Multilingual Voice Command Understanding: A Bilingual Joint SLU Approach

Abstract

We introduce a bilingual Spoken Language Understanding (SLU) system for smart-home commands in English and Urdu, which maps speech inputs to intent-slot representations. The proposed pipeline integrates OpenAI Whisper for speech recognition, optional T5-based paraphrase augmentation with similarity filtering, and Joint SLU model for joint intent detection and BIO-based slot filling with the help of XLM-RoBERTa model which toxenize input into vectors. A knowledge injection layer further normalises slot values using multilingual alias dictionaries and fuzzy matching. Training is conducted on parallel English-Urdu examples, which are synced using an identifier that ensures both sentences convey the same meaning. Slot filling is mainly learned using English text, as it provides clearer supervision, while a cross-lingual intent consistency term helps the model to produce similar predictions for both English and Urdu inputs. There is also an optional code-mixed extension that gives synthetically generated mixed-language sentences, which provides an added oversight, thereby improving the model's robustness when handling combinational languages.

1 Introduction

Spoken language understanding (SLU) is a core component of voice-driven Assistants where the system infers a compact semantic representation, which consists of an intent label (what the user wants to do) and slot spans (who/what/where/when) from noisy spoken input. In smart homes, this problem is further complicated when there is a mix of languages used, the user is saying English and Urdu in the same sentence.

In such settings, two recurring failure modes are particularly critical. First, automatic speech recognition (ASR) may get transcription errors that modify the surface form of the input while preserving its underlying intent, causing downstream intent interpretation from textual input. Second, even when transcriptions are accurate, users may express identical functional intents using diverse linguistic forms; consequently, a narrowly trained text classifier can become brittle unless there is sufficient variability in phrasing during training.

Previous discussions of robustness often treat "ASR noise" and "linguistic paraphrase variability as separate engineering concerns, but work on Multilingual text encoders have shown that a shared representation helps in handling multiple languages in a single model, which is attractive for bilingual SLU because it reduces the overhead of managing distinct pipelines per language.

We describe a complete, modular cascaded architecture, where responsibilities are decomposed across stages instead of being handled by a single end-to-end speech model: (i) Whisper-based ASR makes multiple hypotheses; (ii) a T5 paraphrase generator expands hypotheses under a semantic similarity filter; (iii) Joint SLU model performs joint intent detection and BIO-based slot filling with the help of XLM-RoBERTa; and (iv) a knowledge-injection layer maps noisy slot mentions to canonical smart-home entities using multilingual alias tables and fuzzy matching. Training uses parallel English–Urdu JSONL records linked by a shared identifier. The baseline objective supervises slot filling mainly using the English representation, while enforcing cross-lingual consistency between intent predictions derived from English and Urdu encodings. An optional code-mixed extension adds additional supervision on synthetic mixed utterances constructed from aligned bracketed annotations. The training notebooks configure a single-GPU workflow that overlaps slower CPU work with accelerator execution: PyTorch DataLoaders use worker processes, pinned host memory, and prefetch queues so batches are ready while the GPU runs the previous step; tensors are moved to the device with non-blocking copies where ap-

propriate; automatic mixed precision (autocast plus GradScaler) is enabled on CUDA; cuDNN benchmark mode and TF32 matmul flags are turned on for throughput; and epoch-level metrics can be computed in a background thread, so the next epoch can start without waiting for CPU-only scoring to finish.

2 Background

Spoken-language assistants usually rely on a step-by-step process where speech recognition (ASR) comes first, followed by understanding language (NLU). A lot of research is focused on how errors in speech recognition affect steps like understanding what the text means and classifying it. Kubis et al. (Kubis et al., 2023) tried to figure out how speech recognition errors can be simulated; they did this by converting text to speech, then back to text using speech recognition, which made the text noisy. This way, they could test how well natural language understanding works when the text is not perfect. Their findings suggest that using noise can help test how strong natural language understanding is. This is important because it can be used along with tests on audio to make sure the understanding works well. We have also used this approach, but with some differences in how it's implemented, using Whisper N-best lists and paraphrasing. The main idea is that understanding language must be tested with different types of transcription variability.

