Eötvös Loránd University

Faculty of Informatics

Department of Artificial Intelligence

Homophonic Translation in Hungarian in the Age of Large Language Models

Supervisor:
Indig Balázs
Department of Artificial Intelligence, ELTE IK

Author:
Molnár Líviusz
Computer Science MSc

Budapest, 2025

Contents

Abstract

Homophonic translation aims to render the sounds of a source sentence in another language, using target-language words whose pronunciation mimics the original. This paper studies automatic homophonic translation from Hungarian to English in the age of large language models (LLMs), with a focus on preserving Hungarian phonology rather than meaning.

I propose a modular evaluation framework that isolates the homophonic translation module from the surrounding speech technology and quantifies the preservation of sound. Hungarian text is first converted to IPA with the morphology-aware G2P system emPhon, mapped to English proxy phones using PanPhon feature distances, decoded into English word sequences, then passed through an English TTS and a Hungarian ASR system. This TTS→ASR loop yields word, character and phoneme error rates (WER, CER, PER) against the original Hungarian, and supports phoneme-class and cost–accuracy analyses.

Within this framework three systems are implemented: (i) DP-Phone, a dynamic-programming phone/character decoder that searches CMUdict with PanPhon-derived costs; (ii) Morph-Dict, a morph n-gram based English dictionary indexed by Hungarian morph contexts; and (iii) a Gemini-based LLM baseline prompted to produce homophonic English paraphrases. All three systems are evaluated inside the same TTS→ASR framework.

The systems are evaluated on a deliberately hard, phonetically balanced Hungarian test set (28 sentences) and on 15 real-life news-style fragments. On the hard-case set, DP-Phone achieves the lowest PER (1.20) with competitive WER/CER, while Morph-Dict attains the best WER (1.02) at the cost of substantially higher PER (1.50). The Gemini baseline yields the highest WER/CER despite fluent English output. On real-life sentences, DP-Phone again provides the best CER/PER and a WER similar to Morph-Dict, while Gemini lags behind in WER. A calibration with reference-quality human homophonic translations shows that the TTS→ASR evaluator itself introduces a non-negligible

noise floor, so in some cases the translation quality is higher than what the metrics can reveal. Nevertheless, explicit phone-level control consistently outperforms the LLM baseline for cross-lingual sound preservation, and the proposed framework offers a reusable testbed for future homophonic translation systems and for extending the approach beyond Hungarian.

Chapter 1

Background and Problem Definition

Homophonic translation involves translating a text based on its sound rather than its meaning. It uses real words from the target language that resemble the pronunciation of the source text, even if the resulting text is semantically nonsensical. This concept has its origins in poetry and humour. For instance, translators and poets have used homophonic translation as a form of 'sound poetry' to playfully reimagine texts across languages. As a personal example, my rubber duck at home has "Kis Kacsa für dich" written on it, which alludes to the Hungarian nursery rhyme "Kis kacsa fürdik" ("little duck is bathing"). The joke relies on the fact that the German phrase "für dich" ("for you") is pronounced similarly to the Hungarian "fürdik" ("is bathing"). A closely related phenomenon is *soramimi* (Japanese for 'misheard lyrics'), whereby listeners interpret foreign-language lyrics as meaningful words in their own language. Soramimi and its English counterpart, the *mondegreen*, highlight how cross-language phonetic similarities can deceive human perception. In soramimi, song lyrics in one language are perceived as a phrase in another language - effectively a *naturally occurring homophonic translation*. These examples demonstrate the potential for conveying the sounds of one language via the words of another.

In essence, this thesis aims to express Hungarian sentences as sequences of English words. Although the problem of homophonic translation is not new, the underlying speech and language technologies are evolving rapidly, so this work focuses on a quantitative and extensible evaluation setup that captures a current snapshot of the field and facilitates future experiments. Within this setup, *solution modules* (my own and those proposed in later studies) can be plugged in and out of a common testing framework. To achieve homophonic translation, an end-to-end LLM baseline will be compared with a small pipeline of state-of-the-art components based on linguistic research. The testing framework is also

modular, allowing it to be easily improved as English TTS and Hungarian ASR tools evolve. It will primarily compare solution modules based on precision metrics, but it will also be able to integrate "ease of use" and other types of metrics. The approach that best preserves Hungarian pronunciation while producing a readable English output will be identified.

1.1 Scope

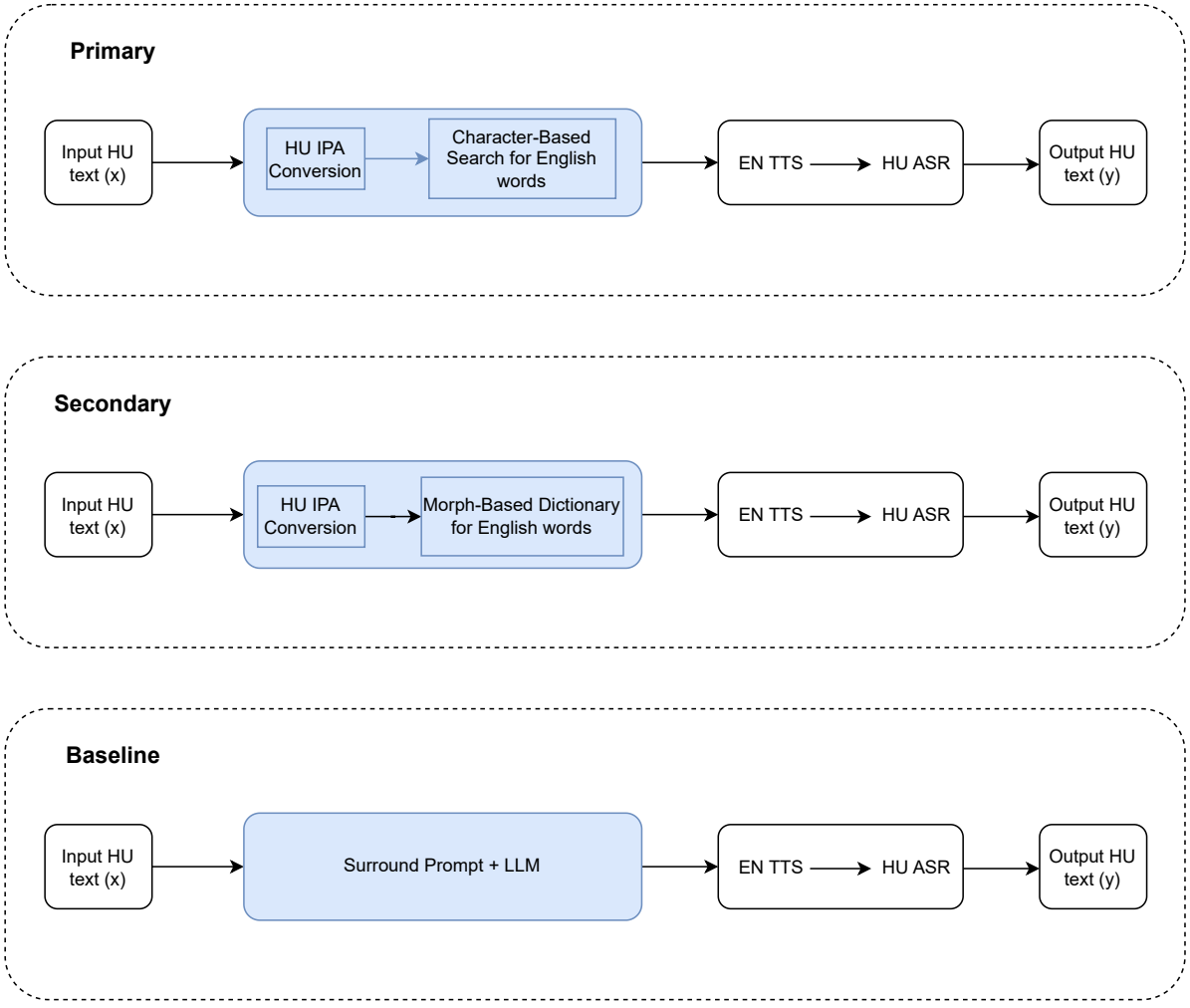

Figure 1.1: The architecture of the three main approaches examined in this thesis. The non-blue parts makes up the shared testing framework(see section 1.2.)

The goal is a simple, modular architecture (see Figure 1.1). The only difference among the systems is the blue module. The remaining parts, which are used only for benchmarking, are composed from the same state-of-the-art solutions.

- **Primary System:** converts Hungarian text to IPA and maps the resulting Hungarian phoneme sequence to a target sequence of English phonemes. It then searches a constrained English pronunciation lexicon (words annotated with IPA) to select a sequence of words whose concatenated phonemes best match the target sequence.
- **Secondary System:** converts Hungarian text to IPA, uses a pre-created Hungarian morph n-gram to English word dictionary for a matching word sequence.
- **Baseline System:** a *minimal LLM* that is prompted to produce an English phrase which, when read aloud, resembles the Hungarian pronunciation. Prompting is kept simple (no heavy prompt engineering), and restrict output to common-word vocabulary where possible. This baseline helps quantify what an off-the-shelf generative approach can do without specialized phonetic machinery.

All three of these systems are open to variation; the homophonic translation module in each system can be replaced by new ideas in future research. Figure 1.1 is a high-level schematic of the approaches used, which is elaborated on in Chapter 2.

1.2 Evaluation and Metrics

To evaluate the system, output of the blue module is fed into an English Text-To-Speech (TTS) engine, which simulates reading the phrase aloud. The resulting audio is then processed by a Hungarian Automatic Speech Recogniser (ASR) to simulate a native listener interpreting the sounds as Hungarian. The Hungarian transcription produced by the ASR is then compared with the original Hungarian input sentence(see 1.1.). This TTS-to-ASR loop enables me to objectively and reproducibly measure how well the generated English text preserves the sound of the Hungarian input. For comparison, industry-standard TTS and ASR evaluation metrics are applied.

Word Error Rate (WER). the primary metric is WER [1], computed after dynamic-programming alignment of hypothesis (HYP) to reference (REF):

$$\mathrm{WER} = \frac{S + D + I}{N_{\mathrm{ref}}},$$

where S, D, I are substitutions, deletions and insertions, and N_{ref} is the number of reference words. NIST `sclite` [1] is used to align and score,

Character Error Rate (CER). As a secondary metric CER is reported, the same normalized edit distance is computed at the *character* level. CER is informative for short items and morphologically rich languages [2], and is widely used in modern ASR/TTS evaluation toolchains.

Phone/Phoneme Error Rate (PER). To directly assess pronunciation fidelity, PER is also computed between the *intended* phoneme sequence (phonemized version of input) and a phonemized version of the ASR output. PER has been used as an objective proxy for intelligibility [3] and to select TTS models with better human-rated intelligibility.

Alignment outputs and per-class analysis. In addition to scalar scores, the sclite alignment outputs are used to derive error breakdowns and confusion profiles. The test set (Appendix A.0.1) is populated with phonetic 'hard cases' such as front-rounded vowels, long-short pairs, palatals (ty/gy/ny), sibilants, affricates and assimilation Per-class WER/CER values and phone-level confusion statistics are also reported for interpretability.

1.3 Hungarian-specific challenges

Several core components used in this work are *English-centric* and widely used across speech pipelines—for example, the **CMU Pronouncing Dictionary (CMUdict)** [4] for word→phone lookups, and strong open source ASR baselines such as **Whisper** [5] and **wav2vec 2.0** [6]. These tools have seen broad success in English and have multilingual variants or fine-tunes, but Hungarian differs in ways that matter for a homophonic objective, so a direct transfer is not trivial. Further details can be seen in Section 2.1.

Where Hungarian diverges from English (and why it matters)

- **Vowel inventory and length.** Hungarian contrasts seven short–long vowel pairs and includes *front-rounded* vowels (/ö, ő, ü, ű/) that standard English lacks. English TTS voices tend to neutralize length and have no [y]/[ø] category; therefore, these targets must be approximated with English sequences and verified empirically. These properties are core to Hungarian phonology [7].

- **Palatals and affricates.** The palatal series (/ty [c], gy [ɟ], ny [ɲ]/) and the affricate *dzs* [dʒ] have no one-to-one English counterparts. English realizations often surface as clusters (e.g., [tj], [dj]/[dʒ], [nj]), which can shift perception on a Hungarian ASR and must be handled in my phone mapping and lexicon constraints [7].

