Práctica de whisper: Reconocedor de fechas

Tecnologías del lenguaje humano

Giuseppe Di Palma
gdipal1@upv.edu.es

January 15, 2026

1 Model Overview and Generation Strategy

Recent advances in automatic speech recognition are largely driven by Transformer-based encoder–decoder architectures, as popularized by Whisper-like models. In this work, we adopt a standard attention-based sequence-to-sequence framework and focus on its application to multilingual date recognition, instruction following, and function-oriented generation. The model follows a classical encoder–decoder Transformer design and therefore we omit a detailed discussion of well-known architectural components, concentrating instead on the audio frontend, the training objective, and the generation mechanisms required by the different tasks.

1.1 Audio Encoding and Training Setup

Raw audio waveforms are transformed into log Mel-spectrograms, which are then linearly projected into the model embedding space. During training, SpecAugment is applied to improve robustness through time warping and masking in both time and frequency domains. The encoder processes the resulting feature sequence using stacked self-attention layers, producing a sequence of latent representations that summarize the acoustic content. The decoder is trained autoregressively using teacher forcing, predicting the next token given the previous ground-truth tokens and the encoder outputs. The training objective is standard cross-entropy loss, ignoring padding symbols.

1.2 Autoregressive Generation

At inference time, the decoder operates in an autoregressive manner, generating one token at a time until an end-of-sequence symbol is produced or a maximum length is reached. Unlike training, generation does not rely on teacher forcing and therefore requires specialized decoding strategies.

The model supports multiple generation approaches:

- **Greedy decoding**, selecting the most probable token at each step.
- **Sampling with temperature**, enabling stochastic generation.
- **Top-k sampling**, restricting sampling to the k most likely tokens.
- **Beam search**, maintaining multiple hypotheses in parallel to improve sequence-level optimality.

These strategies allow adapting the decoding behavior depending on the task requirements, ranging from deterministic transcription to more flexible generation.

1.3 Caching for Efficient Decoding

A key implementation aspect concerns decoding efficiency. Naively recomputing self-attention over the entire generated sequence at each time step leads to unnecessary computational overhead. To address this issue, a caching mechanism for attention keys and values is implemented in the decoder. During generation, previously computed keys and values are stored and reused, so that each decoding step only processes the newly generated token. This significantly reduces the computational cost and makes the approach scalable to longer sequences, while preserving identical outputs with respect to the non-cached formulation. Overall, this design provides a compact yet flexible framework capable of supporting multilingual transcription, translation, and function-oriented generation within a unified Transformer-based architecture.

2 Task-Specific Adaptations

All tasks are implemented using the same Transformer-based audio encoder–decoder architecture described previously. The differences between Task 1, Task 2, and Task 3 lie exclusively in the target representation, tokenization strategy, and inference procedure, while the underlying model and training objective remain unchanged.

2.1 Task 1: Date Recognition Baseline

Task 1 establishes the baseline system for spoken date recognition. A word-level tokenizer is constructed from the training transcripts, including special symbols for padding and sequence boundaries. The model is trained to directly transcribe spoken dates into text using paired audio–text examples. During training, teacher forcing is applied and the model is optimized with a cross-entropy loss. At inference time, different decoding strategies are evaluated, including greedy decoding, stochastic sampling, top-k sampling, and beam search. Performance is measured using Word Error Rate (WER), and attention visualizations are used to qualitatively inspect the alignment between audio frames and output tokens. This task serves as the foundation for all subsequent extensions.

2.2 Task 2: Instruction-Conditioned Generation

Task 2 extends the baseline by enabling instruction-based behavior within a single unified model. This is achieved by introducing a set of special action tokens (e.g. `<transcribe_es>`, `<translate_es_en>`) that are prepended to the decoder input sequence. The dataset is constructed by combining Spanish and English audio samples and pairing each audio signal with multiple target texts corresponding to different actions: transcription or translation. The desired operation is therefore fully specified by the instruction token, while the acoustic input remains unchanged. At inference time, the instruction token is explicitly provided to the decoder before generation starts. This allows the same trained model to dynamically switch between transcription and translation tasks without architectural changes. Evaluation is again performed using WER, comparing the generated text against the task-specific reference.

2.3 Task 3: Function-Oriented Generation

Task 3 further extends the system by training the model to generate executable function calls rather than plain text. The target sequences include a separator symbol followed by a short Python expression encoding a date computation (e.g. relative or next-day calculations). Training data from both languages are merged into a single bilingual dataset, preserving the same tokenization scheme as Task 1. The model learns to jointly recognize the spoken date expression and generate the corresponding function call. At inference time, the generated output is post-processed to extract the code segment, which is then executed by an external interpreter. The final prediction is obtained from the function output rather than from the raw text. Consequently, evaluation is performed by comparing the computed date against the reference value, reporting accuracy instead of WER. This task demonstrates how the same sequence-to-sequence architecture can be adapted to structured output generation and symbolic reasoning by modifying only the output format and evaluation pipeline.

3 Experimental Results

Due to limited computational resources, all experiments were conducted without GPU acceleration. As a consequence, the number of training epochs and the overall model capacity were intentionally kept small. For this reason, the reported performance metrics should not be interpreted as competitive with state-of-the-art systems.

The primary goal of the experimental evaluation is therefore to verify the correctness and completeness of the implementation, rather than to optimize accuracy. This design choice is consistent with the assignment guidelines, which explicitly state that functional correctness is the main requirement.

For Task 1 (date recognition), the model is able to generate syntactically valid date strings and successfully overfits small subsets of the training data, demonstrating that the encoder–decoder pipeline and loss formulation are correctly implemented. Quantitative results in terms of Word Error Rate (WER) remain high when evaluated on the full test set, which is expected given the limited training regime.

For Task 2 (instruction-conditioned generation), the model correctly reacts to different instruction tokens, producing either transcriptions or translations depending on the specified action. While the linguistic quality of the outputs is limited, the experiments confirm that the instruction tokens effectively control the decoder behavior without requiring architectural modifications.

For Task 3 (function-oriented generation), the model is able to generate well-formed function calls that can be parsed and executed by an external interpreter. Although prediction accuracy is low, the end-to-end pipeline—from speech input to executable output—is fully functional, validating the proposed approach.

Overall, the experimental results confirm that the system behaves as expected across all tasks, despite suboptimal numerical performance.

4 Conclusions

In this work, we implemented a Transformer-based audio encoder–decoder model capable of handling multiple speech-related tasks within a unified architecture. Starting from a baseline date recognition system, we extended the model to support instruction-conditioned generation and structured function output without modifying the core architecture. A key contribution of this implementation is the explicit separation between training and inference procedures, including the integration of multiple decoding strategies and an efficient caching mechanism for autoregressive generation. These components are essential for practical deployment and align with modern large-scale speech and language models.