiveness of the fine-tuning process, a comparative evaluation was conducted between the original Llama 2 model and the fine-tuned. This assessment aimed to highlight the enhancements achieved through the fine-tuning. The results of this comprehensive evaluation are presented in Table 3.17. The table offers a detailed comparison, showcasing the performance of both models across various metrics.

Table 3.17. Evaluation of navigation agent before and after fine-tuning

Phase	Rouge-1 (%)	Rouge-2 (%)	Rouge-L (%)	SKS (%)	STS (%)
Before	33.1	17.2	29.4	85.1	92.2
After	68.9	42.3	60.5	95.7	97.3

Evaluation of the Workflow before and after fine-tuning

Moving forward in the development process of our workflow with the fine-tuned Llama 2 model, we directed our focus towards a critical evaluation phase. This was particularly pertinent with the fine-tuning efforts that had been applied to the Llama2 model, as we aimed to comprehensively understand the influence of these modifications on the model's performance. It was imperative to understand the impact of the fine-tuning process on the model's performance in crucial workflow tasks. This phase of evaluation was especially focused on tasks such as intent and destination recognition.

To ensure the accurate classification of user intent by the model, an extensive evaluation process was implemented. This process began with the creation of an evaluation dataset, comprising 99 carefully selected samples, evenly distributed with 33 samples for each of the three categories. Once the evaluation dataset was established, each user query from the dataset was inputted into the model to generate responses. The effectiveness of the model was then determined by comparing these responses against the ground truth labels provided in the dataset. Key performance metrics such as accuracy, precision, recall, and F1 score were calculated to provide a comprehensive assessment of the model's performance.

To ensure a comprehensive assessment, the evaluation process was conducted on both the original Llama 2 model and its fine-tuned version. This was crucial to verify that the fine-tuning, while enhancing the model's performance for navigation instruction tasks, did not compromise its effectiveness in other areas. The goal was to maintain the model's versatility and ensure its proficiency across a broad spectrum of tasks, not just those it was specifically fine-tuned for.

In the evaluation of the Intent Recognition task, the results post-fine-tuning showed a decrease in key performance metrics compared to the pre-fine-tuning phase. Specifically, there was a reduction in accuracy, precision, recall, and F1 score. This decline suggests that while the fine-tuning process targeted certain aspects of the model's functionality, it may have had unintended effects on its ability to perform in the intent recognition task. It's important to recognize that such outcomes are part of the iterative process of model development and refinement. Future iterations and enhancements are expected to address these areas, aiming to strike a balance between specialized improvements and overall task proficiency. The results are presented in Table 3.18.

Table 3.18. Evaluation of intent recognition agent before and after fine-tuning

Phase	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Before	85.8	86.1	85.8	85.8
After	79.7	80.4	79.7	79.7

For the destination recognition task, a similar evaluation method to the one used for intent recognition was implemented. A dataset with 100 samples, evenly distributed across various destination categories, was used to assess the model's performance. Additional samples were included in the 'no location' category to test scenarios involving non-existent locations within the building, as previously mentioned, and instances where users request navigation guidance without specifying a destination. This evaluation process involved analyzing the model's responses to the dataset queries and comparing them with the correct labels. The key metrics accuracy, precision, recall, and F1 score were calculated again for the original Llama 2 model and the fine-tuned version, and the results are succinctly presented in Table 3.19.

Table 3.19. Evaluation of destination recognition agent before and after fine-tuning

Phase	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Before	98	98.1	98	97.8
After	97	97.1	97	96.8

In the case of the destination recognition task, the performance metrics post-fine-tuning remained robust, showing no significant drop in effectiveness. The accuracy, precision, recall, and F1 score maintained high levels, with only marginal variations observed. This consistency indicates that the fine-tuning process did not adversely affect the model's ability to perform in the destination recognition task. Such stability in performance is encouraging, as it demonstrates the model's resilience and effectiveness in maintaining high standards of task proficiency even after undergoing specific modifications.

Deployment

The conference assistant provides a REST API, developed using FastAPI, interface for interacting with a conference-related virtual assistant. This assistant is designed to handle user queries related to conference navigation, program information, and general inquiries. The API is structured around two main endpoints, each catering to different aspects of the assistant's functionalities. The detailed explanation of the API is presented in Appendix III.

The first endpoint is the intent destination endpoint (Figure 3.25), which processes user queries to determine the user's intent and the corresponding destination within the conference context. This function takes a string input representing the user query and returns a response identifying the intent and destination.

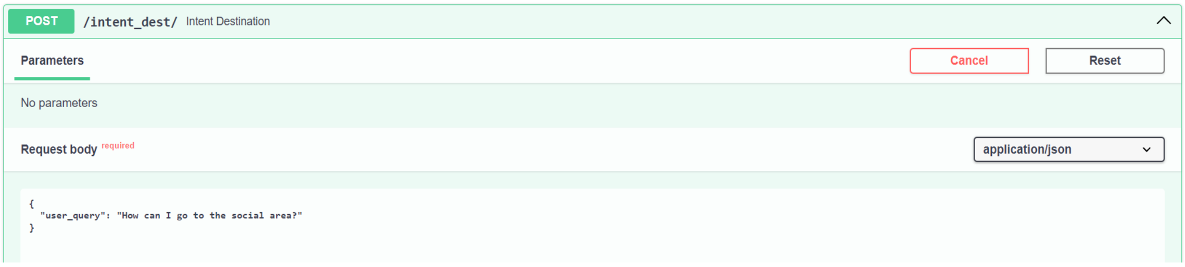


Figure 3.25. The intend and destination recognition endpoint

The second and the last endpoint is the response generation, which caters to generating responses based on user queries, and additional Mozilla inputs. It offers personalized assistance depending on whether the query is related to navigation, program information, or general conference queries. The Figure 3.26 demonstrates the graphical representation of the response generation endpoint in the FastAPI framework.

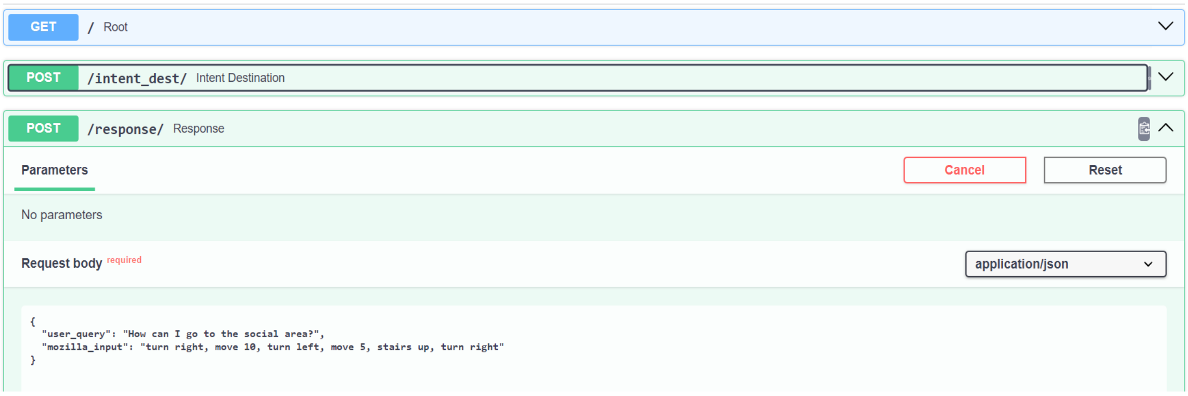


Figure 3.26. The response generation endpoint

Next Steps

As the next step, enhancing the conference agent workflow will involve expanding its capabilities to include providing insights about the trade show and offering spatial information to users. This enhancement is aimed at delivering a more comprehensive and user-friendly experience. Integrating the model with extensive details about the trade show, such as exhibitor lists, product details, event schedules, and special highlights, will guide the user through the trade show experience. This feature will be particularly beneficial for attendees seeking to maximize their time at the event. Additionally, incorporating spatial information is crucial for a more intuitive user experience. This will be performed with the integration with the visual language model. By leveraging this model, the agent can interpret and describe the spatial layout and visual aspects of the conference environment. Users will benefit from detailed visual descriptions, and insights about the spatial arrangement. To further enhance user satisfaction, improving the inference time of the model is also a priority.