- **Sibilant pairs and orthography.** Hungarian *s* = [ʃ] and *sz* = [s] (opposite of English readers' expectation), and similarly *z* vs. *zs* ([z]/[ʒ]). Steering an English TTS toward the intended phone depends on orthographic control (e.g., "sh", "zh"). [7].

- **Contextual phonology.** Productive obstruent *voicing assimilation* across word boundaries is typical in Hungarian, but not in English; an English TTS will not apply it automatically. If too strict orthographic pronunciations are targeted, a Hungarian ASR may penalize naturally assimilated outputs (or vice versa), so both orthographic and assimilated targets will be considered in evaluation [7].

Implications for the pipeline approach: English resources (e.g., CMUdict and English TTS voices) are treated as *tools of convenience*, not ground truth for Hungarian. Concretely, (i) Hungarian phones are explicitly mapped to English approximations; (ii) the decoder searches a *constrained* lexicon so that the TTS realises the intended phones; and (iii) evaluate with **Hungarian** ASR (WER/CER/phone-level analyses) to measure success in the intended language. This framing acknowledges what English-centric pipelines do well while addressing the specific phonological hurdles of Hungarian [7].

1.4 Research Questions and Hypotheses

RQ1 (Primary). Does a phone-based homophonic translation module achieve *lower Hungarian ASR WER* on a phonetically balanced HU test set than the end-to-end architecture that uses an LLM as its baseline?

H1. A phone-based module produces a statistically significant WER reduction vs. the baseline.

RQ2. Does restricting search to a curated, morph-based English lexicon improve output quality and/or decoding speed for Hungarian→English homophonic translation compared with searching the entire English word vocabulary by matching phone sequences at the character/phone level?

H2A. A full phone/character-level search over the entire English vocabulary achieves equal or better accuracy (e.g., lower WER/CER; higher intelligibility) than a morph-constrained lexicon search.

H2B. The morph-constrained lexicon reaches its optimal operating point with fewer/simpler hyperparameter adjustments (HU–EN phone cost, beam width) than phone/character-level search.

H2C. For the same decoder (beam, pruning), the morph-constrained lexicon yields lower decoding time and fewer search expansions than phone/character-level search.

RQ3. Which Hungarian phoneme classes are most error-prone, and how low an error rate can be achieved for each class (front-rounded vowels, long–short vowel pairs, palatals ty/gy/ny, sibilant pairs s/sz and z/zs, affricates cs/dzs, obstruent assimilation)?

H3. The largest relative gains occur for front-rounded vowels and palatals.

RQ4. How do search constraints affect results and fluency? (lexicon size 1k/5k/10k/15k; optional bigram penalty λ.)

H4. A 5k–10k lexicon outperforms a 1k one in performance (lower WER, more natural English) and a higher lexicon does not yield statistically significant improvements.

Chapter 2

Methodology and System Design

This chapter outlines my end-to-end pipeline for the homophonic translation of Hungarian sentences into English word sequences that preserve pronunciation. The design can be run and reproduced entirely locally.

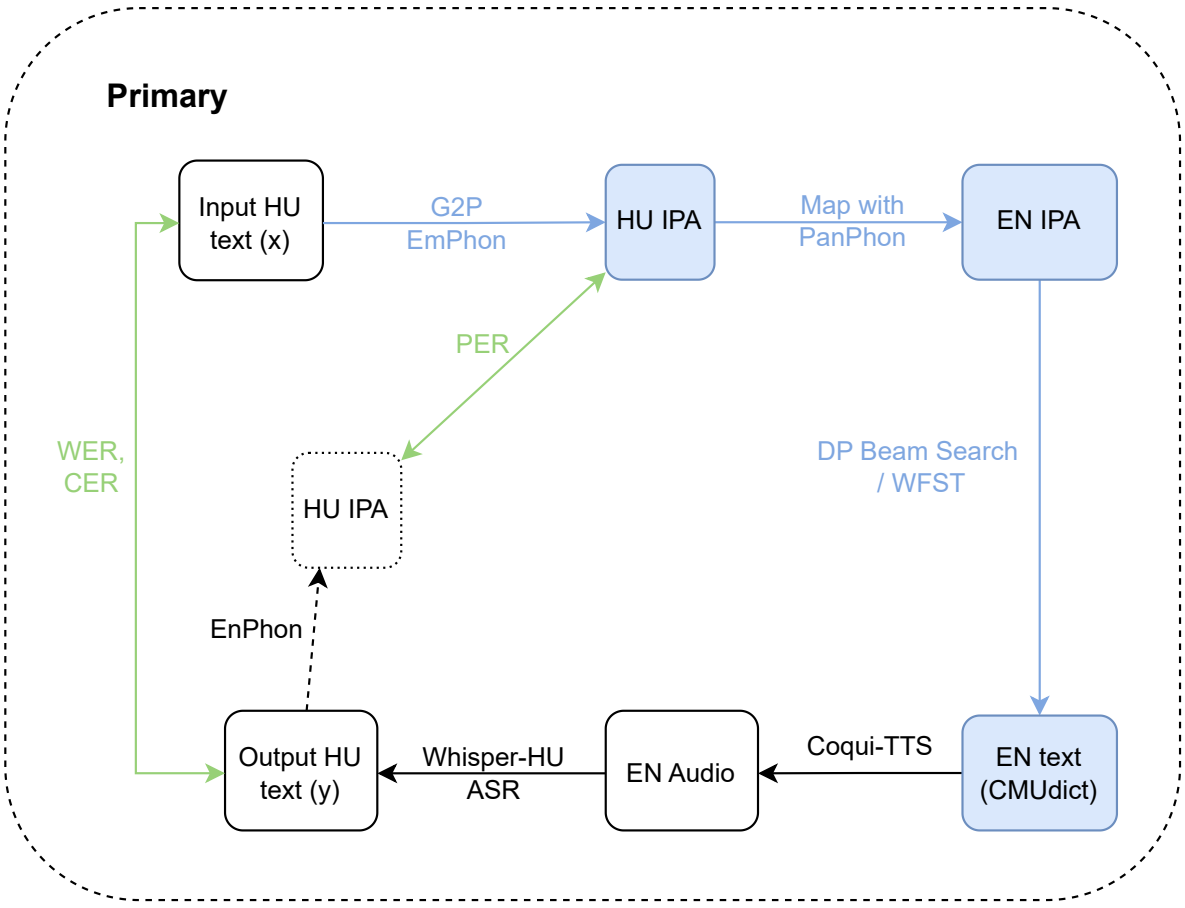

Figure 2.1: Overview of the primary system from input text to evaluated output: phoneme conversion, cross-language phonetic mapping, decoding to English words, text-to-speech, re-transcription, and error measurement (WER/CER/PER).

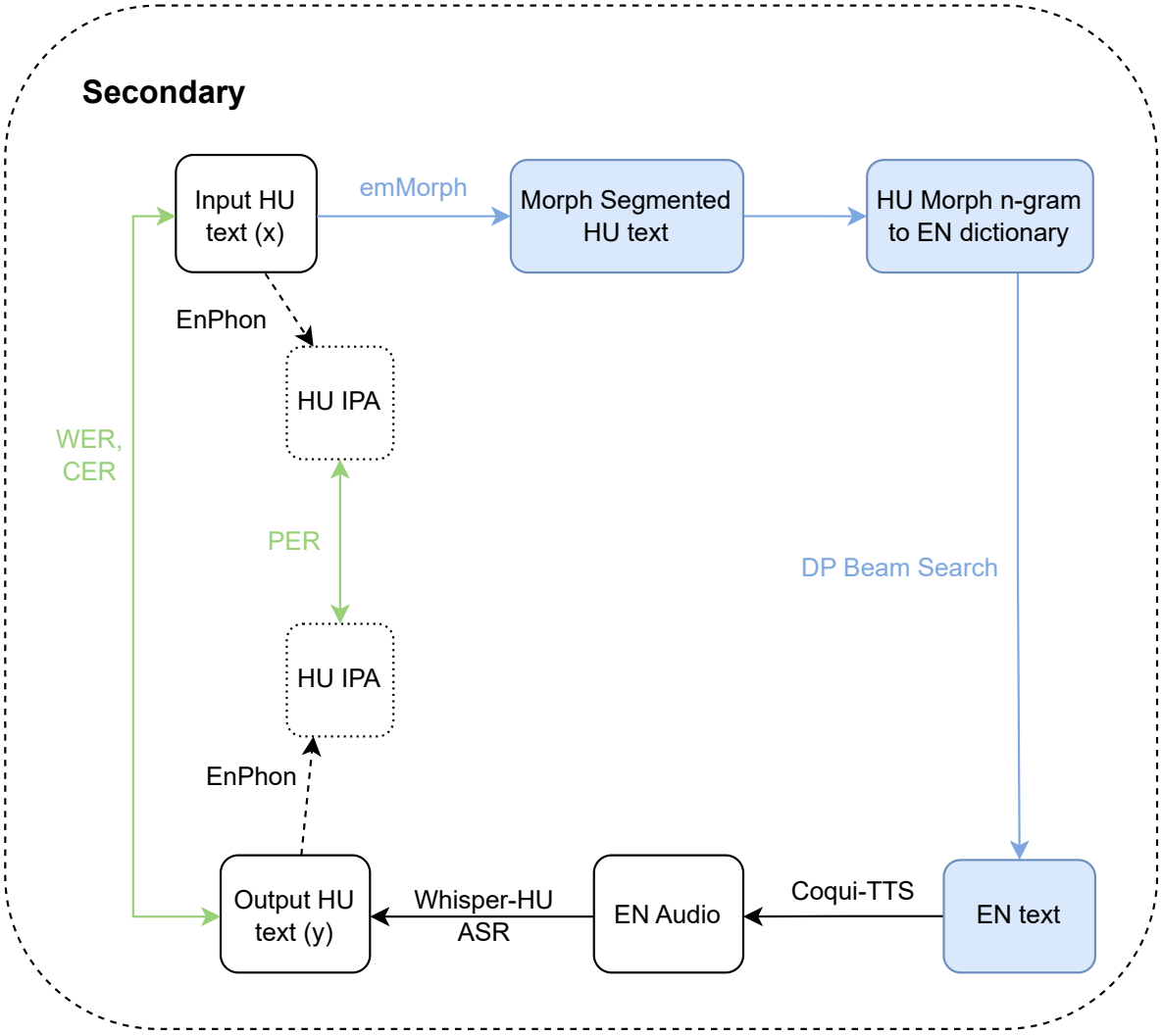

Figure 2.2: Overview of the secondary system from input text to evaluated output: emMorph segmentation, decoding to English words with preprared morph-based EN dictionary, text-to-speech, re-transcription, and error measurement (WER/CER/PER).

Figures 2.1 and 2.2 show that the *Primary* and *Secondary* systems share the same back-end evaluation pipeline: in both cases, the generated English text is synthesised with Coqui-TTS, transcribed with Whisper-HU, and compared to the original Hungarian input using WER, CER and PER. The essential difference lies in how the English text candidates are generated. In the *Primary* system, Hungarian text is first converted to HU IPA by a G2P module, then mapped to English IPA with EmPhon and PanPhon, and a beam search over a phoneme-annotated CMUdict lexicon selects the best matching word sequence. In contrast, the *Secondary* system first performs morphological segmentation of the Hungarian text and then uses a Hungarian morph n-gram–to–English dictionary plus beam search to construct an English word sequence directly, relying on morphology-based lexical units rather than purely phoneme-based search.

2.1 Component overview

emPhon for HU grapheme→phoneme (G2P)

Hungarian text is converted to IPA using **emPhon** [8], a morphology-aware G2P for Hungarian that encodes language-specific rules (e.g. long/short vowels, palatals and productive assimilations) and segments at morpheme/compound boundaries before applying phonology. This substantially reduces classic Hungarian (HU) pitfalls (e.g. affix-boundary effects, degemination and *h* allophones) that generic G2Ps miss. The authors of the paper report low intrinsic errors in lexicon tests (WER in low single digits) and consistent behavior in extrinsic TTS checks.