Evaluation practice is now going beyond word error rate (WER) as the sole proxy for spoken-language quality. Wan et al. (Wan et al., 2025) have created a way to test voice understanding in big language models. This new way, called SpeechIQ (Wan et al., 2025), checks levels of understanding. It looks as if a machine can just repeat what it hears and understand a situation or make sense of what's being said in a real-life way. It was found that sometimes a machine can get a lot of words right but still not understand what's being said; similarly, it can understand what's being said but gets a lot of words wrong. Also, some systems that use both speech recognition and text work better than systems that try to do everything once. Sometimes these machines even make up answers when they're not sure. Our smart home system focuses on understanding what people want to do. It checks what a person wants and what details they give. It looks at all the speech recognition results and paraphrases. Then it uses a majority vote to decide what the person wants to do. This way, if the machine gets a word wrong, it does not determine the final command interpretation (Wan et al., 2025).

When we train SLU models using the mistakes made by one speech recogniser, they can become too good at handling those specific mistakes and not work well with a different speech recogniser. Jung et al. (Jung et al., 2024) found a way to make the training better by interventional speech noise injection (ISNI), which adds some noise to the speech that's similar to what a speech recogniser would produce. This helps the SLU models learn to understand language in a way that is not affected by the specific speech recogniser being used. The goal of this is to make sure that the SLU models can still understand what people are saying if the speech recognizer changes. The way it is done is by looking at the connection between the speech recognizer and the natural language understanding (NLU) part of the system. It is trying to make the system work better by using versions of what the user said and by adding more information to the system so it can understand what the user is talking about. This includes using different names for the same device or room so the system can understand what the user means, even if they say it in a way that sounds similar to something else. This way, the user does not have to switch to a speech recognizer. SLU models and speech recognizers are important for this to work (Jung et al., 2024).

Paraphrase generation to help train models has been around before the recent large-model tooling. Cho et al. (Cho et al., 2019) describe Para-SSL, where a generator comes up with many different ways to say something for semi-supervised NLU. This helps developers cover ground with limited examples. The gains they reported show that having words is a big deal for making assistants robust. In this code, a T5 paraphraser that uses similarity helps with this at inference time. The training also exposes the encoder to different language surfaces. The design is modular, so this paraphrasing part can be changed or removed without affecting the XLM-RoBERTa part (Cho et al., 2019).

Across (Kubis et al., 2023)–(Cho et al., 2019), English-centric benchmarks and methods remain common. Urdu smart-home commands, code-mixed phrasing, and paired bilingual annotations are still comparatively scarce in the public literature tied to those lines of work. The gap is both in

terms of language coverage and how the systems are built. Usually, paraphrase robustness, noise in speech recognition, and entity normalisation are optimised separately. This implementation tries to address these issues. It does this by (i) training XLM-RoBERTa on both English and Urdu data so they share the representations using paraphrase expansion and speech recognition hypotheses together before scoring, (ii) combining Whisper hypotheses with paraphrase expansion before SLU scoring, (iii) resolving slots using knowledge injection, and (iv) optionally fine-tuning, with a code-mixed loss. These design choices make the system bilingual. Focused on semantic structure rather than exact wording.

3 Proposed Model

This study adopts a progressive cascaded design instead of a single end-to-end speech-to-semantics network. The proposed model is made up of parts that work together. The model uses (i) Automatic Speech Recognition to get text from speech, (ii) optional paraphrase expansion to come up with different ways of saying the same thing, (iii) a multilingual encoder-only Transformer to figure out what the user wants to do and what they are talking about and (iv) finally a deterministic knowledge layer to match what the user said to the devices or rooms in their smart home. The full data flow is illustrated in Fig. 1.

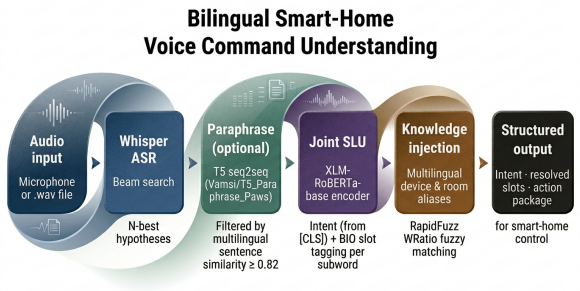

Figure 1: Cascaded bilingual SLU pipeline: Whisper hypotheses, optional T5 paraphrase filter, XLM-RoBERTa joint intent and BIO slot heads, and multilingual knowledge injection.

Input: The model takes raw audio from a WAV file or a microphone and text in both English and Urdu. The domain is smart-home style commands (devices, locations, timers, media).

Speech Recognition: Uses OpenAI Whisper to turn speech into text. It can give possible transcripts and rank them based on how likely they are to be correct.

