A Multilingual Speech-Driven Semantic Input Module for University Indoor Wayfinding

Gokhan Ceylan
Sapienza University of Rome
Italy
ceylan.2124426@studenti.uniroma1.it

Gabriella Trasciatti
Sapienza University of Rome
Italy
trasciatti@di.uniroma1.it

Emanuele Panizzi
Sapienza University of Rome
Italy
panizzi@di.uniroma1.it

Abstract

This paper presents an offline speech-driven semantic input module born to support an indoor navigation application in the context of Sapienza University of Rome. The system addresses the difficulty international students face when searching for rooms, as Italian names are often mispronounced and incorrectly transcribed by standard speech-to-text (STT) systems. To operate in buildings with limited connectivity, the solution relies entirely on on-device processing and does not require cloud services or dedicated speech recognition models. Spoken queries are normalized through domain-specific vocabulary correction and spoken number extraction before being matched against a local repository of 243 rooms across four campus buildings. The matching engine combines Levenshtein similarity with an Italian-specific phonetic representation to improve robustness against transcription errors and pronunciation variation. Evaluation on 150 spoken queries achieved 92% top-candidate accuracy, with an additional 6% correctly resolved through disambiguation. The complete resolution pipeline executes in 8.0 ms on consumer mobile hardware, demonstrating the feasibility of lightweight, real-time, offline voice search to support indoor navigation applications.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**; *Ubiquitous and mobile computing*; • **Computing methodologies** → *Natural language processing*.

Keywords

voice search, indoor navigation, offline speech recognition, fuzzy matching, multilingual NLP, mobile HCI

ACM Reference Format:
Gokhan Ceylan, Gabriella Trasciatti, and Emanuele Panizzi. 2026. A Multilingual Speech-Driven Semantic Input Module for University Indoor Wayfinding. In *Companion of the INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION (ICMI Companion '26), October 05–09, 2026, Napoli, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3776591.3832512

1 Introduction

This work was born as part of a university project that aims to help students navigate the campus by implementing a mobile navigation system. The challenge of navigating the campus stems from the large number of rooms, the scarcity of information about them, and the use of Italian names for rooms and buildings, which many international students find difficult to spell, pronounce, and remember, especially at the start of each semester. While modern smartphones provide built-in speech-to-text (STT) functionality, it has proven insufficient for accurately transcribing Italian names, especially when spoken by users who are not native Italian speakers, regardless of the language setting used by the STT engine. In our context, campus buildings often suffer from poor or nonexistent network coverage, making cloud-based transcription impractical. Furthermore, to the best of our knowledge, there is currently no widely adopted approach for implementing fully offline, mobile domain-adapted voice search in indoor navigation systems.

The system proposed in this paper is thus defined by the following three constraints that shaped its design. First, it must operate without any network connection at query time, since basement floors and dense concrete structures routinely block both Wi-Fi and cellular signals. Context-aware mobile systems increasingly exploit information available from the user environment and device context to adapt their behavior to specific operational constraints [3]. Second, it must handle two languages, Italian and English, reflecting the real user base at Sapienza where international students and visitors cannot be expected to speak Italian as natives. Third, it cannot bundle large dedicated speech models: storage requirements are prohibitive. Achieving acceptable bilingual accuracy with Whisper requires the 'base' model (approximately 142MB), while Vosk requires bundling separate acoustic models per language (nearly 100MB total), which is structurally unjustifiable for a mobile application already bundling offline features for indoor navigation. By contrast, the room repository used by our system totals under 100KB across all four buildings, as it requires no speech model and relies entirely on the OS native STT engine. The speech-driven semantic input module presented in this work is implemented in Flutter and integrated into the existing navigation application built for Sapienza. It resolves spoken room queries against a local repository of 243 rooms across four buildings and operates with no network dependency at query time. To recover from common STT transcription errors, the system applies phonetic preprocessing and custom *fuzzy matching*, allowing approximate rather than exact string matches and compensating for character-level transcription noise. The building repository was assembled through direct field visits to all four campus buildings, capturing both official room designations and the informal names students actually use, since those two differ significantly.