PanPhon for HU→EN phone mapping

To map HU phones to EN proxies, this work uses **PanPhon** articulatory feature vectors and distances [9]. This makes it possible to select English substitutes that are maximally similar in features (height, backness, roundness, place/manner/voicing, length), rather than relying on ad-hoc rules. A small set of alternatives is kept for each Hungarian phone (e.g., /y/ [y] ↦ [juː] or [uː]; /ö/ [ø] ↦ [ɜː] or [oʊ]), and the search stage then chooses the globally best sequence. PanPhon also enables the implementation of rules and overrides when consistently high error rates are observed for specific edge cases.

Constrained search over CMUdict (Beam search / WFST)

A fixed English lexicon (**CMUdict** [4]) is searched for a word sequence whose concatenated pronunciation best matches the mapped EN IPA. Two equivalent views will be implemented: (i) dynamic-programming beam search decoder, as standard in ASR [10], that extends phone-prefix matches with dictionary entries, and (ii) a *weighted finite-state* composition where a phone-substitution lattice (costs from PanPhon) composes with a phones→words lexicon FST; where the shortest path gives the solution. [11, 12]. Vocabulary is restricted to common words to improve TTS clarity and avoid degenerate outputs.

Coqui-TTS for EN speech synthesis

The English text is synthesised using **Coqui-TTS** [13] pretrained English voices from the neural Tacotron/VITS family. Reported English Mean Opinion Score (MOS) values are typically around ∼4.3–4.5 (on a scale of 1–5), which is sufficient to expose fine phonetic detail to ASR. The synthesis parameters are kept fixed across experiments for fairness.

Whisper-HU for ASR evaluation

The synthetic audio is transcribed with **Whisper** (large-v2, Hungarian mode) [5]. Whisper is a multilingual transformer ASR trained on ∼680k hours; it is robust to accent and pronunciation variation, which is crucial here because my audio is English speech intended to be heard as Hungarian. It achieves about 17–18% WER for Hungarian input.

HU morph n-grams → EN word dictionary

Goal. Build an English word dictionary indexed by Hungarian *morph* contexts (rather than raw phones), so decoding can operate on higher-level units that are more robust to assimilation and other phonological alternations.

Creation pipeline

1. **Morph segmentation.** Segment Hungarian text into morphs with `emMorph` [14]. Let a segmented sequence be $m_1, m_2, \ldots, m_T$.

2. **n-gram extraction.** Extract uni-, bi-, and tri-grams of morphs:

$$\mathcal{G}_1 = \{m_t\}, \quad \mathcal{G}_2 = \{(m_{t-1}, m_t)\}, \quad \mathcal{G}_3 = \{(m_{t-2}, m_{t-1}, m_t)\}.$$

3. **Constrained search over CMUdic.** For each $g \in \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3$, compute the best-matching English word by beam search

Rationale. Moving from phones to morph n-grams reduces spurious matches caused by local alternations. The overlapping context in bi-/tri-grams constrains phone patterns, which tightens the search space even though we still score at the phone level.

Use in decoding. Decoding is a global least-cost tiling over the HU morph sequence using the precomputed morph n-gram → English word dictionary. As we slide a window over morphs, an n-gram $g = (m_{t-n+1} \dots m_t)$ is admissible iff g exists in the dictionary; its English word $w(g)$ and cost $c_{\text{dict}}(g)$ are taken directly from the dictionary.

We find the optimal segmentation with a Viterbi-style DP:

$$F(t) = \min_{n \in \{1,2,3\} : g \in \mathcal{D}} \big[F(t-n) + c_{\text{dict}}(g)\big], \qquad F(0) = 0,$$

where $\mathcal{D}$ is the dictionary and $g = (m_{t-n+1} \dots m_t)$. Backpointers yield the lowest-cost sequence of English words $w(g)$ covering the input.

Backoff. If a tri-gram is absent, the DP naturally backs off to bi-/uni-grams present in $\mathcal{D}$.

2.2 Initial HU→EN phone map (used by the search)

Table 2.1 shows the compact map initialized based on phonological characteristics. The method will be further refined on PanPhon feature distances and empirical test results. The English side lists *preferred* and (when useful) *fallback* realizations that English TTS is likely to produce clearly.

Table 2.1: Seed HU→EN phone correspondences (TIPA). Preferences may be swapped by the search if a lower global cost is found.

HU target	EN proxy	Notes
/y, yː/ (ü/üː)	/juː/, /uː/	Front-rounded ∉ EN; /juː/ often closest perceptually; /uː/ keeps rounding/length.
/ö, öː/ (ö/öː)	/ɜː/, /oʊ/	Rounding sacrificed in /ɜː/; /oʊ/ preserves rounding but is backer.
/i, iː/	/ɪ, iː/	Lax vs. tense approximates short–long.
/u, uː/	/ʊ, uː/	Lax vs. tense approximates short–long.
/o, oː/	/[ʕ] or ɔ, oʊ/	Dialect-safe approximations; long often diphthongal.
/c/ (ty)	/tʃ/, /tj/	EN lacks palatal stop; affricate /tʃ/ is robust in TTS.
/ɟ/ (gy)	/dʒ/, /dj/	Voiced postalveolar affricate as proxy for palatal stop.
/ɲ/ (ny)	/nj/	Onset or coda environments tested in evaluation.
/s/ vs. /ʃ/	/s/ vs. /ʃ/	Orthography steers TTS: “sz” → /s/, “s/sh” → /ʃ/.

2.3 Constrained phone→word search

Let P_{HU} be the emPhon IPA for x, and P_{EN} the mapped English IPA sequence. English word sequence $w_{1..n}$ is sought from a fixed lexicon $\mathcal{L}$ (CMUdict) whose pronunciation best matches P_{EN} while staying linguistically reasonable. The objective can be formalized as:

$$\min_{w_{1..n}\in\mathcal{L}} \mathrm{Cost}\big(\mathrm{phones}(w_{1..n}),\, P_{\mathrm{EN}}\big) + \lambda\,\mathrm{LM}(w_{1..n}) \tag{2.1}$$

where phones$(\cdot)$ concatenates CMUdict pronunciations, Cost is a feature-weighted phone edit distance derived from PanPhon (0 for identical phones, small penalties for near-neighbors), and LM is a light length/rarity penalty to avoid degenerate outputs. This paper implements Eq. 2.1 in two ways:

1. **DP beam search:** grow phone-prefix matches with dictionary entries; keep top-K partial paths.
2. **WFST composition:** compose a phone-substitution lattice (PanPhon costs) with a phones→words lexicon FST and extract the shortest path [11, 12].

Both produce the same candidate set in practice; the one that is simpler to run on the machine will be used.

Dynamic-programming beam search decoding. Given the mapped English IPA target P_{EN}, decoding is implemented as Viterbi-style dynamic programming with *beam pruning*. At each phone prefix $P_{\mathrm{EN}}[1\!:\!t]$, each partial hypotheses is extended by any CMUdict entry whose pronunciation matches (exactly or with a small PanPhon-weighted substitution penalty), caching best costs per end position. Beam pruning keeps only the top-K partial paths per t, which yields near-optimal solutions while keeping complexity manageable [10]. This follows the standard ASR decoding pattern. In practice, a light word penalty and a rarity penalty (the $\lambda\,\mathrm{LM}$ term) are also included so the search prefers shorter, common-word sequences when phonetic costs are tied.

Weighted finite-state composition. Decoding is also realized with *weighted finite-state transducers* (WFSTs): a phone-substitution lattice (encoding PanPhon-based costs) is composed with a phones → words lexicon transducer derived from CMUdict; the best English sentence is then the *shortest path* under the tropical semiring. This

WFST view cleanly factorizes the problem (phone similarity vs. lexicon constraints), admits classic optimizations (determinization, minimization, weight pushing), and makes the objective in Eq. 2.1 an instance of standard shortest-path decoding [11]. These grammars is compiled with `Pynini` [12] on top of OpenFst, which provides stable, reproducible composition and shortest-path routines.

2.4 Synthesis and recognition

The chosen English sentence is synthesized with **Coqui-TTS** (single English voice, fixed settings) [13]. Then the audio is transcribed with **Whisper** in Hungarian to get the output text y (see 2.1.) [5]. Keeping both tools fixed ensures that differences across methods come from the *search/mapping* decisions, not from synthesis or recognition variability.

2.5 LLM Baseline: Gemini API (Developer API)

Goal and scope. As a lightweight baseline, a general-purpose large language model (LLM) is queried to directly propose an English phrase that *sounds like* the given Hungarian sentence when read aloud. This paper uses Google's Gemini Developer API via the `google-genai` Python SDK [15, 16]. Deliberately simple (no heavy prompt engineering) prompting is used and this baseline is compared against the modular pipeline using exactly the same TTS and HU ASR for evaluation (WER/CER, plus PER).

Prompting setup (zero/one/few-shot). Gemini's documented prompting patterns are followed for text generation: a *zero-shot* request consists only of the task and the Hungarian input; a *one-shot* request appends one worked example; a *few-shot* request appends k short input→output pairs (here: HU→EN homophonic examples) before the test input [16]. A concise *system instruction* is attached to fix behaviour ("perform homophonic translation; prioritize phonetic similarity over meaning; prefer common English words; avoid punctuation; output just the English phrase"). Safety settings remain at defaults (no special policy tuning). The experiments use the Gemini 2.5 Flash model [16].

2.6 Implementation notes

All components are open and local: emPhon for HU G2P [8], PanPhon for features [9], CMUdict for pronunciations [4], Pynini/OpenFst for WFSTs [12, 11], Coqui-TTS for synthesis [13], and Whisper for ASR [5]. A project manifest pins all component versions and exports the mapping table and lexicon snapshot alongside the results for full reproducibility.

Chapter 3

Implementation

This chapter documents how the system is built and executed so that the results in Chapter 4 can be reproduced. It focuses on the repository layout, core modules, the decoding/search implementations (DP-Phone and Morph-Dict), the LLM baseline integration, the fixed TTS+ASR harness, configuration knobs, logging, and safeguards. The chapter is intentionally framework-oriented: it specifies *what varies* (decoders, lexicon constraints, IPA map) and *what is held fixed* (TTS+ASR pipeline and evaluation procedure).

For brevity, the following short names are used throughout this paper:

- **DP-Phone** – the dynamic-programming phone/character decoder (Section 3.4), used as the primary system.
- **Morph-Dict** – the morph n-gram dictionary decoder (Section 3.5), used as the secondary system.
- **Gemini-LLM** – the Gemini-based large language model baseline (Section 3.6).

3.1 Design Goals and Invariants

- **Single evaluation harness.** All systems (DP-Phone, Morph-Dict, Gemini-LLM baseline) run inside the same $TTS \rightarrow ASR$ loop; only the decoder producing English text varies.
- **Strict comparability.** The TTS voice, ASR model, and scoring metrics are identical across runs (implemented in `tts_asr.py`).
- **Transparent constraints.** Lexicon slices (Zipf$\geq$4.5, Zipf$\geq$5.0, top-8k) are explicit inputs to the DP-Phone decoder.

- **Map isolation.** v1 vs. v2 experiments differ *only* in the HU→EN IPA proxy map file.
- **Decoding time policy.** All reported timings in Chapter 4 are *decoding/search only*; TTS and ASR are excluded by design.

3.2 Repository Layout and Build

```
src/
  g2p_hu.py              # Hungarian grapheme-to-IPA pre-processing
  phone_mapping.py       # Shared helpers for IPA manipulation
  ipa_map_v1.py          # HU→EN IPA proxy map (variant 1)
  ipa_map_v2.py          # HU→EN IPA proxy map (variant 2)
  search_dp.py           # DP/beam decoder over constrained lexicon
  llm_baseline.py        # Gemini 2.5 Flash baseline (single pass)
  tts_asr.py             # Fixed English TTS + Hungarian ASR harness
  metrics.py             # WER, CER, PER + alignments; CSV writers
  runtime_translate.py   # Timing hooks for decoding only
  transcriber.py         # ASR wrapper (called inside tts_asr.py)

data_v1/
  results/               # Per-sentence WER/CER/PER (v1 runs)
data_v2/
  results/               # Per-sentence WER/CER/PER (v2 runs)
aggregation/             # Per-class aggregations, summaries
archive/                 # Negative or resource-bounded runs (e.g., WFST)
```

3.3 Core Modules and Responsibilities

`g2p_hu.py`. Deterministic grapheme-to-phoneme pre-processing for Hungarian using emPhon; produces HU IPA strings suitable for mapping.