3.4.2. AR Training Assistant

We first introduce the autonomous training agent architecture powered by large language models (LLMs) [Wang et al., 2023], aiming at a solution for training assistants in an Augmented Reality (AR) environment. The core idea is to adopt a new workflow (Figure 3.27) powered by LLMs and driven by four functional submodules: memory, tools, planning, and actions. Specifically, a LLM, the brain of the dialogue agent, can plan the next action based on memory such as historical conversations (short-term) and knowledge base (long-term memory) and reaction of the executable tools that links to use case. To implement this architecture, we adopt LangChain[10], the flexible abstractions and extensive toolkit for LLM applications.

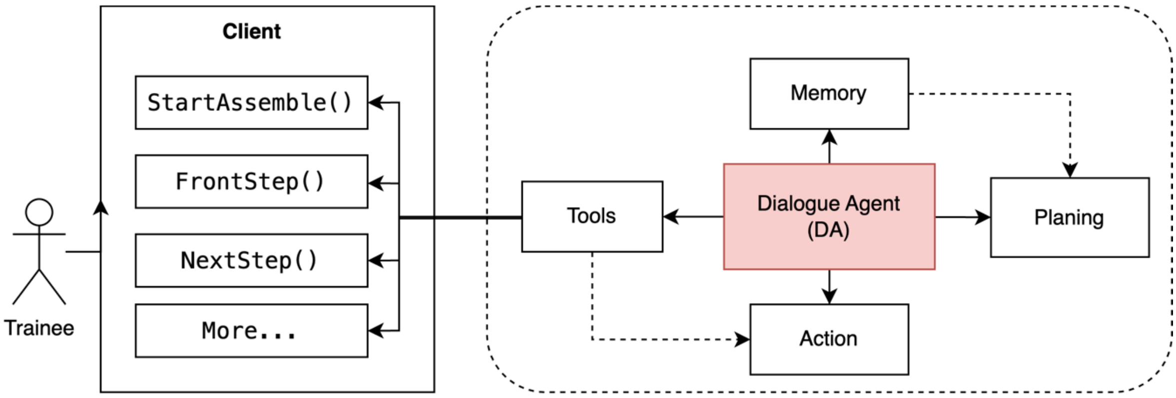


Figure 3.27. Autonomous agent powered by large language models

Workflow and Motivation

In this work, we aim to introduce a workflow (Figure 3.28): we use large language models (LLMs) as a human-like "brain" to augment XR systems with Artificial Intelligence; and vice versa, we enable XR systems to provide services by external API tools to augment general LLMs specifically for the AR Training Assistant (ARTA) use case, so that the AR system can better plan which tool to use and when to use it based on fully understanding the users' need.

The core idea of this workflow is the autonomous Dialogue Agent (DA), which can 1) utilize a language model such as Llama2 to conduct planning of a sequence of actions to take, 2) invoke a set of useful tools of real-world XR applications and observe the execution results and 3) make the decision of the next action based on all observations in the memory (e.g., historical utterances, response from XR environment, knowledge base, database, etc).

Particularly, the revolution of the autonomous agent powered by LLMs brings two benefits:

- Unlike general chains, where a sequence of actions is hardcoded (e.g., traditional dialogue pipeline with NLU followed NLG), the autonomous agents are powered by a language model and a prompt, which can serve as a reasoning engine to determine which actions to take and in which order automatically.
- The real-world applications can be learned to use automatically through reasoning, reducing the human's efforts on guidelines or defining training needs.

[10] https://python.langchain.com/docs/get_started/introduction

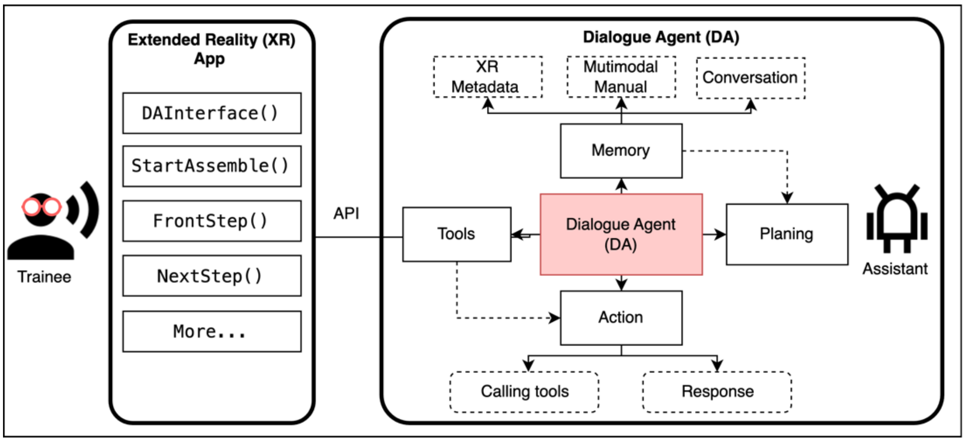


Figure 3.28. The workflow of ARTA

In this workflow, there are three key phrases to develop the model:

Phase 1: Develop the workflow and test the existing LLMs such as "gpt-3.5-turbo-16k-0613"[11]. It aims to investigate if this workflow based on the dialogue agent can integrate and immersivity interact with a XR application seamlessly.

Phase 2: Create domain-specific datasets for the use case. We created datasets by two ways: (1) we reconstruct a TEACH-EDH dataset from human Commander-Follower conversations for household tasks in the VR simulation environment; (2) We generate and collect a VOX-ARTA dataset for LEGO brick assembly tasks in the XR simulation environment, using the workflow with a GPT3.5 and multimodal LEGO manuals.

Phase 3: Finetuning open-resourced LLMs such as “Llama2-chat-7b”[12] with context-response pairs obtained from various datasets for teaching conversations (i.e., the reconstructed TEACH-EDH[13] dataset), tool-calling and multimodal conversations (i.e., VOX-ARTA-LEGO dataset[14]).

Model description

Following the workflow, we develop the model in terms of three aspects: 1) conversational agent walkthrough for planning and prompt engineering by fine-grained instructions 2) XR Application tools’ API integration and 3) core LLM upgrade by finetuning the prevailing open-resourced LLM on use-case specific datasets.

Conversation agent walkthrough

We adopt a conversation agent walkthrough that optimizes the standard ReAct [Yao et al., 2023] with prompt engineering for conversation. This walkthrough demonstrates how to use an agent plan for conversation. LM can obtain a set of observations from the environment and decide the next action by reasoning the trace. The observations are a chain of information from diverse sources, e.g., historical conversations and prompts, XR tool responses, etc. The

[11] https://platform.openai.com/docs/models/gpt-3-5
[12] https://huggingface.co/meta-llama/Llama-2-7b-chat
[13] https://huggingface.co/datasets/Jiahuan/teach_edh
[14] https://huggingface.co/datasets/Jiahuan/vox_arta_lego

final response is decided by observing the action of XR tools. Following is an example of one-loop chain-of-thought in the walkthrough:

Human: Can you show the detail of the roof?
AI: Action: ShowPieces
Action Input: roof
Response: Pieces available for roof: 2 x 2 red, 2 x 2 blue, 1 x 2 black, 1 x 2 white. Please select which piece you want to use.

In contrast to many other agent walkthroughs that primarily focus on optimizing tool usage for determining the best response, our approach diverges, recognizing the importance of fostering conversational capabilities. This distinction becomes evident when comparing it to the conventional ReAct (Reason and Act) walkthrough, with the key disparity lying in the nature of the prompt. Our emphasis is on cultivating a much more conversational prompt to enhance the agent's ability to engage in meaningful dialogue with users. Specifically, we integrate task-specific prompt and XR tool responses in the observations so as to enable the agent to consider these in the reasoning traces for the decision of a next action. The walkthrough procedure is illustrated in Figure 3.29.