ICMI Companion '26, Napoli, Italy

ACM ISBN 979-8-4007-2319-3/2026/10
https://doi.org/10.1145/3776591.3832512

2 Related Work

Voice-based interaction in mobile applications has been studied extensively. Commonly found commercial systems such as Google Assistant [8] and Siri [2] route audio to remote servers for transcription, which makes them unsuitable for environments where network access is unreliable. On-device alternatives such as Vosk [1] offer offline transcription but provide raw text with no domain adaptation, leaving the gap between transcribed input and application-specific entities unresolved.

String similarity metrics have a long history in record linkage and name matching tasks [6, 9]. Levenshtein edit distance [11] remains a strong baseline for short strings with character-level errors, and Cohen et al. [5] show it compares favorably against more complex metrics for name matching. Phonetic encoding, introduced by Russell [14] and extended by Philips [13], addresses sound-alike errors that edit distance alone cannot catch. Winkler [16] demonstrates that combining similarity metrics improves precision in record linkage tasks. Our system applies this insight directly: Levenshtein similarity and a language-specific phonetic key are combined at the token level to handle both transcription noise and accent-driven pronunciation variation.

The most architecturally relevant prior system is Soundscape [15], which applies fuzzy string matching to geographic search. However, Soundscape operates over typed input only and has no internal voice resolution engine: it delegates spoken interaction to platform-level accessibility layers like VoiceOver or TalkBack, which means it cannot process a free-form noisy spoken phrase and map it to a semantic indoor destination. The specific combination this paper addresses, fully offline bilingual voice resolution against a curated indoor room database, is not covered by any prior system we are aware of. Accessibility-focused navigation tools such as GoodMaps [7] and Lazarillo [10] support spoken output but rely on typed input or platform screen readers for destination entry, leaving the voice-input problem unsolved.

3 System Overview

The module is organized as a three-stage pipeline. Raw audio is captured and transcribed by the speech recognition layer, the resulting text is normalized and parsed by the NLP preprocessing layer, and the cleaned query is matched against the room repository by the semantic matching engine. All three stages run on-device with no network calls at query time, following a broader trend in mobile computing where smartphones are used as autonomous sensing and processing platforms rather than merely as clients of remote services [4, 12].

The room repository covers four buildings on Sapienza's campus, summarized in Table 1. Each room entry stores Italian and English names, a list of room number identifiers, room type, floor, and an optional list of staff occupants, such as professors' names. Room numbers are stored as arrays to accommodate rooms known by multiple identifiers; for example, a basement room may carry both a positive identifier and a floor-prefixed one.

Figure 1 shows the three possible outcomes implemented for voice queries: a confirmed single match, a disambiguation list when multiple rooms score above the confidence threshold, and a no-match response when no room clears the minimum score.

Table 1: Composition of the building repository dataset.

Building	Code	Rooms	Occupants
Ex Vetrerie Sciarra	RM103	47	19
Edificio Marco Polo	RM021	82	43
Facoltà di Lettere e Filosofia	CU003	48	9
Villa Mirafiori	RM052	66	45
Total		**243**	**116**

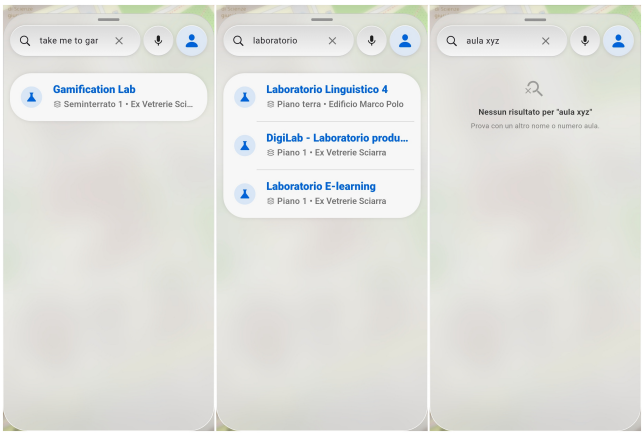

Figure 1: Voice query outcomes: single confirmed match (left), disambiguation list of three (center), and no match found warning (right).

3.1 Speech Recognition and Connectivity Detection

The system uses the device operating system's native speech-to-text API via the `speech_to_text` Flutter package. Before each recognition session, the module performs a DNS lookup solely to determine whether the device is online or offline; no data is downloaded. When online, the module explicitly sets the STT locale to match the user's chosen language: for Italian it selects the first available locale beginning with *it*; for English it evaluates a priority list of en-US, en-GB, en-AU, en-IN, and en-CA in order. When offline, the locale is left unset and the OS STT engine falls back to whichever on-device language model it already has installed, preserving basic transcription functionality without any network dependency.