`ipa_map_v1.py` **vs.** `ipa_map_v2.py`. HU→EN IPA proxy maps. v2 tightens the treatment of front-rounded vowels and palatals/sibilants; in all experiments, *map choice* is the sole difference between v1 and v2 runs.

`search_dp.py`. Primary decoder: dynamic programming / beam search over a constrained English lexicon (CMU-derived). It consumes (i) the mapped EN IPA target sequence and (ii) lexicon constraints (Zipf≥4.5, Zipf≥5.0, top-8k), and returns an English word sequence plus search statistics.

In the CMUdict/wordfreq context, a word's *Zipf* value is a logarithmic frequency scale defined as

$$\mathrm{Zipf}(w) = \log_{10}\big(\text{occurrences of } w \text{ per } 10^9 \text{ tokens}\big).$$

Intuitively, higher Zipf means a more common word in English. For example, Zipf = 6 corresponds to about once per 10^3 words, while Zipf = 3 is about once per 10^6 words. Typical values range roughly from 1–7 (1–3 = rare; 4–7 = common), providing a human-friendly way to compare frequencies across corpora.

In CMUdict, Zipf > 4.5 corresponds to roughly the 3,196 most frequent words, while Zipf > 5 captures about the top 1,146. The top-8k test set is approximately equivalent to Zipf > 3.95.

`llm_baseline.py`. Minimal LLM baseline using `gemini-2.5-flash`, zero thinking budget and a *single pass* per sentenc. Post-processing normalizes punctuation/casing to keep the TTS behavior stable.

`tts_asr.py`. Invariant harness for all runs: a fixed English TTS voice and a fixed HU ASR.

`metrics.py`. Computes WER, CER, and PER based on the levenshtein distance.

3.3.1 Test configurations used in Chapter 4

All quantitative results in Chapter 4 are obtained by varying a small set of well-defined configuration knobs on top of the fixed architecture sketched above. For clarity, the final test configurations are summarised here.

Axes of variation. The following configuration dimensions are explored systematically in Chapter 4:

- **IPA map variant (v1 vs. v2).** Both DP-Phone and Morph-Dict can operate with either `ipa_map_v1.py` or `ipa_map_v2.py`. As described in Section 2.2, v2 tight-

ens the treatment of front-rounded vowels and palatals/sibilants, while v1 is more compact.

- **Lexicon slice for DP-Phone.** The DP-Phone decoder always works with a closed vocabulary derived from CMUdict and wordfreq. Three slices are used:
  - Zipf$\geq$ 4.5 (roughly 3.2k word forms),
  - Zipf$\geq$ 5.0 (roughly 1.1k word forms),
  - a frequency-based "top–8k" list (an exactly 8k lexicon).

  These settings trade off coverage and search complexity: higher thresholds favour very common, readable words but risk excluding good homophonic matches, whereas larger lexicons admit more candidates at the cost of a harder search.

- **Morph-Dict dictionary variants.** For the morph-based decoder considered:
  - plain morph-based dictionaries (no syllabification) with n-gram frequency order $m > 3$,
  - versions with additional grapheme-level syllabification of long stems

- **Gemini-LLM prompting policy.** The LLM baseline uses `gemini-2.5-flash` with zero thinking budget and exactly one generation per sentence (no retries or sampling sweeps), guided by a minimal prompt that instructs it to perform "homophonic translation" using the most common English words (Section 3.6).

Fixed hyperparameters. To keep comparisons clean, several lower-level parameters are held constant across all runs:

- **DP-Phone.** Uses a single beam width (top-K partial paths per prefix) and a single set of cost weights (α, β, γ) for substitution, insertion/deletion, and boundary penalties as defined in Section 3.4.2. These values are chosen once and then reused for all DP-Phone experiments.

- **Morph-Dict.** Uses fixed weights $(\lambda_\Delta, \lambda_z, \lambda_r)$ in its scoring function (Section 3.5.6) and a fixed maximum tile length for the tiling decoder (Section 3.5.7); experiments only toggle the dictionary variants listed above.

- **Evaluation harness.** The TTS and ASR components, along with the WER/CER/PER computation, are completely invariant and shared by all systems (Sections 2.4 and 3.7).

3.4 Primary DP-Based Character-Level Decoder

3.4.1 Search Space and Constraints

The DP-Phone decoder searches over sequences of English words drawn from a *constrained* lexicon. The size is either given exactly (top-8k), or by zipf (see section 3.3) value. Zipf scale is used because it provides an intuitive cutoff for filtering out internet acronyms and slang, which typically cluster around Zipf ≈ 4.3.

Multiple pronunciations per word are allowed; homographs expand the search but are disambiguated by the cost function. Unknowns are disallowed (closed vocabulary per setting).

3.4.2 Cost Function

Let $\phi(\cdot)$ be the EN IPA sequence produced from the HU input by the chosen map (v1 or v2). Let $\psi(\cdot)$ be the IPA sequence of a candidate English word sequence from the lexicon. The decoder minimizes a weighted alignment cost:

$$\mathrm{Cost}(\phi, \psi) = \alpha \cdot \mathrm{Sub}(\phi, \psi) + \beta \cdot \mathrm{InsDel}(\phi, \psi) + \gamma \cdot \mathrm{Bound}(\psi),$$

where:

- Sub is a feature-based phonetic substitution cost (e.g., derived from segmental feature distances for place/manner/voicing, vowel height/backness/rounding, and length).
- InsDel penalizes insertions/deletions.
- Bound adds small regularization for word boundaries (discouraging excessive splitting).

Hyperparameters α, β, γ are fixed across runs; v1 vs. v2 affects only ϕ via the map.

3.4.3 Beam Search (Sketch)

```
function DP_BEAM(hu_ipa, map, lexicon, beam_width):
    target = MAP_TO_EN_IPA(hu_ipa, map)       # v1 or v2
    agenda = [(0, BOS, empty_hyp)]            # (score, state, partial
    ↪ hypothesis)
    for t in 1..|target|:
        next_agenda = []
        for (score, state, hyp) in TOPK(agenda, beam_width):
            for word in LEXICON_EXPAND(state, lexicon):
                for pron in PRONUNCIATIONS(word):
                    s = ALIGN_COST_STEP(target[t], pron, state)
                    push(next_agenda, (score + s, NEWSTATE(state, pron),
                    ↪ hyp+[word]))
        agenda = PRUNE(next_agenda, beam_width)
    return BEST(agenda)
```

The implementation caches alignment micro-costs and prunes by beam width.

3.5 Morph n-gram Decoder

3.5.1 Rationale and overview

The morph n-gram decoder translates Hungarian text into English word sequences by tiling short Hungarian units with English words selected to be phonetically similar when read aloud. Unlike the phone/character DP-Phone decoder, which aligns full sentence-level phone strings, this approach operates at the level of local units to encourage compositionality and speed. In practice, this decoder is computationally light and straightforward to run.

3.5.2 Morphological segmentation and its limitation

emMorph is used as a morphological analyzer to obtain the finest available segmentation of each Hungarian token. Even in this setting, many stems remain unsegmented, especially longer or morphologically opaque roots. As a result, the dictionary contains not only sub-word units but also many full-word units, so a large portion of words cannot be

reconstructed from shorter parts. This undermines the intended compositionality of the method.

3.5.3 Syllabification refinement

To introduce additional structure inside long stems, a grapheme-level syllabification pass was also evaluated. Each syllable is defined by a single vowel nucleus drawn from the Hungarian vowel inventory; onsets are attached maximally and codas are restricted to short clusters to avoid pathological splits. Only morphs exceeding a length threshold are syllabified; very short items are left intact. Single-character syllables created by this pass are merged back into neighbors to prevent degenerate fragments. Despite this refinement, single-character morphs originating from the analyzer may still occur and propagate. The empirical impact of this syllabification refinement, compared to a version without it, is evaluated in Table 4.1 and analysed in Chapter 5.

3.5.4 Unit inventory and n-gram statistics

After refinement, unit sequences are collected and uni-, bi-, and tri-gram statistics are computed in the Hungarian text. Frequent units form the keys of a mapping table. Syllabification substantially increases the number of unit types ; the inventory becomes large because many near-duplicate sequences differ only in local boundaries. This growth later manifests itself as a large (around 60k enties) Hungarian-to-English mapping table and longer preprocessing times. Even after pruning units that occur at most three times ($n \leq 3$), the map will contain around 8k entries. This variant slightly improved performance (see table 4.1).

3.5.5 English candidate generation and filtering

For each Hungarian unit (length 1–3), candidate English words are drawn from a pronunciation dictionary and filtered to improve the robustness and readability of the synthesis:

- frequency filter to exclude rare words (Zipf threshold in the low-to-mid 4 range);
- orthographic filter to remove symbols and non--alphabetic forms;
- phonotactic filter that prefers common English onsets and codas and limits word length and syllable count according to the unit length.

A small class of interjections and chat-style tokens is excluded because they degrade perceived quality. Their orthography and pronunciation are highly inconsistent (for example, English users may realise "LOL" as a spelled-out sequence or as a single syllable, while Hungarian users often say "eloel"), so they do not form stable, predictable units for synthesis. Any apparent improvement from including such items would therefore be mostly accidental, driven by the fact that contemporary models are heavily trained on internet slang rather than by systematic pronunciation modelling. These guards materially reduce problematic output, but cannot fully protect against the selection of one-letter English tokens when the Hungarian unit itself is a single character.

3.5.6 Scoring objective

Let ϕ denote the target IPA sequence for a Hungarian unit and ψ denote the candidate English pronunciation. The mapping cost is

$$\mathcal{L}(\phi, \psi) = \mathcal{A}(\phi, \psi) + \lambda_\Delta \, | \, |\phi| - |\psi| \, | + \lambda_z \, \max\big(0, \tau - z(\psi)\big) + \lambda_r \, \rho(\psi),$$

where $\mathcal{A}$ is an alignment cost based on phonological feature differences, the second term penalizes length mismatches, the third term penalizes low-frequency English words below a Zipf threshold τ, and ρ is a readability prior that down-weights forms known to reduce fluency (for example, one-letter tokens outside tightly constrained contexts). The weights $(\lambda_\Delta, \lambda_z, \lambda_r)$ are fixed in all experiments. This scoring produces, for each Hungarian unit, a small set of English candidates ranked by $\mathcal{L}$.

3.5.7 Tiling decoder

Given a sequence of refined units $m_{1:N}$, decoding seeks a sequence of English words by dynamic programming over tile lengths $k \in \{1, 2, 3\}$. At position j, the decoder considers the span $m_{j:j+k-1}$ when available, retrieves the precomputed cost or computes it on demand, and updates a best-first chart. A small step penalty encourages covering the sequence with longer tiles where possible. The final English output is obtained by backtracking from the last position. This procedure is linear in the number of positions up to a constant factor determined by the maximum tile length and the cost of candidate retrieval.

3.5.8 Practical settings

Experiments used uni-, bi-, and tri-gram units; a Zipf threshold in the low-to-mid 4 range for English candidates; phonotactic constraints that limit unusual clusters and excessively long words; and a moderate penalty for length mismatch. Decoding times reported in the experiments chapter refer to the tiling step only and exclude synthesis and recognition.

3.6 LLM Baseline Integration

Model and policy. The baseline uses `gemini-2.5-flash` with *zero thinking budget* and exactly *one* generation per sentence (no retries or sampling sweeps). This keeps runtime low.

Used Prompt A minimal prompt that emphasises phonetic resemblance without inviting free-form storytelling:

```
SYSTEM: "Perform homophonic translation from hungarian to english. Meaning
    give me common english word sqequence which spoken aloud sounds exactly
    like the hungarian sentence I gave you. Please use the most common
    10000 english words. I will give you input sentences one by one, just
    answer with the english word sqequence. No comma, nothing else:"

USER: "<HU sentence here>"
```

Outputs are normalized (case, punctuation) before TTS to stabilize synthesis. The model performed perfectly with zero-shot prompting; accordingly, one- and two-shot variants did not yield better outcomes in our small experiments.

