Bi-Directional, Emotionally Intelligent Sign-Speech Interpreter

Explanation of the Problem:

Despite advancements in accessibility, there remains a significant communication barrier between the hearing-impaired community and individuals without hearing difficulties. Traditional sign language translators or captioning services often fail to capture the nuances and emotional undercurrents of a conversation, leading to stilted or incomplete interactions. Additionally, few existing solutions that offer true bi-directionality—where both hearing and hearing-impaired participants can actively engage in natural, fluid dialogue are expensive or half-baked. Current tools typically struggle with variations in accents, dialects, regional sign language differences, and the subtlety of emotional expressions in facial cues and vocal intonations.

This challenge seeks to develop an end-to-end, real-time solution that translates spoken words into sign language visualisations (such as on-screen avatars or simplified graphic prompts) and sign language gestures into spoken language with synchronised text /speech output.

Beyond mere lexical translation, the system will incorporate emotional intelligence—identifying sentiment through facial expressions, vocal tone, and body language—and integrate a dynamic suggestion engine.

This engine will propose contextually relevant phrases or signs, enabling both parties to maintain conversational flow, personalise the experience, and improve mutual understanding over time. By addressing accuracy, latency, personalisation, and emotional context, this solution aims to foster a more inclusive, empathetic communication platform.

Context and Relevance:

Hearing-impaired individuals often rely on sign language to communicate. However, in everyday scenarios—such as workplace meetings, educational settings, healthcare consultations, and social gatherings—communication can stall or require a dedicated human interpreter. This dependency hinders independence and can lead to misunderstandings. Conversely, neurotypical individuals may lack the proficiency or confidence to engage in sign language. The proposed solution empowers both groups to communicate effortlessly, respecting cultural and linguistic diversity, emotional authenticity, and user preferences.

From a technical perspective, recent advances in computer vision, natural language processing (NLP), speech recognition, and reinforcement learning provide fertile ground for developing a robust system. Highly accurate sign recognition models (leveraging deep learning for pose estimation and gesture classification), emotion detection algorithms (trained on multimodal datasets), and translation models (fine-tuned on bilingual sign-spoken corpora) are now feasible on consumer-grade hardware. Integration of these components, combined with context-aware suggestion engines, can yield a practical, user-friendly tool that transcends current limitations.

10-Day Development Plan:

The condensed timeframe of a 10-day hackathon demands a focused, iterative approach:

Day 1-2: Problem Framing & Data Preparation

- Finalise the scope, user scenarios, and key functionalities (bi-directional translation, emotional cues, contextual suggestions).

- Assemble and preprocess a dataset consisting of sign language video samples (preferably from open-source repositories), aligned spoken audio transcripts, and a small parallel corpus to train translation models. Include emotional datasets (e.g., audio recordings tagged with emotions, facial expression images).
- Revert to experts with contextual queries to understand the problem statement better.

Day 3-4: Baseline Model Development

- Implement a baseline pose estimation model for identifying hand and body keypoints from sign language video input (e.g., using MediaPipe or a lightweight pose estimation model).
- Integrate an off-the-shelf Automatic Speech Recognition (ASR) system (e.g., Whisper by OpenAI or wav2vec2-based model) for converting spoken input to text.
- Develop a basic NLP pipeline for text-to-text translation, enabling a draft of speech-to-sign and sign-to-speech mapping.
- Initiate a rule-based emotional recognition component (e.g., simple facial expression detection via OpenCV and a trained emotion classifier).

Day 5-6: Enhanced Translation & Emotional Intelligence Integration

- Fine-tune NLP models on sign-to-text and text-to-sign pairs, improving translation accuracy by incorporating transformer-based architectures (e.g., BERT, T5, or GPT-based models).
- Integrate a sentiment analysis and prosody-based emotion detection (using pre-trained models) to add emotional context to the translation layer.
- Begin incorporating context-aware suggestion logic—initially, rule-based, using keywords and common phrases from the conversation history.

Day 7: Contextual Suggestion Engine & Personalisation

- Implement a recommendation system that suggests next possible phrases or signs based on the conversation so far. This could use a small language model fine-tuned on frequently used sign phrases and their corresponding spoken language pairs.
- Introduce a simple user feedback loop (mock-up UI button to rate suggestions), setting the stage for reinforcement learning in future iterations.