The recognition session is configured to emit partial results: the API delivers an updated transcript after each recognized word, and each emission is immediately passed through the full NLP pipeline. For short queries such as a room number or a single occupant surname, the resolver may accumulate sufficient confidence to commit to a result before the speaker has finished the utterance. If no confident interpretation can yet be established, however, the resolver simply continues processing subsequent partial transcripts as they arrive from the STT system. The search bar in Figure 1 shows an example of this behavior: the confirmed match on the left was produced while the query was still live.

3.2 NLP Preprocessing

As a first step, before the actual matching, the raw transcript goes through a normalization step that strips diacritics, removes filler words common in Italian and English speech, and applies a domain-specific ontology substitution map. The ontology covers two categories of correction: phonetic variants that STT engines commonly produce for Italian academic vocabulary (for example, *aula* transcribed as *aola*, *laura*, or *hola*), and abbreviation expansions used in spoken navigation requests (for example, *lab* to *laboratorio* in Italian mode, or *lab* to *laboratory* in English mode). The Italian dictionary contains 25 entries and the English dictionary 22. All entries were sourced from actual failed transcripts collected during field sessions across the four buildings by students, rather than constructed synthetically, which ensures each correction targets a hallucination pattern the system genuinely encounters. The filler removal step applies a complementary list of 47 navigational phrases in both languages, covering constructions such as *take me to*, *portami al*, *where is*, and *dove si trova*, so that the token passed to the matching engine contains only the destination identifier. After normalization, the module attempts to extract a room number from the query using a five-strategy cascade: regular expression matching, sequential single-digit grouping, English and Italian spoken number parsers, Roman numeral recognition (commonly found in Italian universities), and a standalone letter fallback for rooms identified by a single character. The Italian parser handles fully composed spoken forms of numbers like *duecento diciotto* by hierarchical decomposition into hundreds, tens, and units, yielding 218. The English parser handles additive forms like *two hundred and eighteen* the same way.

3.3 Semantic Matching Engine

After attempting number extraction, the module checks whether the extracted number exactly matches any entry in the repository's number arrays. If a match is found, the corresponding result is returned immediately. If no valid number is extracted, the query proceeds to fuzzy name matching. In this case, each room name is scored token by token using a novel combined metric: Levenshtein similarity plus a phonetic bonus that depends on whether the consonant skeleton of the query matches that of the candidate token, either exactly (+0.20) or as a substring (+0.05). The consonant skeleton is built by lowercasing the token, collapsing the Italian digraphs *chi*, *che*, *ghi*, *ghe*, *ci*, and *ce* into a single k/g-based consonant (deliberately merging the hard/soft distinction), reducing *gl* to l and *gn* to n, collapsing *sc* to s, dropping the silent letter h, substituting q with k, collapsing any remaining repeated consonants, and finally removing all vowels. Table 2 shows examples of how consonant skeletons stay the same even when STT introduces transcription errors.

The resolver also applies score bonuses according to a set of heuristics specific to a navigation application context. For example, it favors candidates who signal the entrance of a building. Staff occupant names are scored separately using the same combined metric, with surnames weighted at 70% and first names at 30%. A single-token query scoring above 0.80 against a surname is returned directly without penalty for the missing first name, as it is deemed sufficient to correctly identify a particular surname, letting the user manually disambiguate if more than a single match of that surname

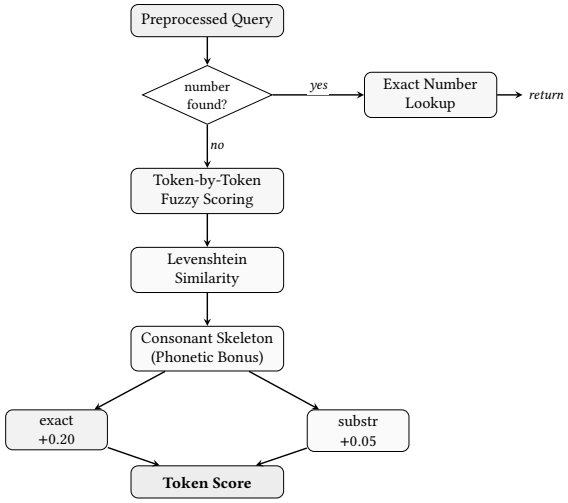

Figure 2: Semantic Matching Engine resolution flow.