3.7 Invariant TTS + ASR Harness

Fixed across all runs. The synthesis (**EN TTS**) and recognition (**HU ASR**) stack is *identical* for DP, Morph, and Gemini evaluations. Audio parameters (sample rate, channels) are kept constant (default parameters).

Known TTS failure mode. After certain trigger sentences—most often ones ending in '-s'—the TTS can become 'stuck' and emit between 15 seconds and 2 minutes of meaningless audio. This is reproducible for the same input. This behavior is external to the decoding logic and does not affect the comparability of decoders; Chapter 4 reports results only from valid (non-stuck) or cleaned (manually trimmed the stuck part off) syntheses.

3.8 Extensibility Notes

- **Maps.** New HU→EN proxy sets (or other language pairs) can be added as separate `ipa_map_.py` files, with the rest of the pipeline unchanged.
- **Lexicon slices.** Frequency thresholds and curated wordlists can be composed to test readability vs. fidelity trade-offs.

3.9 Pilot calibration of the TTS→ASR evaluation loop

The TTS→ASR evaluation harness described above is deliberately kept fixed across all experiments so that differences in WER, CER and PER can be attributed to the decoders rather than to synthesis or recognition variability. However, this setup also introduces its own noise: even if the English input is a very good homophonic translation, the English TTS and Hungarian ASR may still disagree with the intended Hungarian reference.

To characterise this noise more explicitly, a small pilot calibration will be carried out on two sentences that are widely recognised as high-quality Hungarian–English homophonic translations. For each pair (Hungarian reference sentence and its English homophonic counterpart), the English sentence is passed ten times through the same Coqui-TTS and Whisper-HU configuration as in the main experiments, and the resulting ASR outputs are scored (WER, CER, PER) against the original Hungarian reference.

This calibration does not evaluate any decoder: instead, it estimates how much error the evaluation pipeline itself introduces when the English side is already "gold" from a human perspective. The aggregate scores are reported in Section 4.3 and later used in Chapter 5 to interpret what level of WER/CER/PER can still be considered practically usable.

Summary

This implementation enforces a clean separation between the variable decoding module (DP-Phone, Morph-Dict, Gemini-LLM) and the fixed evaluation harness (TTS+ASR+metrics). Map choice (v1 vs. v2) and lexicon constraints are isolated, enabling the controlled comparisons reported in Chapter 4.

Chapter 4

Experiments and Results

This chapter evaluates the homophonic translation systems defined earlier on the updated hard-case Hungarian test set (Appendix A.0.1), which contains **28** items (n=28). It compares DP-Phone, Morph-Dict, Gemini-LMM[1] within the common TTS→ASR loop introduced in Chapters 1– 2.

4.1 Data and Protocol

Test items. The dataset was intentionally curated to pack in challenging edge cases. As a result, translating ordinary sentences should perform better than what the test set suggests. Note that the ratio/distribution of phone types was not normalized to their frequencies in natural language.

Systems evaluated

- **DP-Phone (primary)**: Hungarian IPA → mapped EN IPA → constrained search over CMUdict to English words; beam decoder with PanPhon-derived costs.
- **Morph-Dict (secondary)**: emMorph segmentation; uni/bi/tri-gram HU morph contexts mapped to English words via precomputation; Viterbi tiling during decoding.
- **Gemini-LLM (baseline)**: prompt to produce an English phrase that, when read aloud, resembles the HU input; evaluated with the same TTS/ASR loop.

[1] See chapter 3 for these short names

Real-life sentence fragments

To complement the deliberately hard phonetic test set, the systems are also evaluated on a small collection of more natural sentences. This real-life set consists of fifteen fragments sampled from Hungarian online news articles and event reports; the full list of sentences is provided in Appendix A.0.2.

In contrast to the hard-case set, these sentences are not phoneme-balanced and were not designed to target specific phonological contrasts. They reflect typical news-style usage, including multiword named entities (such as *Nemzetközi Labdarúgó Szövetség* and *Gyenes Gábor*) and compound noun phrases. All systems are evaluated in this set using exactly the same TTS→ASR protocol as for the hard-case sentences, and the same phoneme-class aggregation is applied a posteriori based on the classes present in each sentence.

4.2 Main Results

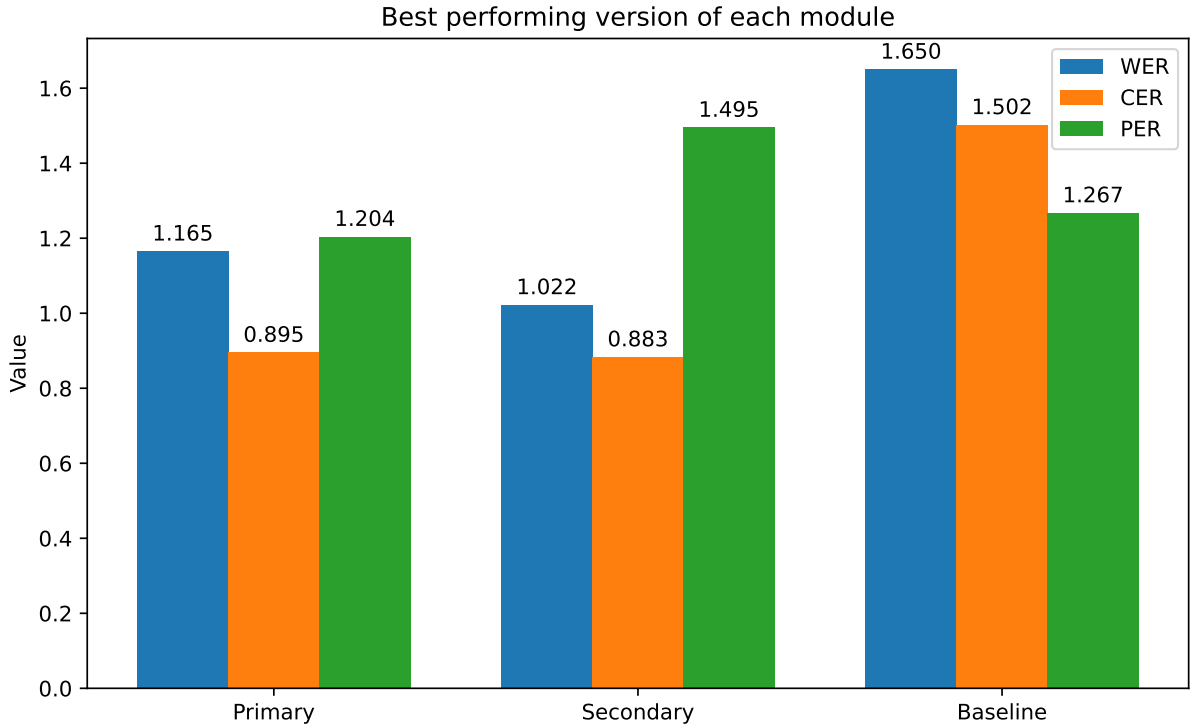

Figure 4.1: Bar chart of the best performing variant of each module. (see figure 1.1 for modules)

Table 4.1 gives aggregate scores (means over all items). Lower is better.

Table 4.1: Aggregate HU ASR error rates on the TTS→ASR loop for DP-Phone, Morph-Dict and Gemini-LLM (`gemini-2.5-flash`, averaged over multiple passes). For setting name explanation see chapter 3. For Zipf meaning clarification see section 3.3.

Setting	**WER**	**CER**	**PER**
DP-Phone Zipf ≥4.5 (v1)	1.150	0.886	1.275
DP-Phone Zipf ≥5.0 (v1)	1.380	1.155	1.343
DP-Phone Top-8k (v1)	1.165	0.895	**1.204**
DP-Phone Zipf ≥4.5 (v2)	1.440	1.181	1.550
DP-Phone Zipf ≥5.0 (v2)	1.420	1.320	1.574
Morph-Dict (n>3) (v1)	1.225	1.053	1.535
Morph-Dict (n>3) (v2)	**1.022**	**0.883**	1.495
Morph-Dict syllabified (v1)	1.552	2.034	2.030
Morph-Dict syllabified (v2)	1.869	2.783	2.040
Morph-Dict syllabified (n>3) (v2)	1.595	2.4869	1.4849
Gemini-2.5-flash (averaged)	1.650	1.502	1.267

Takeaways. (i) v1 character-level search outperforms Gemini on WER and CER across lexicon settings (Table 4.1); (ii) Morph-Dict decoding achieves the best WER and CER but clearly higher PER than DP-Phone; (iii) v2's alternative phone map reduces average search cost but does not reduce error on this test bed.

Effect of Search Constraints (Lexicon Size) on Accuracy and Fluency

To study the effect of search constraints, the size of the decoding lexicon was varied via Zipf-based cutoffs and a frequency-based top-N list. In the DP-Phone decoder, Zipf ≥ 4.5 corresponds to a lexicon of roughly 3.5k word forms, Zipf ≥ 5.0 to roughly 1.1k, and the "top–8k" condition approximates a mid-sized 8k lexicon. Early experiments with a much larger 15k lexicon showed clearly degraded performance, most likely due to beam and pruning limits being too tight for the expanded search space.

For *v1*, the results indicate a non-monotonic relationship between lexicon size and error rates:

- A very small lexicon (Zipf ≥ 5.0, $\sim$1.1k items) yields the worst scores (WER = 1.380, CER = 1.155, PER = 1.343), suggesting that over-constraining the search harms recognition by removing many valid candidates.

- A mid-sized, frequency-constrained lexicon (top–8k) achieves the best PER (1.204) and competitive WER/CER (1.165 / 0.895), indicating an optimal trade-off between having enough lexical coverage and not overwhelming the search with low-frequency noise.

- The intermediate Zipf ≥ 4.5 setting (3.5k items) slightly improves WER (1.150) and CER (0.886) compared to top–8k, but at the cost of a higher PER (1.275), suggesting that word-level correctness and phoneme-level robustness respond differently to lexicon pruning.

For *v2*, tightening the lexicon from Zipf ≥ 4.5 to Zipf ≥ 5.0 does not produce systematic gains. WER changes only marginally (1.440 vs. 1.420), while CER and PER slightly worsen (1.181 vs. 1.320 and 1.550 vs. 1.574, respectively). This again points to diminishing returns—and even mild degradation—when the lexicon is pushed below a certain size threshold.

Taken together, these results suggest that search constraints via lexicon size exhibit a U-shaped effect on accuracy: both very large (15k) and very small ($\sim$1k) lexicons perform worse than an intermediate range (roughly 3.5k–8k), given fixed beam and pruning parameters. From a fluency perspective, restricting the lexicon to frequent words removes many rare or noisy items and therefore biases the decoder towards more typical, high-frequency realisations. However, if the lexicon becomes too small, this constraint forces systematic substitutions of rarer but correct words with more frequent, but incorrect, alternatives, which ultimately degrades both perceived fluency and objective error metrics.

Runtime (decoding only).

Figure 4.2 compares *search/decoding time per sentence* (DP-Phone vs Morph-Dict vs Gemini-LLM). Although character-level search achieved the best accuracy regarding *PER*, its decoding time was about two orders of magnitude longer than the dictionary-based approach and three orders longer than the LLM-based module.

4.3 Noise level of the TTS→ASR evaluator (gold sentences)

Before interpreting the system-level scores in Table 4.1, it is useful to quantify how much error comes from the evaluation pipeline itself. Even if the English input is a very good homophonic translation, the English TTS and Hungarian ASR may still diverge from the intended Hungarian reference in ways that inflate WER, CER and PER.