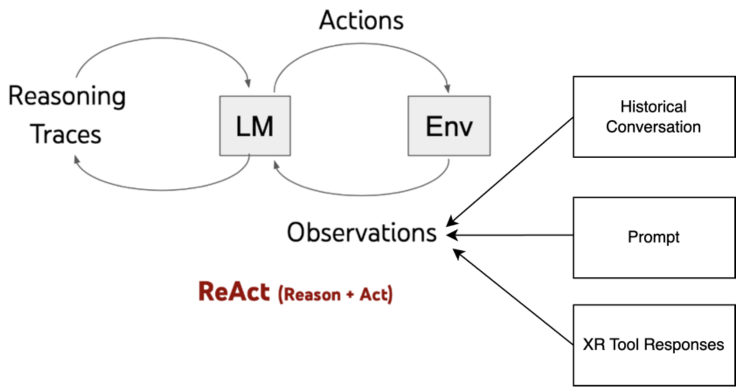


Figure 3.29. The illustration of conversational ReAct walkthrough

XR Application tools' API integration

We develop an XR test application and integrate the information interaction flow (Figure 3.30) between the application and the AI agent. This will be considered as a part of the whole workflow when the agent decides to call a specific tool for the user.

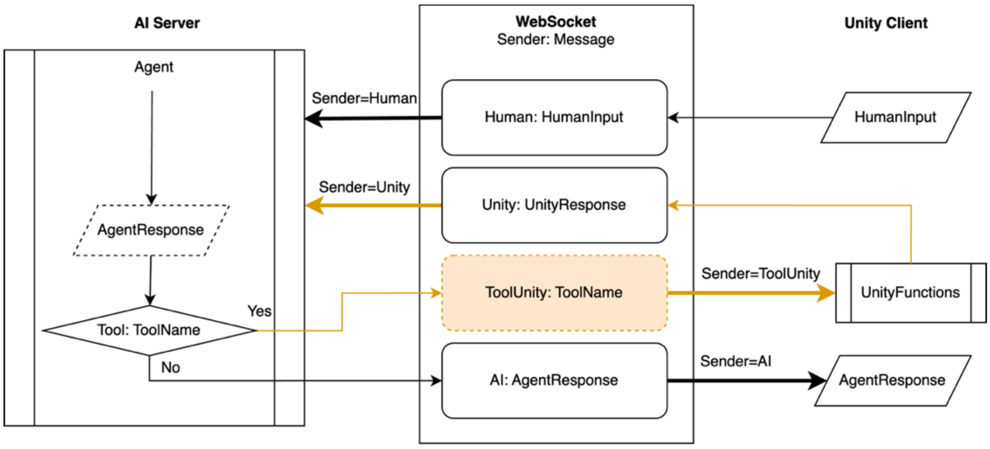


Figure 3.30. Interactive information flow between XR test application and AI agent

We develop a collection of executable interactive functions, presented in Table 3.20 within the XR test application, serving as external tools available for the agent. These tools play a crucial role in enabling agents to engage with the XR world and systematically navigate the XR environment. To facilitate the utilization of these functionalities, we have developed a toolkit—a set of API calls. This toolkit allows LLMs to access and invoke the services or functions outlined in the accompanying table, empowering them to strategize and determine the optimal for utilization within the XR system with minimum latency.

Table 3.20. Descriptions of the functions for the XR tools in the test application

No	Tool Name	Description
1	StartAssemble	Initiates the assembly process.
2	NextStep	Moves to the next assembly step.
3	FrontStep	Goes back to the previous assembly step.
4	Explode	Triggers an explosion for detailed viewing.
5	Recover	Restores the initial state of AR objects after an explosion.
6	FinishedVideo	Ends the assembly process and shows a video of the assembled LEGO bricks.
7	ReShow	Repeats the current assembly step.
8	Enlarge	Enlarges or zooms out the current object.
9	Shrink	Shrinks or zooms in the current object.
10	GoToStep	Goes to the given assembly step number.
11	Rotate	Rotates the current object to a direction.
12	ShowPieces	Shows all candidate LEGO pieces to be assembled.
13	HighlightCorrectComponents	Highlights correct attachment points and components.
14	GetCurrentStep	Gets the number of the current step.
15	GetRemainingStep	Gets the number of the remaining steps.
16	CheckStepStatusVR	Checks if the current step in Unity is accomplished correctly or not.
17	APICallObjectRecognitionAR	Calls the VLM agent to identify LEGO pieces based on the provided video streaming data from AR glasses and highlights the recognized pieces in the AR environment.
18	APICallCheckStepStatusAR	Calls the VLM agent to determine if the current assembly step is completed correctly or not, using the provided video streaming data from AR glasses as input.

Parameter-Efficient Fine-Tuning of Quantized LLMs

Fully finetuning an LLM requires powerful computation resources as a result of the recent growth of LLM sizes. For example, the smallest version of Llama2 contains 7 billion learnable parameters that require a GPU with ~28GB memory. To be able to utilize LLMs in the GPU we use to perform our developments (GeForce RTX 3090 with 24GB), we adopt 4-bit Quantized Low Raw Adaptor (QLoRA) [Dettmers et al, 2022] [Dettmers et al, 2023]. We use this approach for finetuning the quantization of Llama2-7b-chat and conduct parameter-efficient fine-tuning (PEFT) of the quantized model with Low Raw Adaptor (LoRA). The comparison of the graphical representation of full fine-tuning, LoRA and QLoRA is presented in Figure 3.31.

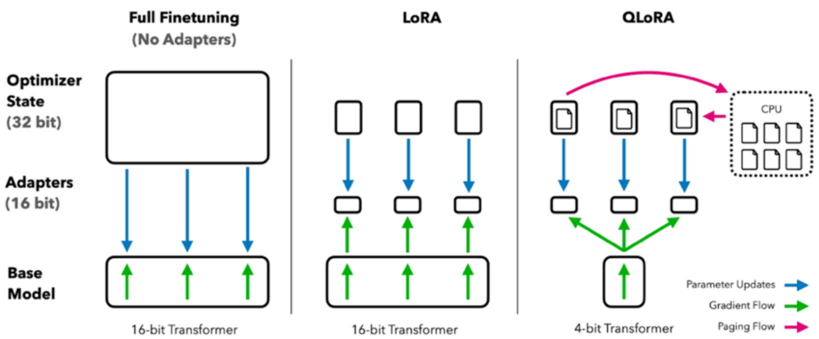


Figure 3.31. Comparison of full fine-tuning, LoRA, and QLoRA

Dataset Construction and Statistics

In VOXReality, we produced two datasets that can be used for teaching assistant conversation agents, TEACH-EDH and VOX-ARTA-LEGO.

TEACH-EDH

We reconstruct the TEACH-EDH (Task-driven Embodied Agents that Chat - Execution from Dialog History) dataset by curating conversations between human Commanders and human Followers (Drivers) who collaboratively use natural language to accomplish household tasks within the AI2-THOR simulation environment [Kolve et al., 2017]. Each data sample consists of all historical utterances as an “input” and the current utterance as an “output”. We construct 14,740 context-response pairs for TEACH-EDH dataset, including 9740 training, 1140 validation and 3860 test data. We represent examples of the dataset in Figure 3.32.