Paraphrase Expansion: Model can also come up with ways of saying the same thing using a T5 conditional generator (Vamsi/T5_Paraphrase_Paws). It filters out with a multilingual sentence encoder (paraphrase-multilingual-MiniLM-L12-v2) so that only strings above a cosine similarity threshold (0.82) to the source are kept, reducing off-topic rephrasings while increasing lexical coverage.

Joint SLU Encoder: XLMRobertaTokenizerFast maps each surface string to subwords; XLM-RoBERTa-base (hidden size 768) produces contextual embeddings. Dropout (0.1) is applied before a linear intent head that reads the [CLS] vector and before a linear slot head that labels every subword position. Word-level BIO supervision is aligned to the first subword of each word; other subwords and special positions are masked with -100 in the slot loss. For a paired example, English and Urdu strings share one intent label. The training-time objective has the form:

$$\mathcal{L} = \mathcal{L}_{\text{int}}^{\text{EN}} + \mathcal{L}_{\text{slot}}^{\text{EN}} + \alpha \cdot \text{KL}(p_{\text{int}}^{\text{EN}} \parallel p_{\text{int}}^{\text{UR}})$$

where $\mathcal{L}_{\text{int}}$ and $\mathcal{L}_{\text{slot}}$ are cross-entropy terms and the Kullback-Leibler term pulls Urdu-conditioned intent probabilities toward English-conditioned ones.

Knowledge Injection: Takes the text and matches it to the actual devices or rooms in the smart home. It uses a registry of aliases to do this.

Code-Mixed Branch: The same encoder architecture can be trained with an additional mixed-language surface per example which adds an extra term to the objective function to handle the mixed-language text.

4 Experimental Setup

The experimental pipeline has two supervised tracks: bilingual joint training and optional code-mixed fine-tuning, and a demonstration track that exercises ASR, paraphrasing, and checkpoints on sample audio.

Datasets and Splits: Parallel smart-home style sentences are stored in data/en-US.jsonl and data/ur-PK.jsonl. These sentences are lined up with each other using an id field. Each sentence has partition tags (train/dev/test). Intent and BIO slot vocabularies are built from the training and development partitions only, then frozen for evaluation. We also have another way to split the sentences

that helps us test how well our system can handle sentences that're a mix of different languages, in data/mix-test.jsonl.

Bilingual Training: The JointSLUModel is initialized from xlm-roberta-base with intent and slot head dimensionalities set by the vocabulary files. Optimization uses AdamW with weight decay 0.01, learning rate 2×10^{-5}, batch size 32 for training (64 for validation), maximum sequence length 128, dropout 0.1 on head inputs, and consistency weight $\alpha = 0.3$. Training runs for 10 epochs on CUDA when available, with automatic mixed precision (autocast and GradScaler), cuDNN benchmark mode, TF32 matmul flags, DataLoader worker processes (num_workers = 2), pinned memory, prefetch_factor = 2, persistent workers, non_blocking host-to-device copies, and epoch metrics optionally computed in a background thread pool so validation work can overlap scheduling. The checkpoint with the highest validation macro slot F1 is saved together with label maps.

Code-Mixed Fine-Tuning: Training continues from a warm-started bilingual encoder using CodeMixedPairedSLUDataset and compute_loss_with_mix. Hyperparameters in the shipped run use 3 epochs, batch size 16, gradient accumulation 2 (effective batch 32), learning rate 1×10^{-5}, $\alpha = 0.3$, $\beta = 0.5$, mix_en_prob = 0.5, mix sample ratio 1.0, max length 128, and the same AdamW weight decay and throughput-oriented CUDA settings as above. The best code-mixed weights are saved.

ASR and Paraphrase Stacks: Loads Whisper (default base), the paraphrase generator, and either checkpoint. For each utterance, every ASR hypothesis and accepted paraphrase is scored by the frozen SLU model; intent decisions use majority voting over candidates to reduce sensitivity to a single noisy transcript.

Metrics: During training, we monitor validation intent accuracy and macro-averaged slot F1 (with zero_division handling for rare tags). After training, the bilingual checkpoint is scored on the English and Urdu test partitions in the notebook, reporting accuracy, precision, recall, and F1 separately for intent classification and for slot token prediction. The code-mixed notebook reports analogous validation curves and can score English, Urdu, and mixed views on the test loader.