To estimate this evaluation noise, two reference-quality homophonic translations predating this work are used:

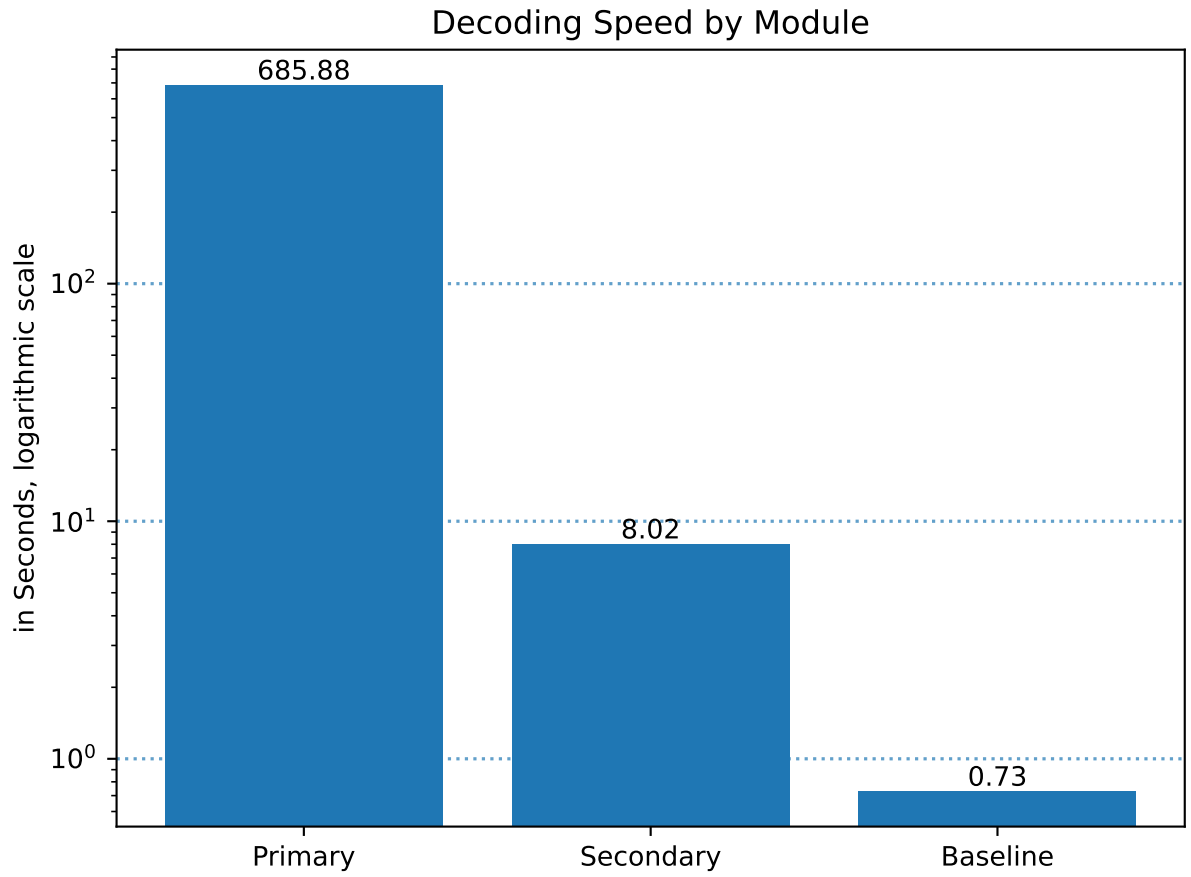

Figure 4.2: Averaged decoding (search) time per sentence (lower is better) visualised in logarithmic scale. TTS and ASR are excluded.

- HU: *Hol a nyuszi?* → EN: *hole a new see*
- HU: *Tépett varjú van a vason.* → EN: *tape at war you wanna wash on*

For each English sentence, the same Coqui-TTS + Whisper-HU pipeline was run ten times and scored the resulting ASR transcripts against the Hungarian reference with WER, CER and PER, exactly as in the main experiments. The scores below are the averages over the ten runs for each sentence:

- *Hol a nyuszi?* → *hole a new see*:

$$\text{WER} = 0.333, \quad \text{CER} = 0.167, \quad \text{PER} = 0.500.$$

  Even in this short and relatively simple case, the ASR makes errors at all three levels.

- *Tépett varjú van a vason.* → *tape at war you wanna wash on*:

$$\text{WER} = 1.340, \quad \text{CER} = 0.621, \quad \text{PER} = 0.774.$$

  Here, both word- and phone-level errors are clearly more pronounced, with WER exceeding 1.0 and PER close to 0.8, despite the English phrase being a widely known good homophonic translation.

Averaging over the two sentences (treating them equally) yields approximate "gold noise" scores of

$$\text{WER}_{\text{gold}} \approx 0.84, \quad \text{CER}_{\text{gold}} \approx 0.39, \quad \text{PER}_{\text{gold}} \approx 0.64.$$

These values can be read as a rough noise floor for this particular TTS→ASR configuration, where the evaluation loop still reports around 0.64 PER on average and can exceed 1.0 WER for more complex sentences. In other words, a substantial part of the observed error in Table 4.1 is due to the variability of the synthesis and recognition, not just to errors made by the decoders.

Relative to this baseline, the best DP-Phone configuration on the hard-case set (PER = 1.204), the best Morph-Dict configuration (PER = 1.495) and the Gemini baseline (PER = 1.267) are roughly 1.9×, 2.3× and 2.0× worse than the calibrated PER floor, respectively.[2] This does not make the systems "bad" in absolute terms—rather, it shows that on the intentionally hard test set, roughly half of the DP-Phone error and an even larger share of the Morph-Dict error can plausibly be attributed to the evaluation pipeline's own limitations (mismatched TTS vowels, ASR confusions on clusters, etc.), with the remainder reflecting genuine differences in how well each method preserves Hungarian pronunciation.

4.4 Performance on real-life sentences

To assess how the systems behave on more natural inputs, the best-performing configurations are also evaluated from Chapter 3 on the real-life sentence fragments described in Section 4.1. As before, the English outputs are passed through the Coqui-TTS and Whisper-HU pipeline and scored with WER, CER and PER against the original Hungarian sentences, then aggregated over the phoneme classes present in the set.

[2] All PER values are taken from the aggregate scores in Table 4.1.

System	WER	CER	PER
DP-Phone (top-8k, v1)	**1.1827**	**0.77**	**0.952**
Morph-Dict (v2)	1.1832	0.894	1.214
Gemini-LLM	1.994	0.838	1.034

Table 4.2: Average WER, CER and PER on the real-life test set of 15 news-style sentence fragments.

Table 4.2 reports the average WER, CER and PER obtained on this real-life set for the three systems considered: the DP-Phone decoder with the top-8k lexicon and IPA map v1, the Morph-Dict decoder with IPA map v2, and the Gemini-LLM baseline.

Overall error levels on the real-life set are somewhat lower than on the deliberately hard phonetic set (cf. Table 4.1), which is expected: the news fragments contain fewer extreme clusters and long artificial compounds. The DP-Phone decoder achieves the lowest average CER and PER (0.777 and 0.952, respectively), with a WER essentially identical to that of the Morph-Dict decoder. Gemini shows the highest WER by a large margin (≈ 2.0 versus ≈ 1.18 for the other systems), indicating more word-level deviations, but its PER on this easier set (1.034) lies between DP-Phone (best) and Morph-Dict (worst).

Per-class scores on the real-life set are shown in Table 4.3. On this small, unbalanced sample, DP-Phone attains the lowest PER in most phoneme classes (affricates, plain sibilants, voicing assimilation targets and vowel length, among others), ties with Gemini for the /z/ vs /zh/ contrast and is only slightly worse on front rounded vowels and sibilant clusters. Morph-Dict tends to lag behind DP-Phone in PER while occasionally outperforming it on WER for some consonant classes, and Gemini remains competitive only on a subset of contrasts despite its much higher WER.

4.5 Per-Class Error Analysis

The primary module is the most informative to analyze, as it achieves the lowest PER. Tables B.1 (v1) and B.2 (v2) show the result of representative phoneme classes.

4.5.1 Overall Difficulty Landscape

- **Hard families.** Classes involving *dense consonant structure* and *rich sibilant/affricate inventories* (e.g., `clusters`, `sibilants`, `affricate_cs`, `z_vs_zh`) tend to

	DP-Phone			Morph-Dict			Gemini-LLM		
Class	WER	CER	PER	WER	CER	PER	WER	CER	PER
affricate_cs	0.944	0.750	0.732	0.889	0.662	1.498	1.611	0.640	0.990
clusters	1.200	0.829	1.250	1.000	0.707	1.219	1.200	0.902	1.406
front_rounded	1.163	0.756	1.055	1.558	1.206	1.231	2.104	0.798	0.997
geminate	1.275	0.794	1.101	1.550	1.300	1.079	2.100	0.772	1.133
palatal_gy	1.250	0.797	1.026	1.833	1.625	1.088	1.875	0.764	1.138
palatal_ny	1.335	0.883	0.980	1.475	1.205	1.367	2.615	0.991	0.996
sibilant_clusters	1.400	0.720	1.095	1.600	0.920	1.571	2.400	0.760	1.000
sibilants	1.099	0.766	0.897	0.925	0.673	1.082	1.835	0.928	1.004
voicing_assimilation_target	1.344	0.694	0.828	0.952	0.714	0.940	1.900	0.700	1.033
vowel_length	1.208	0.788	0.967	1.220	0.927	1.262	2.053	0.855	1.054
z_vs_zh	0.833	0.629	0.750	0.667	0.429	0.536	1.167	0.600	0.750

Table 4.3: Per-class WER, CER and PER on the real-life test set for DP-Phone (top-8k, v1), Morph-Dict (v2) and the Gemini-LLM baseline.

exhibit elevated PER, indicating persistent phonemic ambiguity (see RQ3 in section 1.4).

- **Vowel complexity.** `front_rounded` vowels are systematically harder than most other vowels, reflecting limited cues and higher confusability in that subspace.
- **Easier cores.** Voicing cues and well-delimited lateral contrasts (e.g., `voicing_assimilation_target`, `laterals`) typically show lower PER, suggesting those categories are better captured by current representations/constraints.

4.5.2 Metric Behavior (WER, CER, PER)

- **Metric divergence.** Several classes display CER/PER moving differently from WER (i.e., character/phoneme confusions that do not always flip whole words). Thus, fine-grained errors can be masked at the word level.
- **Sensitivity.** PER is most sensitive to phonological ambiguity: where inventories are dense (sibilants, affricates, clusters), PER rises more clearly than WER, making it the most informative signal for per-class tuning.

4.5.3 Search Cost vs. Accuracy is Not Monotonic

- **Wide cost range.** Average search cost spans from single digits to the mid-20s, but low cost does not guarantee low error.

- **Decoupling.** Some classes (e.g., `vowel_harmony`) are relatively accurate yet expensive, while others remain error-prone at moderate cost. This suggests search depth alone is insufficient in certain phonological regions; better priors/constraints or class-specific modeling are more effective than uniform budget increases.

4.5.4 Granularity Matters

- **Subclasses vs. aggregates.** Within broad labels (e.g., "front_rounded", "affricates", "sibilants"), specific subclasses (e.g., `front_rounded_short` vs. `front_rounded_long`; `affricate_dzs` vs. `affricate_cs`) show distinct error profiles. Treating them separately (or conditioning on onset/coda/cluster position) yields a clearer picture than aggregate reporting.

- **Homogeneous micro-classes.** Cases like `geminate_dd` are simpler than the generic `geminate` class, implying that intra-class heterogeneity drives much of the difficulty.

4.5.5 Phonological Context Effects

- **Positional and combinatorial constraints.** Classes defined by onset/coda position or cluster formation expose different failure modes. Errors increase where legal continuations branch densely (e.g., multi-consonant `clusters`) and decrease where context provides strong cues (e.g., certain onsets/voicing patterns).

4.5.6 Practical Implications

Taken together, the per-class tables show that error rates are highly heterogeneous: classes with dense consonant structure or rich sibilant and affricate inventories tend to remain difficult across decoders, while homogeneous or context-rich classes are easier. Aggregated metrics can hide these differences, because improvements in some classes may be offset by regressions in others.

4.6 IPA Map v1 vs v2

Map v2 implements stricter rounding/length preferences for front-rounded vowels and more selective palatal/sibilant proxies. For exact weights used see table 4.4 and table 4.5.

Compared the two character-level decoders (*v1* and *v2*) across 35 phonological classes using word error rate (WER), character error rate (CER), phoneme error rate (PER), and average search cost. On average, *v2* reduces decoding cost substantially, at the expense of higher error:

$$\begin{aligned} \text{avg. cost} &: 10.77 \rightarrow 7.97 (\approx -26\%) \\ \text{WER} &: 1.18 \rightarrow 1.42 (\Delta = +0.24) \\ \text{CER} &: 0.85 \rightarrow 1.07 (\Delta = +0.22) \\ \text{PER} &: 1.24 \rightarrow 1.39 (\Delta = +0.16) \end{aligned}$$

The effects are not uniform across classes: in 20/35 classes, *v2* improves PER while also reducing cost, whereas in 15/35 classes PER increases, despite the lower cost. See tables B.1 (v1) and B.2 (v2).