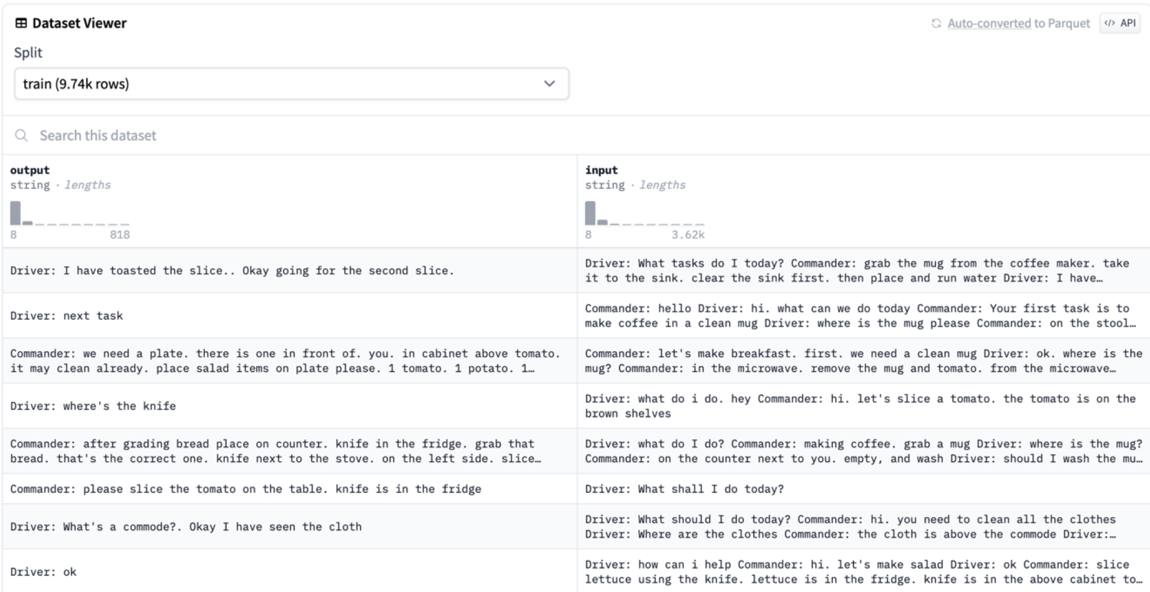
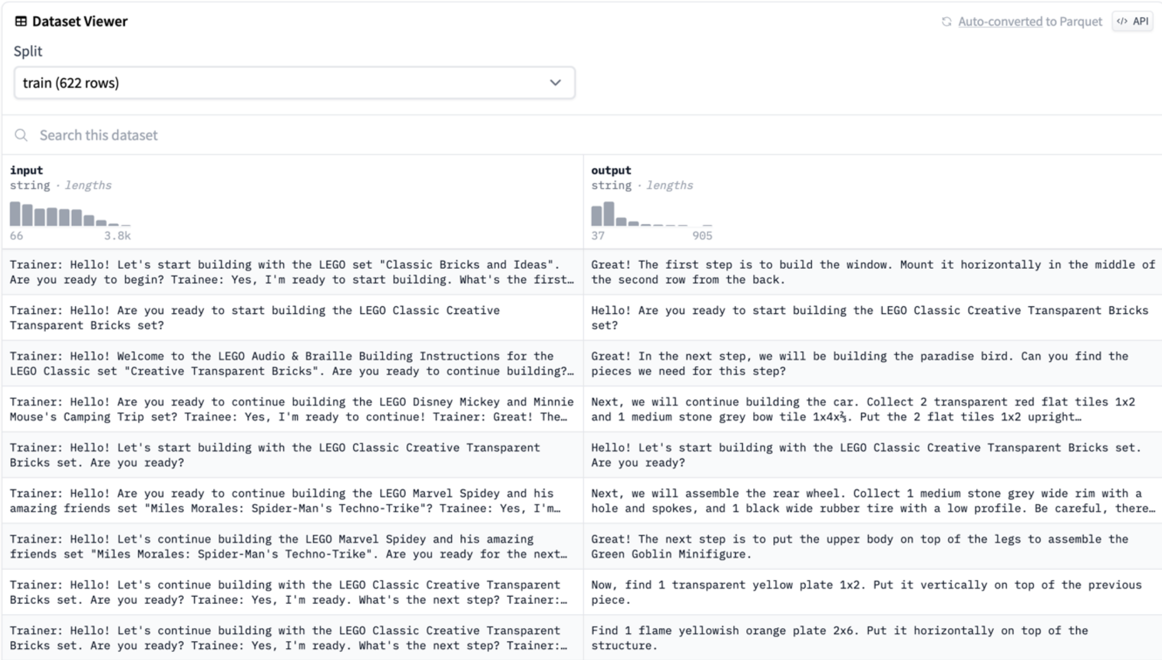


Figure 3.32. Examples of TEACH-EDH dataset

VOX-ARTA-LEGO

We generate and collect conversations between a user (trainee) and an AI teaching assistant (AI trainer) following the proposed workflow, which is powered by a commercially available LLM using GPT-3.5-turbo-16k-0613 and multimodal manuals for LEGO brick assembly. Specifically, we first performed web crawling in 65 multimodal manuals that are openly available on LEGO’s official website. Every manual consists of a summary and multiple assembly steps with both text instructions and images. This provides the fine-grained alignment between vision and language that are important to teach users how to assemble a LEGO brick set with a step-by-step guide. Then, we design prompts using the manuals and XR tools to generate textual simulated conversations. Similar to TEACH-EDH, each data sample consists of all historical utterances as an “input” and the current utterance as an “output”. We construct 866 context-response pairs for VOX-ARTA-LEGO dataset, including 622 training, 70 validation and 174 test data samples. We present examples of the dataset in Figure 3.33.



Dataset Viewer

Auto-converted to Parquet API

Split

train (622 rows)

Search this dataset

input string · lengths (66 – 3.8k)	output string · lengths (37 – 905)
Trainer: Hello! Let's start building with the LEGO set "Classic Bricks and Ideas". Are you ready to begin? Trainee: Yes, I'm ready to start building. What's the first…	Great! The first step is to build the window. Mount it horizontally in the middle of the second row from the back.
Trainer: Hello! Are you ready to start building the LEGO Classic Creative Transparent Bricks set?	Hello! Are you ready to start building the LEGO Classic Creative Transparent Bricks set?
Trainer: Hello! Welcome to the LEGO Audio & Braille Building Instructions for the LEGO Classic set "Creative Transparent Bricks". Are you ready to continue building?…	Great! In the next step, we will be building the paradise bird. Can you find the pieces we need for this step?
Trainer: Hello! Are you ready to continue building the LEGO Disney Mickey and Minnie Mouse's Camping Trip set? Trainee: Yes, I'm ready to continue! Trainer: Great! The…	Next, we will continue building the car. Collect 2 transparent red flat tiles 1x2 and 1 medium stone grey bow tile 1x4x⅔. Put the 2 flat tiles 1x2 upright…
Trainer: Hello! Let's start building with the LEGO Classic Creative Transparent Bricks set. Are you ready?	Hello! Let's start building with the LEGO Classic Creative Transparent Bricks set. Are you ready?
Trainer: Hello! Are you ready to continue building the LEGO Marvel Spidey and his amazing friends set "Miles Morales: Spider-Man's Techno-Trike"? Trainee: Yes, I'm…	Next, we will assemble the rear wheel. Collect 1 medium stone grey wide rim with a hole and spokes, and 1 black wide rubber tire with a low profile. Be careful, there…
Trainer: Hello! Let's continue building the LEGO Marvel Spidey and his amazing friends set "Miles Morales: Spider-Man's Techno-Trike". Are you ready for the next…	Great! The next step is to put the upper body on top of the legs to assemble the Green Goblin Minifigure.
Trainer: Hello! Let's continue building with the LEGO Classic Creative Transparent Bricks set. Are you ready? Trainee: Yes, I'm ready. What's the next step? Trainer:…	Now, find 1 transparent yellow plate 1x2. Put it vertically on top of the previous piece.
Trainer: Hello! Let's continue building with the LEGO Classic Creative Transparent Bricks set. Are you ready? Trainee: Yes, I'm ready. What's the next step? Trainer:…	Find 1 flame yellowish orange plate 2x6. Put it horizontally on top of the structure.

Figure 3.33. Examples of VOX-ARTA-LEGO dataset

Analysis of the datasets

We conduct a comparative analysis of the statistics, as presented in Table 3.21, and delve into the specifics of the two proposed datasets. Regarding the statistical aspects, the TEACH-EDH dataset boasts a larger number of data samples, featuring a higher count of dialogues and utterances. Conversely, the VOX-ARTA-LEGO dataset exhibits longer utterances from both users and systems. From a linguistic point of view, the utterances within the VOX-ARTA-LEGO dataset resemble a more human-like language, characterised by rich and diverse descriptions. On the other hand, the TEACH-EDH dataset leans more towards a machine-like language expression.

Table 3.21. Statistics of TEACH-EDH and VOX-ARTA-LEGO datasets

Phase	TEACH-EDH	VOX-ARTA-LEGO
Before	98	98.1
Context-Response	14,740	866
Dialogue	3320	171
Utterance	45,000	1,670
Avg Context Length	49.81	1,302
Avg Response Length	10.90	184
Avg Utterance Per Dialogue	13.67	9.77

Experimental Setup

Task definition

In the context of generative dialogue agents, the task is to generate responses to the user (trainee) given a sequence of diverse dialogue contexts, including historical utterances from trainer and trainee, multimodal manual, and XR metadata such as hand gesture input and head pose. In general, this task can be seen as a context-response generation task.

Evaluation metrics

We adopt the following metrics to evaluate models on both TEACH-EDH and VOX-ARTA-LEGO datasets.