is returned. A room passes the decision threshold when its top score reaches 0.85 or higher, or when it leads the next candidate by a margin of at least 0.10. Queries that do not clear either threshold return no match. All of the scores and thresholds chosen in this phase have been tuned to achieve a desired behavior, in particular to avoid returning certain low-confidence results. For example, it was decided to show results matching surnames even with low confidence, whereas first names are penalized more because they are more common. These values are application-specific, however, and should be re-evaluated and adjusted when adapting the system to a different domain, dataset, or set of user requirements.

Table 2: Phonetic key and Levenshtein distance for four common STT errors.

Spoken Input	Target Room	Phonetic Key	Levenshtein Distance
Aola	Aula	l	1
Labratorio	Laboratorio	lbrtr	1
Bibiloteca	Biblioteca	bbltc	2
Semianrio	Seminario	smnr	2

4 Evaluation

We evaluated the module against a test set of 150 spoken queries (75 Italian, 75 English) collected through informal field testing at each of the four campus buildings. Individuals present at each location were asked to speak a natural navigation query for a room they were already familiar with, without being shown any room list, spelling, or dataset beforehand, so that pronunciation and phrasing reflected genuine unprompted speech. Six participants took part overall (5 non-native Italian speakers, 1 native Italian speaker), each recruited from within the building where they were tested.

The system resolved 92% of queries to the correct room as the top candidate. In 6% of cases it correctly identified ambiguity and returned a short candidate list for the user to confirm. Only 2% of queries returned no match, all due to severe acoustic distortion

where the transcript bore no phonetic resemblance to any repository entry. Queries containing room numbers resolved with 100% accuracy when the spoken number was correctly transcribed by the STT engine.

A representative edge case illustrates the strength of the phonetic layer. The phrase "Gamification Lab", when spoken with a non-native accent, was transcribed by the OS engine as "game vacation lab". A standard Levenshtein comparison fails here because the two phrases share almost no character structure. The phonetic key function reduces *gamification* to `gmfctn` and *game vacation* to `gmvctn`. The high partial key overlap, combined with the phonetic bonus, correctly resolves the noisy transcript to the target room above all other candidates.

Table 3 reports average execution times measured on a Google Pixel 6a. The latency measurement covers the worst-case path: a query with no extractable room number, forcing a full fuzzy scoring pass across all 243 rooms and their occupant lists. The test was repeated 1,000 times to produce a stable average. The total resolution latency of 8.0 ms falls well within the 16.6 ms frame budget for a 60 FPS application, confirming that synchronous execution on the main UI thread is viable for datasets of this scale. The voice module adds no meaningful storage overhead: the room repository totals under 100KB across all four buildings and no speech model is bundled. To isolate the contribution of individual components,

Table 3: Average execution time per pipeline stage.

Pipeline Stage	Time (ms)
JSON Repository Load (initialization)	14.2
Normalization and Preprocessing	0.8
Number Extraction (failed path)	0.5
Fuzzy Token Scoring	6.4
Decision Gate and Sorting	0.3
Total Resolution Latency	**8.0**

we ran two ablation conditions on the same query set. Removing the phonetic key layer and relying on Levenshtein similarity alone dropped accuracy from 92% to 41%, with most failures concentrated on vowel-heavy STT errors and English speakers attempting to pronounce Italian room names. Removing spoken number translation caused all number-based queries to fall through to the slower fuzzy name path, increasing average latency and introducing ambiguity in cases where a room name contains a number word. Both results confirm that each component carries its weight in the overall pipeline. The system does exhibit specific failure modes worth noting. The vocabulary substitution dictionaries are compiled at build time, so if a user uses an unexpected word, such as referring to a "Laboratorio" as a "Lobby", the system falls back to Levenshtein matching alone, which cannot bridge that structural gap. A second failure mode involves multi-token type conflicts: a query like "Aula Tre" misheard as "Paolo Tre" can cause the occupant scoring path to prioritize a professor named Paolo over returning no match, since the surname weighting logic scores aggressively even on weak evidence. The first failure mode points toward a dynamically generated ontology that adapts to the specific vocabulary of each deployment domain. The second requires a type-aware scoring approach that enforces a stronger confidence gap between the occupant path and the room-name path before committing to a result.