Classes Where v2 Improves Accuracy and Cost

v2 achieves simultaneous PER reductions and cost reductions for several phonologically important classes:

- **Affricates / voicing contexts:** `affricate_dzs` (1.21 → 0.71), `affricate_cs_onset` (2.10 → 1.30), `voiced_coda_g` (2.10 → 1.30).
- **Palatals and rounded vowels (selected):** `palatal_gy` (1.74 → 1.24), `front_rounded_short` (1.92 → 1.50), `front_rounded_long` (1.68 → 1.37).
- **Other improved classes:** `s_vs_sh` (1.18 → 0.82), `long_u` and `palatal_gy_clusters` (both ≈ 1.56 → 1.22), `word_initial` (1.21 → 0.71).
- **High-cost class with better trade-off:** `vowel_harmony` maintains strong accuracy with lower PER (0.92 → 0.73) and markedly lower cost (24.52 → 15.77).

These gains suggest that the *v2*'s weights are particularly effective in contexts where local phonological constraints are strong (e.g., onsets, voicing alternations) and for some palatal and rounded-vowel contrasts.

Classes Where v2 Regresses

At the same time, *v2* exhibits substantial accuracy losses on a smaller set of classes:

- **Large regressions (PER):** `clusters` (1.45 → 4.44), `affricate_cs` (0.98 → 3.79), `sibilants` (1.26 → 2.73), `z_vs_zh` (1.00 → 1.83).
- **Moderate regressions:** `i_vs_long_i_rhyme`, `long_a`, `voicing_assimilation_target`, `affricates` (aggregate), `vowel_length`.

Common to most of these failure cases are dense consonant clusters and broad sibilant inventories. These regions are highly ambiguous locally and typically benefit from stronger lexicon priors or a deeper search.

Conclusion.

v2 improves accuracy for a majority of classes, particularly in onset/voicing and several palatal/rounded contexts. However, pronounced regressions on cluster-dense and sibilant classes argue for a hybrid, class-aware deployment and targeted modeling and search adjustments, rather than a wholesale replacement of *v1* by *v2*.

Source	Target 1 (cost)	Target 2 (cost)
y	juː (0.0)	uː (0.2)
yː	juː (0.0)	uː (0.2)
ʌ	ɜː (0.0)	oʊ (0.2)
ʌː	ɜː (0.0)	oʊ (0.2)
i	ɪ (0.0)	i (0.15)
iː	iː (0.0)	i (0.15)
u	ʊ (0.0)	u (0.15)
uː	uː (0.0)	u (0.15)
o	ɔ (0.0)	ɒ (0.15)
oː	oʊ (0.0)	ɔː (0.2)
c	tʃ (0.0)	tj (0.2)
ɟ͡ʝ	dʒ (0.0)	dj (0.2)
ɟ	nj (0.0)	
s	s (0.0)	
ʃ	ʃ (0.0)	

Table 4.4: IPA mapping rules for v1. Costs in parentheses.

4.7 Human Readability vs Phonetic Fidelity

Table 4.6 shows typical outputs for each system. DP-Phone outputs tend to use unusual English word combinations but closely follow the Hungarian phone sequences; Morph-dict outputs use fewer distinct words and somewhat shorter phrases; Gemini produces fluent English-like text that often departs further from the intended pronunciation.

Source	Target 1 (cost)	Target 2 (cost)	Target 3 (cost)	Target 4 (cost)
y	jʊ (0.0)	juː (0.15)	ʊ (0.25)	u (0.35)
yː	juː (0.0)	uː (0.15)	jʊ (0.25)	
ʌ	oʊ (0.0)	ɔ (0.15)	ɜː (0.40)	
ʌː	oʊ (0.0)	ɔː (0.15)	ɜː (0.45)	
u	ʊ (0.0)	u (0.20)		
uː	uː (0.0)	ʊ (0.20)		
o	ɔ (0.0)	oʊ (0.25)		
oː	oʊ (0.0)	ɔː (0.15)		
i	ɪ (0.0)	i (0.15)		
iː	iː (0.0)	i (0.10)		
ɛ	ɛ (0.0)	e (0.20)		
eː	eɪ (0.0)	iː (0.35)		
ɒ	ɑ (0.0)	ʌ (0.25)		
aː	ɑː (0.0)	aʊ (0.35)		
c	tʃ (0.0)	tj (0.25)		
ɟ͡ʝ	dʒ (0.0)	dj (0.25)		
ɟ	nj (0.0)	n (0.60)		
s	s (0.0)			
ʃ	ʃ (0.0)			
z	z (0.0)	ʒ (0.45)		
ʒ	ʒ (0.0)	z (0.45)		

Table 4.5: IPA mapping rules for v2. Costs in parentheses.

Table 4.6: Sample HU input with DP-Phone (8k, v1) vs Morph-Dict (v2) vs Gemini-LLM outputs

HU	DP-Phone (EN words)	Morph-Dict (EN words)	Gemini-LLM (EN words)
Húzd ki a Szent Biztosítószeget	His key sent beast aw shit said et	His skills sent districts said et	Who's ki a saint bees toast eat all say get
majd azután számlálj el háromig	mm you'd zoo turn small l horse me a	members transformed l hmm e	my doers ooh tan som lie ill harm ig
se többet se kevesebbet	et oh eh tech ever eh share et	share toe get share ev 's et	Say to bet say give a sip bet

4.8 Secondary System: Morph n-gram Dictionary

The morph-dictionary based approach slightly underperforms DP-Phone across metrics (Table 4.1). Two practical issues match our observations:

- **Long morphs**: emMorph often returns long morphs identical to the surface form for compounds/opaque stems, so a morph→English word dictionary still faces full-word complexity (e.g., *fekete*, *bika*, *pata*, ...), defeating the expected simplification.
- **One-character morphs**: 1-character morphs (*a*/*é*/*n*, etc.) induce degenerate tilings and noisy choices.

4.9 Reproducible TTS Failure Mode

Beyond ordinary phone-level realizations, we observed a deterministic TTS failure mode: *after some sentences the synthesizer becomes "stuck" and hallucinates meaningless audio for up to ∼2 minutes*. For the same trigger sentence, it produced the same long, nonsensical waveform every time. These cases are catalogued as reproducible TTS bugs; they do not stem from the decoding logic.

Chapter 5

Discussion

This chapter interprets the empirical results from Chapter 4 in light of the research questions posed in Section 1.4. Whereas Chapters 2–3 described the system design and implementation, and Chapter 4 reported quantitative outcomes and error patterns, the focus here is on what those numbers mean: how the different approaches compare to the LLM baseline, which phonological contrasts remain challenging, how far the current systems are from being practically usable, and how the framework could extend beyond Hungarian.

5.1 Summary of main findings

Across both the deliberately hard phonetic test set and the smaller real-life set of news-style fragments, the dynamic-programming phone/character decoder (DP-Phone) emerges as the most reliable option when the objective is to preserve Hungarian pronunciation. On the hard-case set (Table 4.1), the best DP configuration (top-8k, v1) achieves the lowest phoneme error rate (PER = 1.204) among all systems, while maintaining competitive word and character error rates (WER = 1.165, CER = 0.895). The Morph-Dict decoder attains the best WER and CER (1.022 and 0.883, respectively, with v2), but its PER is consistently higher (1.495), indicating that many of its word-level "wins" are achieved by sacrificing phone-level fidelity. The Gemini-LLM baseline sits between the two decoders in PER (1.267) but yields the highest WER and CER (1.650 and 1.502).

On the real-life news-style set (Table 4.2), overall error levels drop for all systems. DP-Phone again attains the lowest CER and PER (0.777 and 0.952), with Morph-Dict

worse on both metrics (0.894 and 1.214) and Gemini in between (0.838 and 1.034) but with nearly double the WER of the decoders (1.994 vs. ≈1.18).

The per-class breakdowns (Tables B.1- B.8 and Table 4.3) show that phoneme classes involving front-rounded vowels, dense sibilant/affricate inventories and consonant clusters dominate the error profile. In these regions, DP-Phone usually outperforms the other systems but still exhibits elevated PER.

Regarding the HU→EN phone map variants, v2 substantially reduces average decoding cost compared to v1 (from 10.77 to 7.97) but at the price of modestly higher aggregate error (WER, CER, PER; Section 4.6). The gains are unevenly distributed across classes: v2 improves several palatal and rounded-vowel contrasts while regressing on cluster-rich and sibilant-heavy classes. This supports the view that phone-cost tuning is beneficial but must be combined with class-aware search budgeting rather than treated as a single global hyperparameter.

Finally, runtime results (Figure 4.2) confirm the expected trade-offs: DP-Phone is by far the slowest decoder (two orders of magnitude slower than Morph-Dict and three orders slower than Gemini), whereas the Morph-Dict decoder delivers near-instantaneous decoding but at lower phonetic fidelity. From a practical perspective, the choice among these systems depends on whether phoneme-level similarity (PER) or speed and word-level correctness (WER/CER) are the primary concern.

5.2 Comparison to the LLM baseline

The primary research question (RQ1, see section 1.4) asked whether a phone-based homophonic translation module can achieve lower Hungarian ASR WER on a phonetic test set than an end-to-end LLM baseline. The experiments in Chapter 4 provide a clear answer in favour of the modular pipeline.

On the hard-case set, every DP-Phone configuration with IPA map v1 outperforms the Gemini baseline on WER and CER (Table 4.1). The best DP-Phone setting (top-8k, v1) improves WER from 1.650 (Gemini) to 1.165 and CER from 1.502 to 0.895, while also slightly reducing PER (from 1.267 to 1.204). This pattern holds despite the fact that Gemini is free to choose arbitrarily fluent English phrases, whereas the DP-Phone decoder is restricted to a fixed lexicon and must explicitly pay phone-level substitution costs.

On the real-life set (Table 4.2), the gap in WER widens: Gemini's WER of 1.994 is markedly higher than the ≈1.18 achieved by both decoders. PER again favours DP-Phone

(0.952) over Gemini (1.034). The per-class breakdown in Table 4.3 shows that Gemini is competitive or slightly better only in a few classes (notably some front-rounded vowels and sibilant clusters) and typically trails DP-Phone on plain sibilants, voicing assimilation targets and vowel length. On the hard-case test set, Appendix B shows the same pattern more strongly: Gemini is especially fragile on palatal and cluster-heavy classes.

Qualitative examples in Table 4.6 further illustrate the difference in behaviour. DP-Phone outputs tend to be "clunkier" and less fluent but systematically track the Hungarian vowel and consonant patterns. Gemini, in contrast, generates smooth, sentence-like English that often drifts away from the intended phonology. When these outputs are fed through the same TTS→ASR pipeline, the DP-Phone sentences more often return to the original Hungarian, confirming that explicit phone constraints are more effective than generic text generation for this task.

In short, for the homophonic objective as operationalised here (Hungarian ASR WER/CER/PER on the TTS→ASR loop), the phone-based DP-Phone module consistently outperforms the Gemini baseline. The modular approach answers RQ1 in the affirmative: explicit phoneme-level control is advantageous over an end-to-end LLM that is not specifically optimised for cross-lingual sound preservation.

5.3 Practical applicability and usability

The hard-case test set was deliberately constructed to stress phonological edge cases (Appendix A.0.1). Therefore, it underestimates performance on everyday input, a fact confirmed by the real-life fragments in Appendix A.0.2 and Section 4.4. On these news sentences, all systems achieve lower PER than on tongue-twisters, with DP-Phone dropping from 1.204 to 0.952, Morph-Dict from 1.495 to 1.214, and Gemini from 1.267 to 1.034.

The pilot calibration in Section 4.3 provides a first anchor for interpreting these numbers: even for two widely recognised, high-quality homophonic translations, the TTS→ASR loop yields roughly

$$\mathrm{WER}_{\mathrm{gold}} \approx 0.84, \quad \mathrm{CER}_{\mathrm{gold}} \approx 0.39, \quad \mathrm{PER}_{\mathrm{gold}} \approx 0.64.$$

From a human perspective, these outputs are clearly acceptable, yet the automatic metrics still classify a large portion of the phones as errors. This suggests that "perfect"

homophonic translations under this evaluator live in a relatively high-PER regime.