- Perplexity is a commonly used measure of how well a language model predicts a sample. Lower perplexity indicates a better model, as it measures the average uncertainty of the model in predicting the next word in a sequence.
- ROUGE (described in Section 2.5) is a set of metrics used for the automatic evaluation of generated response by comparing it to a set of reference responses.
- METEOR (described in Section 2.5) incorporates various linguistic and semantic features to assess the quality of generated responses compared with reference responses.

Implementation details

The base LLM we use is the open-resourced Llama-2-7b-chat-hf[15], which is a collection of pretrained and finetuned generative text models ranging in scale from 7 billion and up. We use 4-bit quantization of the model and conduct PEFT with LoRA to obtain finetuning of 4.43% of original trainable parameters. The batch size is 8 and the maximum step is 2,000 with early stopping when the validation loss does not decrease with continuous 10 steps. The maximum number of tokens is 512.

Results

We evaluate the Llama2 and its finetuned models on two proposed datasets and represent both quantitative results and qualitative analysis, respectively.

Quantitative Results

We present the evaluation results for the two proposed datasets both before and after finetuning with QLoRA, as outlined in Table 3.22. Perplexity, ROUGE, and METOR metrics were employed for assessment on the test set of both TEACH-EDH and VOX-ARTA-LEGO dataset. A significant reduction in perplexity is observed for both datasets, plummeting from 42.18 to 3.08 for the TEACH-EDH dataset and from 8.81 to 1.06 for the VOX-ARTA-LEGO dataset. This indicates an enhanced model performance in predicting the next token with reduced uncertainty. Lower perplexity corresponds to a heightened proficiency in next token prediction. However, the ROUGE and METOR metrics show a decline on both datasets following QLoRA finetuning. Specifically, ROUGE experiences a 3.42% drop in TEACH-EDH dataset and 5.63% drop in VOX-ARTA-LEGO dataset, indicating a reduction in the overlap of 1-gram tokens between generated and reference responses. METEOR exhibits a slight drop in both the TEACH-EDH dataset (2.01%) and the VOX-ARTA-LEGO dataset (1.09%),

[15] meta-llama/Llama-2-7b-chat-hf

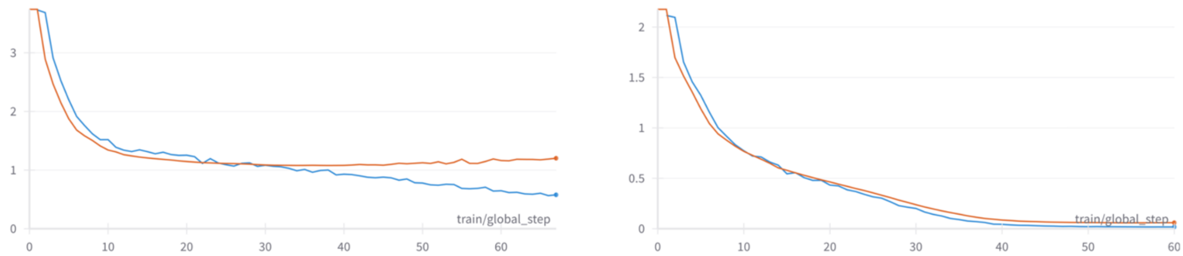
suggesting a decline in linguistic and semantic similarity between generated and reference responses. This decrease can be attributed to two potential factors: firstly, QLoRA finetunes only 4.43% of the original trainable parameters, necessitating a substantial volume of data to effectively influence the limited parameter adjustments. Secondly, the quality of fine-grained, human-like data is crucial; otherwise, the introduction of new data may introduce noise to the original model .

Table 3.22. Evaluation results on TEACH-EDH and VOX-ARTA-LEGO dataset

Model	TEACH-EDH			VOX-ARTA-LEGO		
	Perplexity	ROUGE	METEOR	Perplexity	ROUGE	METEOR
Llama2	42.18	8.17	12.08	8.81	23.36	12.42
Llama2+QLoRA	3.08	4.74	10.07	1.06	17.73	11.37

Qualitative Analysis

We scrutinize the learning curves depicting the loss evolution for both training and validation datasets throughout the finetuning process, illustrated in Figure 3.34, where training loss is coloured blue and validation loss is coloured red. In the case of the TEACH-EDH dataset, the training loss decreases from 3.732 to 1.202, while the validation loss decreases from 3.742 to 1.188. The model employs early stopping, interrupting the training at the 67th step in the 2nd epoch, as the validation loss shows no continuous improvement over 10 consecutive steps. In the case of the VOX-ARTA-LEGO dataset, the training loss exhibits a notable reduction, decreasing 2.116 to 0.177 while the validation loss decreases from 2. 176 to 0.060. The model adopts early stopping at 60th step in the 25th epoch.



a b

Figure 3.34. The learning curve of Llama-2-7b-chat-hf model finetuning on a) TEACH-EDH dataset and b) VOX-ARTA-LEGO

Case Study

We discuss the following showcases to analyse the responses generated by the model and provide an example of a walkthrough to illustrate how the reasoning of traces works with observations of actions. The following are two examples of the inference results from the test set, where the agent can observe actions, action inputs, and historical utterances from both human and assistant. In comparison with the reference responses, the model tends to generate longer responses with richer context and instructions. This motivates us to conduct a human evaluation to reconsider whether the decrease in ROUGE and METEOR scores is a fair metric for evaluating generation performance.

Observations #1
Action: Guide
Action Input: Yes
Response: Sure thing! To find another transparent bright orange roof tile 1x2, 45°, I suggest checking the nearby shelves or drawers. Once you locate it, put it horizontally on the left knob, overhanging to the left and sloping to the left. Could you please let me know once this item is found?
Human (Trainee): No, I followed the instructions and everything looks good so far. What's the next step?
Assistant: Great! In that case, the next step is to attach the control levers to the sides of the vehicle. Can you please find two transparent bright orange roof tiles 1x2, 45°? Place them horizontally on the left and right knobs respectively, overhanging to the left and sloping to the left. Once you have located these tiles, please let me know so I can assist you further.

Prediction #1
Great! In that case, the next step is to attach the control levers to the sides of the vehicle. Can you please find two transparent bright orange roof tiles 1x2, 45°? Place them horizontally on the left and right knobs respectively, overhanging to the left and sloping to the left. Once you have located these tiles, please let me know so I can assist you further.

Reference #1
That completes this step! You've successfully built the control levers and added the plates. Is there anything you're unsure about or any questions you have?

Observations #2
Action: Guide
Action Input: Yes
Human: Can you find a bright red plate 1x2 with a single stud on top?
Response: Of course! To find a bright red plate 1x2 with a single stud on top, I suggest checking the nearby shelves or drawers. Once located, please place it horizontally on the table. Could you please let me know once this item is found?

Prediction #2
Of course! To find a bright red plate 1x2 with a single stud on top, I suggest checking the nearby shelves or drawers. Once located, please place it horizontally on the table. Could you please let me know once this item is found?

Reference #2
Great! Let's proceed to the next step.

The following is an example instance of an interactive conversation session with actual human input. The current agent demonstrates an ability to comprehend the logic behind tool invocation and provides sensible responses in alignment with the overall context of the conversation.

Human: Hi can you start assemble the LEGO car?
Assistant: Sure thing! I'd love to help you with that. Which type of LEGO car would you like to build? We have different options available. Once you've made your choice, I can guide you through each step of the assembly process. Let me know what you prefer!

Human: OK, got it. Can you go to next step?
Assistant: Great! Let's proceed to the next step. Initiating the assembly process...
Action: StartAssemble()
Action Input: None
Response: Assembly process started. Please follow the instructions carefully. Please provide the actual instruction manual text for the LEGO 11001 Classic set so I can generate the appropriate responses.

Human: Can you go to step 3?
Action: NextStep()
Action Input: None
Response: Moving to step 3...

Human: Then what should I do next step?
Action: NextStep()
Action Input: None
Response: Moving to step 4...

Human: Can you tell me something not related to LEGO assembly task?
Assistant: Sorry, I cannot fulfill that request. My primary function is to assist users with their LEGO assembly tasks. If you have any questions or requests related to the task, please feel free to ask. Otherwise, I'll be here to help whenever you need it. How may I assist you today?