Hardware and Software: The training notebooks implement a parallel and distributed-style workflow that overlaps CPU and accelerator execution. PyTorch DataLoaders use worker processes, pinned memory, and prefetching to prepare batches while computation runs. Non-blocking transfers, mixed precision (autocast + GradScaler), and cuDNN/TF32 optimisations improve throughput, while background metric computation enables asynchronous execution. Core tools include Hugging Face Transformers, OpenAI Whisper, Sentence-Transformers, and RapidFuzz.

5 Results

Validation intent accuracy improves from 0.5568 in epoch 1 to 0.8918 by epoch 10, while macro slot F1 rises from 0.0310 to 0.5023 over the same schedule. Checkpoints are written whenever the validation macro slot F1 improves; the final epoch in the stored run ends at 0.8918 intent accuracy and 0.5023 slot F1 on the validation split.

Table 1 summarises the final validation metrics for the bilingual model versus the code-mixed fine-tuning run.

Model	Val intent acc	Val slot F1
Bilingual	0.8918	0.5023
Code-mixed	0.8942	0.5360

Table 1: Validation metrics for bilingual and code-mixed fine-tuning

Table 2 presents the detailed test set performance metrics for the bilingual checkpoint.

Macro slot F1 therefore, remains the harder metric despite high token accuracy, reflecting class imbalance across BIO labels. The performance gap between English and Urdu is expected due to the higher quality of English annotations and the training bias toward English slot supervision.

Figure 2 shows the validation intent accuracy and slot F1 scores over the training epochs for the English-Urdu bilingual training.

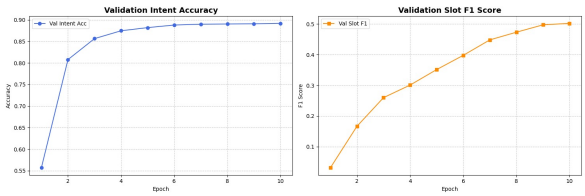

Figure 2: Validation Intent accuracy and validation slot f1 score on training data (eng-urdu)

Figure 3 illustrates the fine-tuning training

Intent Classification		
Metric	**English**	**Urdu**
Accuracy	0.8907	0.8420
Precision	0.8685	0.7916
Recall	0.8666	0.7806
F1-Score	0.8567	0.7803

Slot Tagging		
Metric	**English**	**Urdu**
Accuracy	0.9261	0.8684
Precision	0.5217	0.3792
Recall	0.4719	0.2569
F1-Score	0.4751	0.2814

Table 2: Model performance comparison (English vs Urdu) for bilingual checkpoint

curves on code-mixed data.

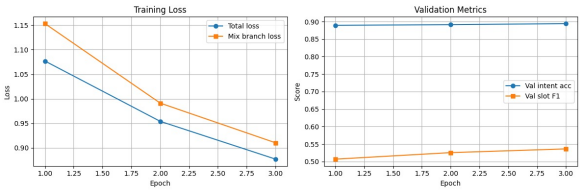

Figure 3: Fine tune training on code-mix data

6 Discussion

The proposed system follows a cascaded (modular) design, prioritizing flexibility and robustness over strict end-to-end optimization. In this setup, errors introduced by OpenAI Whisper can be partially handled through multiple hypotheses (N-best decoding) and paraphrase expansion. This allows the SLU encoder to focus more on capturing semantic meaning rather than dealing with raw audio imperfections. A cross-lingual consistency objective further encourages the model to produce aligned intent predictions for both English and Urdu inputs, improving the stability of the shared representation. However, slot learning still relies more heavily on English annotations due to their higher quality, introducing a controlled bias in training.

In terms of performance, intent detection achieves strong results, while slot filling remains comparatively weaker. This gap is common in joint models, especially when dealing with many rare slot types. Token-level accuracy can appear artificially high because of frequent "O" (non-slot) labels, whereas macro F1 highlights the model's difficulty in handling less frequent categories. The knowledge-injection layer enhances real-world usability by mapping noisy or misspelled inputs to canonical device names. However, its effectiveness is limited to predefined alias lists, and unseen or out-of-vocabulary entities may still fail, particularly under severe ASR errors.

Compared to prior work that focuses only on ASR robustness or only on paraphrase augmentation, this system combines both strategies at inference time and integrates them with bilingual training. This makes it more comprehensive and better suited for handling variability in real-world inputs. However, evaluation is conducted on text derived from ASR outputs rather than directly on audio-based reasoning tasks. While this aligns well with the smart-home SLU setup, it does not fully capture scenario-level understanding from raw speech.