5 Conclusion

This paper presented a speech-driven semantic input module for indoor navigation that runs entirely on-device, with no dependency on cloud infrastructure at query time. The module handles Italian and English, tolerates the kinds of transcription errors that general-purpose STT engines routinely produce on domain-specific vocabulary, and resolves spoken room queries in 8.0 milliseconds on consumer hardware.

The core contribution is the combination of Levenshtein edit distance and a language-specific phonetic key at the token scoring level. Neither technique is new on its own, but their combination inside a structured fuzzy matching pipeline, applied to a bilingual academic building repository, produces robust results that neither approach achieves alone, as the drop from 92% to 41% accuracy in the ablation study confirms.

Several directions remain open. The current phonetic encoder is specific to Italian consonant patterns. Extending the system to additional languages would require a corresponding encoder per language, pointing toward a pluggable phonetic interface as a natural next step. On the data side, manually curating a room repository for each new building is time-consuming. A build-time pipeline capable of extracting and translating room labels from source material (e.g., floor plans) would make deployment substantially faster without affecting runtime behavior.

Safe and Responsible Innovation Statement

This system was built to help people find rooms in university buildings. It does not collect, store, or transmit any audio or personal data. Everything runs locally on the user's device: the microphone input is processed by the operating system's built-in STT engine, and the matching logic runs entirely against a local JSON file. Nothing leaves the device at query time. Microphone access follows the standard platform permission model on both Android and iOS, meaning the user explicitly grants it and can revoke it at any time. The building dataset contains only publicly available room names, numbers, and staff office assignments, the same information posted on public university websites and door signs. There is no user tracking, no behavioral profiling, and no mechanism that could be repurposed for surveillance.

Generative AI Disclosure

Parts of this manuscript were drafted with the assistance of Large Language Models. All technical content, including algorithm descriptions, evaluation results, and system design decisions, was written and verified by the authors, who take full responsibility for the accuracy, originality, and final content of the paper.

References

[1] Alpha Cephei. 2024. Vosk Offline Speech Recognition API. https://alphacephei.com/vosk/

[2] Apple. 2024. Siri. https://www.apple.com/siri/

[3] Enrico Bassetti, Alessio Luciani, and Emanuele Panizzi. 2021. ML Classification of Car Parking with Implicit Interaction on the Driver's Smartphone. *Lecture Notes in Computer Science* 12934 LNCS (2021), 291–299. doi:10.1007/978-3-030-85613-7_21

[4] Alba Bisante, Venkata Srikanth Varma Datla, Gabriella Trasciatti, Stefano Zeppieri, and Emanuele Panizzi. 2024. An Approach to Leverage Artificial Intelligence for Car-Parking Related Mobile Applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 14517 LNCS (2024), 63–71. doi:10.1007/978-3-031-59235-5_7

[5] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web.*

[6] Pierluigi Contucci, Emanuele Panizzi, Federico Ricci-Tersenghi, and Alina Sîrbu. 2016. Egalitarianism in the rank aggregation problem: a new dimension for democracy. *Quality and Quantity* 50, 3 (2016), 1185–1200. doi:10.1007/s11135-015-0197-x

[7] GoodMaps. 2024. GoodMaps Explore. https://goodmaps.com

[8] Google. 2024. Google Assistant. https://assistant.google.com

[9] Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing* (2nd ed.). Prentice Hall.

[10] Lazarillo. 2024. Lazarillo App. https://lazarillo.app

[11] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* (1966).

[12] Emanuele Panizzi. 2016. The SeismoCloud App: Your smartphone as a seismometer. *Proceedings of the Workshop on Advanced Visual Interfaces AVI* 07-10-June-2016 (2016), 336–337. doi:10.1145/2909132.2926070

[13] Lawrence Philips. 1990. Hanging on the Metaphone. *Computer Language Magazine* 7, 12 (1990), 39–44.

[14] Robert C. Russell. 1918. Index. U.S. Patent 1,261,167.

[15] Scottish Tech Army. 2023. Soundscape-Android. GitHub Repository. https://github.com/Scottish-Tech-Army/Soundscape-Android

[16] William E. Winkler. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In *Proceedings of the Survey Research Methods Section, American Statistical Association.* 354–359.