Human: Can you show the detail of the roof?
Action: ShowPieces("roof")
Action Input: None
Response: Pieces available for roof: 2 x 2 red, 2 x 2 blue, 1 x 2 black, 1 x 2 white. Please select which piece you want to use.

Deployment

We deploy our model through the WebSocket API (an example run can be seen in Figure 3.35), encapsulated within a Docker Image, to facilitate interactive communication between the user (trainee) and the assistant (AI trainer). We also prepared a deployment using FastAPI, a robust framework that enables seamless and efficient handling of the communication interface. The details can be found in Appendix IV.

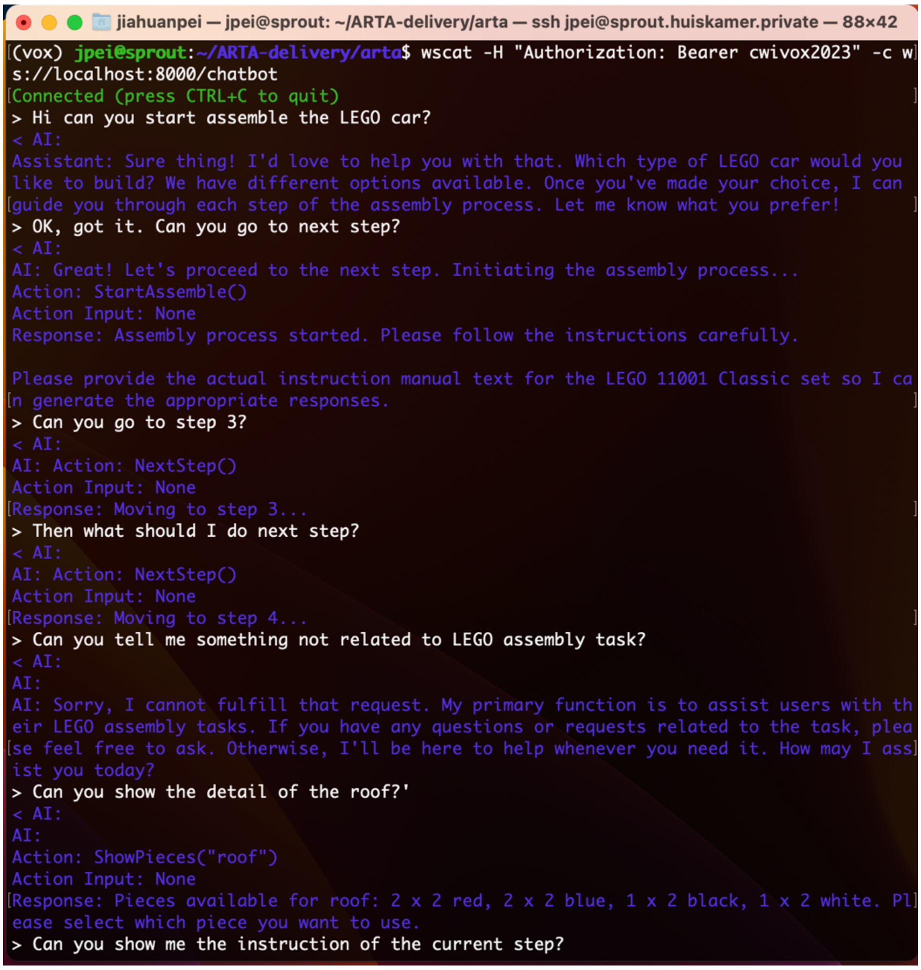


Figure 3.35. Interactive WebSocket API

Next Steps

In the future work, we plan to extend the current VOX-ARTA-LEGO dataset to a larger version for finetuning and evaluation. The number of data samples is around 10 times of the current small version. Besides, we will consider integrating VOXReality vision-language models to enhance the vision-related observations in the conversational ReAct walkthrough in order to facilitate cases where such information would be useful. This enhanced pipeline will be made openly available for the third parties in the Open Calls. Furthermore, we plan to conduct human evaluation of the generated responses in order to further evaluate the responses generated in terms of natural language generation.

4. Conclusions

The main objective of VOXReality WP3 is to implement natural language processing models that are context-aware, multilingual, visually grounded and knowledgeable of the domain specific needs of applications in XR environments. To this end, activities in WP3 invest in providing NLP models for automatic speech recognition, machine translation, vision language, and conversation agents.

This deliverable presents the methodology and the results obtained from the experiments of VOXReality natural language processing models that are developed in the first 15 months of the project timeline. These models include the automatic speech recognition, context-aware machine translation, robust machine translation, simultaneous machine translation, image captioning, visual question answering, spatial scene description, VR conference conversation agent and AR training assistant tasks. This deliverable also describes the APIs developed for the inference capabilities of these models to be utilized in a plug and play approach.

The initial results present that the VOXReality models achieve better or competitive results in their respective tasks compared to pre-trained models that do not involve the added values of the proposed models. The experiments prove that the natural language processing models developed in the project timeline until month 15 show promise and the further research that is planned to be conducted in the second phase of the WP3 tasks will evolve the models to be better suited to their respective tasks. The initial deployment tests that will be presented in a future deliverable (D4.1) also shows promising results of their applicability in XR applications.

The consortium will focus on the deployment of the models in the VOXReality usecase applications in the coming months. The deployment results and optimization results for different deployment options will be presented in "D4.1.1 - Model Deployment analysis" on M17. Furthermore, the models and their performances will evolve throughout the project and additional models will be provided using the models produced from the research that will be performed in the second phase of the WP3 timeline. These changes, additions, and the final results will be presented in the second version of this document "D3.1.2 - Advanced AI multi-model for XR analysis" on Month 30.

References

[Amodie et al, 2016] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z. (2016, June). Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning (pp. 173-182). PMLR.

[Anastasopoulos et al. 2019] Anastasopoulos, Antonios & Lui, Alison & Nguyen, Toan & Chiang, David. (2019). Neural Machine Translation of Text from Non-Native Speakers. 3070-3080. 10.18653/v1/N19-1311.

[Ardila et al., 2019] Ardila, Rosana & Branson, Megan & Davis, Kelly & Henretty, Michael & Kohler, Michael & Meyer, Josh & Morais, Reuben & Saunders, Lindsay & Tyers, Francis & Weber, Gregor. (2019). Common Voice: A Massively-Multilingual Speech Corpus.

[Baevski et al., 2020] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in Neural Information Processing Systems, 33, 12449-12460.

[Banerjee et al., 2005] Banerjee, S., & Lavie, A. (2005, June). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72).

[Bao et al., 2021] Bao, H., Dong, L., Piao, S., & Wei, F. (2021). Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254.

[Bapna et al., 2019] Bapna, Ankur & Arivazhagan, Naveen & Firat, Orhan. (2019). Simple, Scalable Adaptation for Neural Machine Translation.

[Bar-Hillel, 1951] Bar-Hillel, Y. (1951). Fully Automatic High-Quality Translation of Indo-European Languages.

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

[Bryant et al., 2019] Bryant, Christopher & Felice, Mariano & Andersen, Øistein & Briscoe, Ted. (2019). The BEA-2019 Shared Task on Grammatical Error Correction. 52-75. 10.18653/v1/W19-4406.

[Budzianowski et al., 2018] Budzianowski, P., Wen, T. H., Tseng, B. H., Casanueva, I., Ultes, S., Ramadan, O., & Gašić, M. (2018). Multiwoz--a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. arXiv preprint arXiv:1810.00278.

[Cettolo et al., 2015] Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Cattoni, R., & Federico, M. (2015). The IWSLT 2015 evaluation campaign. In Proceedings of the 12th International Conference on Spoken Language Translation.

[Cettolo et al., 2017] Cettolo, M., Federico, M., Bentivogli, L., Niehues, J., Stüker, S., Sudoh, K., ... & Federmann, C. (2017). Overview of the IWSLT 2017 Evaluation Campaign. In Proceedings of the 14th International Conference on Spoken Language Translation (pp. 2-14).

[Chane et al., 2020] Chen, Y. C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., ... & Liu, J. (2020, August). Uniter: Universal image-text representation learning. In European conference on computer vision (pp. 104-120). Cham: Springer International Publishing.

[Conneau et al., 2020] Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition. arXiv preprint arXiv:2006.13979.

[Costa-jussà et al., 2022] Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., ... & NLLB Team. (2022). No language left behind: Scaling human-centered machine translation. arXiv preprint arXiv:2207.04672.