Finally, the system incurs a relatively high computational cost due to the use of OpenAI Whisper, T5-based paraphrase generation, and the deep XLM-R encoder. The use of multiple hypotheses and majority voting further increases computation by requiring repeated forward passes. For practical deployment, optimisations such as reducing the number of paraphrases, using smaller ASR models, or applying model distillation techniques would be important next steps.

7 Conclusion

We described and experimentally instantiated a bilingual English–Urdu smart-home SLU stack that chains Whisper, optional T5 paraphrasing with similarity control, XLM-RoBERTa-based joint intent and BIO slot prediction, and multilingual knowledge injection. Bilingual training reaches 0.89 validation intent accuracy and 0.50 macro slot F1 after 10 epochs; a short code-mixed fine-tuning stage lifts validation slot F1 to about 0.54 while keeping intent accuracy comparable. Test-set intent F1 reaches 0.86 on English text and 0.78 on Urdu text processed by the released checkpoint, with slot F1 near 0.48 (English) and 0.28 (Urdu) under macro token scoring. Future work includes richer entity registries, tighter integration of ASR confidence with voting weights, and evaluation on larger spontaneous bilingual speech beyond the bundled sample audio.

Limitations

The system's slot filling performance remains weaker than intent detection, particularly for rare BIO tags and for Urdu language inputs. The knowledge injection layer is limited to predefined alias lists and may fail on unseen entities. Evaluation is conducted on text derived from ASR outputs rather than directly on raw audio-based reasoning tasks. The cascaded design incurs relatively high computational cost due to multiple hypotheses, paraphrase generation, and deep encoder forward passes.

References

M. Kubis, P. Skórzewski, M. Sowański, and T. Ziętkiewicz. Back Transcription as a Method for Evaluating Robustness of Natural Language Understanding Models to Speech Recognition Errors. In *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, pages 11824–11835, 2023. https://arxiv.org/pdf/2310.16609

Z. Wan et al. SpeechIQ: Speech Intelligence Quotient Across Cognitive Levels in Voice Understanding Large Language Models. In *Proc. 63rd Annu. Meeting Assoc. Comput. Linguist. (ACL)*, volume 1, pages 30381–30398, 2025. https://aclanthology.org/2025.acl-long.1466.pdf

Y. Jung, J. Lee, S. Choi, D. Lee, M. Kim, and S.-w. Hwang. Interventional Speech Noise Injection for ASR Generalizable Spoken Language Understanding. *arXiv:2410.15609v1*, October 2024. https://arxiv.org/pdf/2410.15609

Eunah Cho, He Xie, and William M. Campbell. Paraphrase Generation for Semi-Supervised Learning in NLU. In *Proc. Workshop on Methods for Optimizing and Evaluating Neural Language Generation (NeuralGen), ACL*, pages 45–54, 2019. https://aclanthology.org/W19-2306.pdf

Rune Birkmose, Nathan Mørkeberg Reece, Esben Hofstedt Norvin, Johannes Bjerva, and Mike Zhang. On-Device LLMs for Home Assistant: Dual Role in Intent Detection and Response Generation. In *Proc. Workshop on Noisy User-generated Text (WNUT)*, 2025. https://aclanthology.org/2025.wnut-1.7/

Fabian David Schmidt, Ivan Vulić, Goran Glavaš, and David Ifeoluwa Adelani. Fleurs-SLU: A Massively Multilingual Benchmark for Spoken Language Understanding. *arXiv:2501.06117*, January 2025. https://arxiv.org/abs/2501.06117

Zexun Wang, Yuquan Le, Yi Zhu, Yuming Zhao, Mingchao Feng, Meng Chen, and Xiaodong He. Building Robust Spoken Language Understanding by Cross Attention Between Phoneme Sequence and ASR Hypothesis. In *Proc. Interspeech*, 2024. https://arxiv.org/pdf/2203.12067

Bo Zheng, Zhouyang Li, Fuxuan Wei, Qiguang Chen, Libo Qin, and Wanxiang Che. HIT-SCIR at MMNLU-22: Consistency Regularization for Multilingual Spoken Language Understanding. In *Proc. Workshop on Multi-modal and Multi-lingual NLU (MMNLU)*, pages 15–24, 2022. https://aclanthology.org/2022.mmnlu-1.4.pdf

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. MASSIVE: A 1M-Example Multilingual Natural Language Understanding Dataset with 51 Typologically Diverse Languages. *arXiv:2204.08582*, 2022. https://arxiv.org/pdf/2204.08582