Day 8: Latency Optimisation & Integration

- Optimise the models for real-time performance. Consider model pruning, quantisation, or running inference on GPU-accelerated platforms to reduce latency.
- Streamline the pipeline: sign recognition → translation → suggestion → output (visualisation or speech) with minimal delay.
- Integrate all components into a single, cohesive prototype application. Use a smartphone interface for displaying sign suggestions and a simple AR overlay or a mobile phone screen avatar for sign output.

Day 9: UI/UX Testing & Refinement

- Conduct user testing simulations:
    - Check how smoothly the system handles live, back-and-forth conversations.

- Evaluate how well emotional cues are captured and displayed.
- Test the contextual suggestion quality and its relevance to ongoing conversation.

- Gather quick feedback from test participants (e.g., hackathon peers, mentors) and make rapid adjustments.

Day 10: Final Integration, Polishing & Presentation

- Refine the UI/UX, ensuring the output is clearly visible and understandable to both parties.
- Final optimisations for accuracy and latency.
- Prepare a compelling demo scenario and presentation narrative, highlighting two-way, emotionally intelligent communication and contextual suggestions.

Leveraging AI to Expedite Development and Improve Accuracy:

- **Pre-Trained Models & Transfer Learning:** Using large, pre-trained models (e.g., transformers for text translation, pose-estimation libraries) significantly reduces the model training time and eases integration challenges. Transfer learning allows the team to achieve higher accuracy in fewer training epochs, essential in a tight hackathon schedule.
- **AutoML & Hyper-parameter Tuning:** Automated machine learning tools can rapidly iterate over various architectures and hyper-parameters, quickly converging on a workable solution without extensive manual experimentation.
- **Real-Time Analytics & Feedback Loops:** Continuous monitoring of intermediate outputs (like sign recognition confidence scores, speech-to-text transcription quality) can help developers adjust pipelines on the fly.
- **Cloud-Based Infrastructure:** Utilising cloud compute resources (GPU instances) enables faster model training, scaling, and real-time inference.
- **Pre-Built Emotion & Gesture APIs:** Incorporating pre-built APIs or frameworks (e.g., Microsoft Cognitive Services for emotion detection, Google Cloud for speech recognition) can jumpstart the pipeline, enabling the team to focus on integration and customisation rather than building all models from scratch.

By combining these strategies and focusing on modular, incremental integration, the team can quickly stand up a working prototype. The result will be a promising proof-of-concept that can be polished further into a robust, market-ready solution after the hackathon ends.

Rules of Engagement:

Intellectual Property Ownership:
All solutions and associated work products created during the hackathon directly address Mythicka's flagship product needs(already conceptualised). While students may be recognized as co-authors of any Intellectual Property (IP) developed, all IP rights—including copyrights, patents, trademarks, and any other IP that may arise—will remain with Mythicka. Participation in the hackathon constitutes an understanding and acceptance of this arrangement.

Handover of Deliverables:
Upon completion of the hackathon, students will provide Mythicka with all developed materials, including but not limited to source code, documentation, design assets, and trained model files. These deliverables become the property of Mythicka for further development, commercialisation, or any other purpose Mythicka sees fit.

No Formation of External Entities:
Students are not invited, required, or encouraged to form any separate entity—such as a startup or company—based on their hackathon solutions. The objective is to solve Mythicka's defined problem statements, and any external commercialisation or separate venture creation is not authorised.

Involvement Beyond the Hackathon:
If Mythicka wishes to further develop a promising solution into a Minimum Viable Product (MVP) or beyond, it may invite participating students to continue their involvement. Such an invitation does not imply or grant any equity, shares, or ownership stake in Mythicka. Any terms of engagement, whether paid or otherwise, will be determined at Mythicka's sole discretion, with no entitlement inferred for the students.

Recognition and Credit:
While Mythicka retains ownership, Mythicka may publicly acknowledge and credit students for their contributions, such as through mentions in official communications, press releases, or product documentation. Such recognition will not alter the agreed-upon IP or ownership rights.

By agreeing to champion this problem statement, you agree to the above terms and conditions of engagement.