[Dettmers et al, 2022] Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). Llm. int8 (): 8-bit matrix multiplication for transformers at scale. arXiv preprint arXiv:2208.07339.

[Dettmers et al, 2023] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. arXiv preprint arXiv:2305.14314.

[Devlin et al., 2018] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[Elbayad et al., 2020] Elbayad, M., Besacier, L., & Verbeek, J. (2020). Efficient wait-k models for simultaneous machine translation. arXiv preprint arXiv:2005.08595.

[Fan et al., 2021] Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., ... & Joulin, A. (2021). Beyond english-centric multilingual machine translation. The Journal of Machine Learning Research, 22(1), 4839-4886.

[Hannun et al, 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

[Heffernan et al., 2022] Heffernan, K., Çelebi, O., & Schwenk, H. (2022). Bitext mining using distilled sentence representations for low-resource languages. arXiv preprint arXiv:2205.12654.

[Honnibal et al., 2020] Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M. & Gelly, S.. (2019). Parameter-Efficient Transfer Learning for NLP. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research 97:2790-2799

[Hsu et al., 2021] Hsu, W. N., Bolte, B., Tsai, Y. H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29, 3451-3460.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

[Huo et al., 2020] Huo, J., Herold, C., Gao, Y., Dahlmann, L., Khadivi, S., & Ney, H. (2020). Diving deep into context-aware neural machine translation. arXiv preprint arXiv:2010.09482.

[Jelinek, 1976] Jelinek, F. (1976). Continuous speech recognition by statistical methods. Proceedings of the IEEE, 64(4), 532-556.

[Jon et al., 2021] Jon, J., Aires, J. P., Varis, D., & Bojar, O. (2021, August). End-to-End Lexically Constrained Machine Translation for Morphologically Rich Languages. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 4019-4033)

[Kochen, 1980] Kochen, M. (1981). On the generality of PARRY, Colby's paranoia model. Behavioral and Brain Sciences, 4(4), 540-541. https://doi.org/10.1017/s0140525x00000091

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., ... & Herbst, E. (2007, June). Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions (pp. 177-180). Association for Computational Linguistics.

[Kolve et al., 2017] Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., ... & Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474.

[Kritharoula et al., 2023] Kritharoula, A., Lymperaiou, M., & Stamou, G. (2023). Language Models as Knowledge Bases for Visual Word Sense Disambiguation. arXiv preprint arXiv:2310.01960.

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

[Li et al., 2019] Li, L. H., Yatskar, M., Yin, D., Hsieh, C. J., & Chang, K. W. (2019). Visualbert: A simple and performant baseline for vision and language. arXiv preprint arXiv:1908.03557.

[Li et al., 2021] Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D., Zhang, C., ... & Wei, F. (2021). Trocr: transformer-based optical character recognition with pre-trained models. arXiv 2021. arXiv preprint arXiv:2109.10282.

[Lillicrap et al., 2017] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[Lin et al., 2004] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74-81).

[Lison et al. 2018] Lison, P., Tiedemann, J., & Kouylekov, M. (2018, May). OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[Liu et al., 2021] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 10012-10022).

[Liu et al., 2022] Liu, F., Emerson, G., & Collier, N. (2023). Visual spatial reasoning. Transactions of the Association for Computational Linguistics, 11, 635-651.

[Lopes et al., 2020] Lopes, A., Farajian, M. A., Bawden, R., Zhang, M., & Martins, A. F. (2020, November). Document-level Neural MT: A Systematic Comparison. In Proceedings of the 22nd Annual Conference of the European Association for Machine Translation (pp. 225-234).

[Loshchilov & Hutter, 2017] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[Ma et al., 2018] Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., ... & Wang, H. (2018). STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. arXiv preprint arXiv:1810.08398.

[Ma et al., 2019] Ma, X., Pino, J., Cross, J., Puzon, L., & Gu, J. (2019). Monotonic multihead attention. arXiv preprint arXiv:1909.12406.

[Miyato et al., 2016] Miyato, T., Dai, A. M., & Goodfellow, I. (2016). Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725.

[Müller et al., 2018] Müller, M., Gonzales, A. R., Voita, E., & Sennrich, R. (2018, October). A Large-Scale Test Set for the Evaluation of Context-Aware Pronoun Translation in Neural Machine Translation. In Proceedings of the Third Conference on Machine Translation: Research Papers (pp. 61-72).

[Napoles et al. 2017] Napoles, Courtney & Sakaguchi, Keisuke & Tetreault, Joel. (2017). JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction.

[Nawrot et al., 2021] Nawrot, P., Tworkowski, S., Tyrolski, M., Kaiser, Ł., Wu, Y., Szegedy, C., & Michalewski, H. (2021). Hierarchical transformers are more efficient language models. arXiv preprint arXiv:2110.13711.

[Padmakumar et al., 2022] Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., ... & Hakkani-Tur, D. (2022, June). Teach: Task-driven embodied agents that chat. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 2, pp. 2017-2025).

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

[Peng et al., 2022] Peng, B., Galley, M., He, P., Brockett, C., Liden, L., Nouri, E., ... & Gao, J. (2022). Godel: Large-scale pre-training for goal-directed dialog. arXiv preprint arXiv:2206.11309.

[Pratap et al., 2023] Pratap, Vineel & Tjandra, Andros & Shi, Bowen & Tomasello, Paden & Arun Babu, Thirunavukkarasu & Kundu, Sayani & Elkahky, Ali & Ni, Zhaoheng & Vyas, Apoorv & Fazel-Zarandi, Maryam & Baevski, Alexei & Adi, Yossi & Zhang, Xiaohui & Hsu, Wei-Ning & Conneau, Alexis & Auli, Michael. (2023). Scaling Speech Technology to 1,000+ Languages.

[Rabiner, 1989] Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77(2), 257-286.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[Radford et al., 2023] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023, July). Robust speech recognition via large-scale weak supervision. In International Conference on Machine Learning (pp. 28492-28518). PMLR.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140), 1-67.

[Ranftl et al., 2021] Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision transformers for dense prediction. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 12179-12188).

[Rebuffi et al., 2017] Rebuffi, Sylvestre-Alvise & Bilen, Hakan & Vedaldi, Andrea. (2017). Learning multiple visual domains with residual adapters.

[Reimers & Gurevych, 2019] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[Subramanian et al., 2020] Subramanian, S., Collobert, R., Ranzato, M. A., & Boureau, Y. L. (2020). Multi-scale Transformer Language Models. arXiv preprint arXiv:2005.00581.

[Sun et al., 2020] Sun, Z., Wang, M., Zhou, H., Zhao, C., Huang, S., Chen, J., & Li, L. (2020). Rethinking document-level neural machine translation. arXiv preprint arXiv:2010.08961.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

[Tajiri et al., 2012] Tajiri, Toshikazu & Komachi, Mamoru & Matsumoto, Yuji. (2012). Tense and Aspect Error Correction for ESL Learners Using Global Context. 50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference. 2.

[Tan & Bansal, 2019] Tan, H., & Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. arXiv preprint arXiv:1908.07490.

[Tiedemann & Thottingal, 2020] Tiedemann, J., & Thottingal, S. (2020). OPUS-MT – Building open translation services for the World. European Association for Machine Translation Conferences/Workshops.

[Touvron et al., 2020] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2020). Training data-efficient image transformers & distillation through attention. arXiv 2020. arXiv preprint arXiv:2012.12877.

[Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

[Wang et al., 2023] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. R. (2023). A survey on large language model based autonomous agents. arXiv preprint arXiv:2308.11432.

[Xie et al., 2020] Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. Advances in neural information processing systems, 33, 6256-6268.

[Yannakoudakis et al., 2011] Yannakoudakis, Helen & Briscoe, Ted & Medlock, Ben. (2011). A New Dataset and Method for Automatically Grading ESOL Texts.. 180-189.

[Yao et al., 2023] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. ICLR 2023.

[Zhang & Feng, 2021] Zhang, S., & Feng, Y. (2021). Universal simultaneous machine translation with mixture-of-experts wait-k policy. arXiv preprint arXiv:2109.05238.

[Zheng et al., 2020] Zheng, B., Liu, K., Zheng, R., Ma, M., Liu, H., & Huang, L. (2020). Simultaneous translation policies: From fixed to adaptive. arXiv preprint arXiv:2004.13169.

[Zhu et al., 2022] Zhu, Q., Geishauser, C., Lin, H. C., van Niekerk, C., Peng, B., Zhang, Z., ... & Huang, M. (2022). ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format. arXiv preprint arXiv:2211.17148.

[Zhuang et al., 2019] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... & He, Q. (2019). A comprehensive survey on transfer learning. arxiv e-prints. arXiv preprint arXiv:1911.02685.

Appendix I: Endpoints for the ASR and NMT components

1) Transcribe Audio: The method to transcribe given audio files. The source language is not required.
URL: *<api_url>/transcribe_audio_files*
Method: *POST*
Request parameters:

- audio_files [array]: the audio file to transcribe

Response:

- **Success:** *Status Code:* 200 *Content:* {"transcriptions": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

2) Translate Audio: The method to translate given audio files to the target language. The source language is not required.
URL: *<api_url>/translate_audio_files*
Method: *POST*
Request parameters:

- audio_files [array]: the audio file to translate
- target_language [String]: “en”-English, “el”-Greek, “it”-Italian, “nl”-Dutch, “de”-German, “es”-Spanish
- return_transcription [bool]: true/false

Response:

- **Success:** *Status Code:* 200 *Content:* {"transcriptions": [String], "translations": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

3) Translate Audio with Context: The method to translate given audio files to the target language using additional textual context for better translation of the transcribed text. The source language is not required.
URL: *<api_url>contextual_translate_audio_files*
Method: *POST*
Request parameters:

- audio_files [array]: the audio file to translate
- context [String]: the textual context to use in the translation
- target_language [String]: “en”-English, “el”-Greek, “it”-Italian, “nl”-Dutch, “de”-German, “es”-Spanish
- return_transcription [bool]: true/false

Response:

- **Success:** *Status Code:* 200 *Content:* {"transcriptions": [String], "translations": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

4) Translate Text: The method to translate given text to the target language. The source language is not required.
URL: *<api_url>/translate_text*
Method: *POST*
Request parameters:

- text [String]: the text to translate
- target_language [String]: “en”-English, “el”-Greek, “it”-Italian, “nl”-Dutch, “de”-German, “es”-Spanish

Response:

- **Success:** *Status Code:* 200 *Content:* {"translations": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

5) Translate Text with Context: The method to translate given text to the target language using additional textual context for better translation. The source language is not required.
URL: *<api_url>/contextual_translate_text*
Method: *POST*
Request parameters:

- text [String]: the text to translate
- context [String]: the textual context to use in the translation
- target_language [String]: “en”-English, “el”-Greek, “it”-Italian, “nl”-Dutch, “de”-German, “es”-Spanish

Response:

- **Success:** *Status Code:* 200 *Content:* {"translations": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

6) Translate Text with Context and Terminology: The method to translate given text to the target language using additional textual context and terminological term pairs for better translation. The source language is not required.
URL: *<api_url>/contextual_terminology_translate_text*
Method: *POST*
Request parameters:

- text [String]: the text to translate
- context [String]: the textual context to use in the translation
- target_language [String]: “en”-English, “el”-Greek, “it”-Italian, “nl”-Dutch, “de”-German, “es”-Spanish

Response:

- **Success:** *Status Code:* 200 *Content:* {"translations": [String]}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

7) Upload Terminology: The method to translate given text to the target language. The source language is not required.
URL: *<api_url>/upload_terminology*
Method: *POST*
Request parameters:

- file [String]: the file that contains terminological term pairs for different languages

Response:

- **Success:** *Status Code:* 200 *Content:* {"Successfully uploaded terminology file: <file_name>"}
- **Error:** *Status Code:* 466 *Content:* {"message": "An error occurred in <error_location>. Details: <error_type>-<error_details>"}

Appendix II: Endpoints for the VL components

1) Scenegraph Object Detection: The endpoint takes as input an image and returns a list with all the detected objects of the image.
URL: *<api_url>*/scenegraph_list
Method: *POST*
Request parameters:

- rgb [String]: an RGB input image in JPEG or PNG format

Response:

- **Success:** *Status Code:* 200 *Content:* {"Item list": [String]}
- **Error:** *Status Code:* 422 *Content:* {"message": "Input not supported for prediction"}

2) Scenegraph Spatial Descriptions: The endpoint takes as input an image and returns a list with phrases describing the spatial relationships between all objects that were detected on the image.
URL: *<api_url>*/scenegraph_rgb
Method: *POST*
Request parameters:

- rgb [String]: an RGB input image in JPEG or PNG format

Response:

- **Success:** *Status Code:* 200 *Content:* {"Descriptions": [String]}
- **Error:** *Status Code:* 422 *Content:* {"message": "Input not supported for prediction"}

3) Scenegraph Spatial Descriptions with Depth: The endpoint takes as input an RGB and a depth image and returns a list with phrases describing the spatial relationships between all objects that were detected on the image.
URL: *<api_url>*/scenegraph_rgb_depth
Method: *POST*
Request parameters:

- rgb [String]: an RGB input image in JPEG or PNG format
- depth [String]: a depth input image in PNG format

Response:

- **Success:** *Status Code:* 200 *Content:* {"Descriptions": [String]}
- **Error:** *Status Code:* 422 *Content:* {"message": "Input not supported for prediction"}

4) SOTA Image Captioning: The endpoint takes as input an RGB image and returns the description for this image.
URL: *<api_url>/cap_gpt2*
Method: *POST*
Request parameters:

- rgb [String]: an RGB input image in JPEG or PNG format

Response:

- **Success:** *Status Code:* 200 *Content:* {“Image caption”: [String]}
- **Error:** *Status Code:* 422 *Content:* {"message": "Input not supported for prediction”}

5) SOTA Question Answering: The endpoint takes as input an RGB image and a question related to this image and returns the answer.

URL: *<api_url>/lxmert*

Method: *POST*

Request parameters:

- rgb [String]: an RGB input image in JPEG or PNG format
- question [String]: a question related to the given image

Response:

- **Success:** *Status Code:* 200 *Content:* {“Prediction by LXMERT”: [String]}
- **Error:** Status Code: 422 Content*:* {"message": "Input not supported for prediction”}

Appendix III: Endpoints for the navigation assistant

1) Intent and Destination Detection: This endpoint processes a user query to identify the intent and the destination (topic or area of interest) within the context of the conference.
URL: *<api_url> /intent_dest/*
Method: *POST*
Request parameters:

- user_query [String]: The user's query or question.

Response:

- **Success:** *Status Code:* 200 *Content:* {"intent": [String], "destination": [String]}
- **Error:** *Status Code: 400 Content: {*"message": *"Invalid request: 'user_query' is required."}*

2) Generate Response: This endpoint generates a response for the user's query using the provided user query and additional knowledge input (Mozilla input).

URL: *<api_url> /response/*
Method: *POST*
Request parameters:

- user_query [String]: The user's query or question.
- mozzila_input [String]: Additional knowledge or context to assist in generating the response.

Response:

- ***Success:*** Status Code: 200 Content*: {"response": [String]}*
- **Error:** Status Code: 400 Content: *{*"message": *"Invalid request format. Please check user_query, intent and mozilla_input fields."}*

Appendix IV: Endpoints for the training assistant

1) Textual Chat Agent: This endpoint generates a response for the user's query using the provided user query and XR test application functional tools.

URL: *<api_url> /ask/*

Method: *POST*

Request parameters:

- query [String]: The user's query or question.

Response:

- ***Success:*** Status Code: 200 Content*: {"response": [String]}*
- **Error:** Status Code: 400 Content: *{*"message": *"Invalid request format."}*

2) Unity Chat Agent: This endpoint generates a response for the user's query using the provided user query and XR test application functional tools using requests coming from Unity

URL: *<api_url> /chatbot/*

Method: *Websocket*

Request parameters:

- websocket [WebSocket]: The websocket from Unity with user's query or question as messages.

Response:

- ***Success:*** Status Code: 200 Content*: {"response": [String]}*
- **Error:** Status Code: 400 Content: *{*"message": *"Invalid request format."}*

VOXReality

Voice driven
interaction in XR spaces

Funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Directorate-General for Communications Networks, Content and Technology (DG CNECT). Neither the European Union nor the granting authority can be held responsible for them.