Instead of fixing a rigid numerical threshold for stating "it is usable", it is more informative to compare systems relative to this calibrated regime. On the hard-case set, the best DP-Phone configuration lies about 0.57 PER points above the calibrated baseline ($1.204 - 0.64$), whereas the Morph-dict decoder is 0.86 points above it and Gemini about 0.63 points above it. On the real-life set, DP-Phone sits only ≈ 0.31 PER points above the same baseline (0.952 vs. 0.64), while Morph-Dict and Gemini are at ≈ 0.57 and ≈ 0.39 points above. In other words, DP-Phone is consistently closer to the "gold-standard" regime across both test sets, especially when the phonological difficulty is less extreme.

Listening to the outputs (Table 4.6 and additional examples) supports this view. DP-Phone outputs are generally more intelligible as approximations of the Hungarian source and some of the better translations could already be used for playful applications such as soramimi-style jokes, mnemonic phrases, or language-learning games where approximate sound is more important than meaning. The Morph-Dict decoder tends to produce shorter, sometimes more readable phrases but at the cost of more phone-level distortions, which narrows its applicability to cases where WER/CER matter more than detailed pronunciation. Gemini is best suited for scenarios where fluent English text is required and strict homophony is not essential: its outputs can be entertaining but less predictable in terms of which Hungarian contrasts are preserved.

Therefore, DP-Phone is the most promising candidate for practical homophonic tools: its errors relative to the calibrated baseline are small enough that human listeners can still recognise the source sentence in some cases, especially outside the most challenging phoneme families.

5.4 Generalisation beyond Hungarian

The pipeline examined in this thesis is intentionally modular and, apart from the language-specific resources it uses, does not depend on Hungarian-specific heuristics. Generalising it to another source language would require the following components:

- A grapheme-to-phoneme (G2P) system or pronunciation lexicon for the source language, analogous to emPhon for Hungarian, ideally with at least rough handling of morphophonology.

- A mapping from source-language phones to target-language proxies, derived from articulatory feature representations such as PanPhon. This can mirror the HU→EN maps (v1/v2) used here but must be adapted to the specific phoneme inventories.
- A target-language pronunciation lexicon (similar to CMUdict) that provides phones for a reasonably large, frequency-filtered vocabulary.
- A reasonably high-quality TTS system in the target language and an ASR system in the source language, so that the TTS→ASR evaluation loop can be reused without structural changes.

Languages with rich vowel length and rounding contrasts (e.g., Finnish, Swedish), complex consonant clusters (e.g., German, Polish) or tone (e.g., Mandarin) present different challenges. For length/rounding-rich languages, the main difficulty mirrors the Hungarian front-rounded and long–short contrasts: finding stable proxies in the target language and tuning search costs accordingly. For tone languages, the phone-based search would need to incorporate tone patterns and would require TTS and ASR tools that reliably realise and recognise tones.

In all cases, the role of the LLM baseline is unchanged: given suitable prompts and vocabulary constraints, an LLM can provide a quick end-to-end baseline, but the experiments here suggest that explicit phoneme-level modules will remain competitive or superior whenever fine-grained control over cross-lingual sound is required.

5.5 Limitations and evaluation noise

The TTS→ASR calibration in Section 4.3 highlights a central limitation of the current evaluation: even the best available homophonic translations, judged by humans, produce non-zero and sometimes high WER/CER/PER. The noise arises from several sources:

- **TTS variability.** The English TTS voice may realise the same orthographic sequence with slightly different vowel quality, length or prosody, which can shift how the Hungarian ASR interprets it.
- **ASR confusions.** Whisper-HU occasionally confuses sibilants, mis-segments clusters, or neutralises length, especially when hearing atypical English-accented speech intended as Hungarian.

- **Tokenisation effects.** Hungarian compounds and multiword named entities introduce segmentation ambiguities that affect both phone alignment and word counts, inflating WER and CER.

- **Calibration coverage.** The calibration uses only two gold sentences, so the resulting "noise floor" is indicative rather than statistically robust.

Beyond evaluator noise, the study has several structural limitations. The hard-case test set is small (28 items) and intentionally biased towards difficult phonological contexts, which is useful for analysis but not representative of everyday language use. The real-life set partly compensates for this but is itself small (15 fragments) and unbalanced across phoneme classes. Decoder hyperparameters (beam width, cost weights) were tuned by hand within a limited time budget rather than via large-scale automatic optimisation, and only a single LLM (Gemini-2.5-flash) was evaluated.

Consequently, the reported numbers should be interpreted primarily as *relative* comparisons between systems under a fixed evaluation harness, not as absolute measures of intelligibility. The consistent advantage of DP-Phone over Gemini and Morph-Dict in PER is robust to these limitations, but the exact PER values and their relation to human perception will need to be revisited with richer evaluation protocols (e.g. listening tests, multiple TTS/ASR stacks, and a larger gold set).

5.6 Future work

Several extensions follow naturally from the present findings.

- **Improved search and representations.** A full WFST-based decoder with on-the-fly composition and class-aware pruning could reduce DP-Phone's runtime while preserving or improving accuracy. Incorporating richer phonological features (e.g. tone, stress) may help in languages where these are contrastive.

- **Class-aware modelling;** The per-class analyses in Section 4.5 suggest targeted interventions: larger beams or specialised costs for front-rounded vowels, sibilant clusters and palatals, and possibly separate lexicon slices for cluster-heavy contexts.

- **Better morphological baselines.** The Morph-Dict decoder could benefit from improved segmentation (e.g. joint morph–syllable models) and stricter handling of

one-character units. Alternative higher-level units (subwords, orthographic syllables) might offer a better balance between compositionality and vocabulary size.

- **Human-centred evaluation.** Complementing WER/CER/PER with listening tests would make it possible to directly correlate automatic metrics with perceived "sounds like Hungarian" scores.

- **Cross-lingual experiments.** Re-implementing the pipeline for other language pairs (e.g. Finnish→English, Japanese→English) would test how much of the present behaviour is Hungarian-specific and how much generalises.

- **LLM integration.** Finally, one promising direction is hybridisation: using LLMs to propose candidate translations that are then re-ranked or post-edited by a phone-based module, combining the fluency of LLMs with the control of explicit phoneme search.

Taken together, the results demonstrate that even in the age of large language models, a carefully engineered phonetic pipeline remains competitive and, for homophonic translation, often preferable. The modular evaluation framework developed here provides a starting point for future work on cross-lingual sound-based translation, both for Hungarian and beyond.

Acknowledgements

I am deeply grateful to my supervisor, Dr. Balázs Indig (Department of Artificial Intelligence, Eötvös Loránd University, Faculty of Informatics), for his generous guidance, sharp questions, and steady encouragement throughout this thesis. His clarity of thought, practical advice, and patience helped shape both the research direction and the final manuscript.

Appendix A

Test sentences

A.0.1 Hard-case phonetic test set

Provenance. Items #1–#9 are *tongue-twisters* so they're explicitly designed for phonetic difficulty and articulation training, so they're a legitimate, phone-focused test source; items #10–#29 are *constructed* to fill remaining phonetic gaps.

1. **Mit sütsz, kis szűcs?** [17] ("What are you roasting, little furrier?")
   Cases: front-rounded /y, yː /, sibilants /s, sʲ/, affricate /tʃ/, vowel length.

2. **Sárga bögre, görbe bögre.** [17] ("Yellow mug, crooked mug.")
   Cases: front-rounded /ø/, long vowels /aː/, /eː/.

3. **Fekete bikapata kopog a patika pepita kövezetén.** [17] ("The black bull's hoof clatters on the pharmacy's checkered pavement.")
   Cases: plosives /p, t, k/, front-rounded /ø/, rhythm.

4. **Két pék két szép kék képet kér.** [18] ("Two bakers ask for two beautiful blue pictures.")
   Cases: vowel length /eː/, alveolar vs. velar.

5. **Jamaika a jamaikaiaké.** [18] ("Jamaica belongs to the Jamaicans.")
   Cases: vowel hiatus, final long /eː/.

6. **A szecsuáni síncsiszoló sínt csiszol Szecsuánban.** [18] ("The Sichuan rail-grinder grinds rails in Sichuan.")
   Cases: sibilant/affricate clusters /s, sʲ, tʃ/, long /iː/.

7. **Jobb egy lúdnyak két tyúknyaknál.** [19] (“One goose-neck is better than two hen-necks.”)
   Cases: palatals /c, ɟ, ɲ/, long vowels /uː/, /aː/.

8. **Lali a lila ló elalél.** [19] (“Lali, the purple horse, faints at sight.”)
   Cases: vowel length contrasts /oː/, /eː/, lateral repetition.

9. **Nem lehet a Márta másé, mert a Márta már Tamásé.** [20] (“Márta cannot belong to anyone else, since Márta already belongs to Tamás.”)
   Cases: vowel length /aː/, /eː/, vowel harmony.

10. **Gyűrűt újra fűz.** (“He re-threads a ring.”) *Cases:* /ɟ/, /yː/, /ʊː/, /y/; mixed voicing.

11. **Hány nyár?** (“How many summers?”) *Cases:* /ɲ/ across boundary; vowel length /ʌː/.

12. **Tyúk és csibe.** (“Hen and chick.”) *Cases:* /ʧ/ (ty) vs /t͡ʃ/ (cs).

13. **Dzsinn ül a dzsámi előtt.** (“A djinn sits in front of the mosque.”) *Cases:* /ʤ/ (dzs) word-initial (multiple tokens).

14. **Sárban szárad.** (“It dries in mud.”) *Cases:* /ʃ/ vs /s/ minimal contrast.

15. **Zsákban a zsemle.** (“The bun in the sack.”) *Cases:* /ʒ/ vs /z/.

16. **Egy nagy kertben.** (“In a big garden.”) *Cases:* /ɟ/ + /k/ (voicing assimilation target).

17. **Őr űzi az ürgét.** (“A guard chases the ground squirrel.”) *Cases:* /øː/, /yː/, /y/.

18. **Két bögre teát kér.** (“Ask for two mugs of tea.”) *Cases:* /ˈo/; t+b voicing context.

19. **A kórus a kocsma előtt áll.** (“The choir stands in front of the pub.”) *Cases:* /ɔː/ vs /ɔ/; /t͡ʃ/.

20. **Szép zöld fű.** (“Beautiful green grass.”) *Cases:* /ʒ/ vs /z/; /y/; clusters.

21. **Sütőtökből sütit süt.** (“Bakes cake from pumpkin.”) *Cases:* /ö, öː/ alternation; sibilants.

22. **Hidd el, itt írok.** (“Believe me, I’m writing here.”) *Cases:* /i/ vs /iː/; geminate /dd/.

23. **Vár a víz.** (“The water waits.”) *Cases:* /ʌː/; /i/ vs /iː/ in rhyme.

24. **Füstöl a fűrész.** (“The saw is smoking.”) *Cases:* /yː/, sibilants.

25. **Gyúl a Gyertya.** (“The candle lights.”) *Cases:* /ɟ/ clusters; /ʊː/.

26. **Ég a lámpa.** (“The lamp is on.”) *Cases:* long vs short vowels.

27. **Csőből csöpög.** (“It drips from the pipe.”) *Cases:* /t͡ʃ/ onset; /ö/ vs /øː/ (ö vs ő); voiced coda /g/.

28. **Kő őrzi őzt.** (“Stone guards roe deer.”) *Cases:* /øː/; cluster timing test.

A.0.2 Real-life news-style sentence fragments

The real-life evaluation set described in Section 4.1 consists of the following fifteen sentence fragments drawn from Hungarian online news articles and event reports:

1. *Fekete macska szaladt át az úton*

2. *Nevetve futottunk végig az utcán*

3. *Kint fúj a szél bent meleg van*

4. *Halkan csörög a telefon a zsebemben*

5. *csütörtökön délután a Nemzetközi Labdarúgó szövetség*

6. *budapesten mutatták be Gyenes Gábor lövés című képregényét*

7. *az eseményen jelen volt a képregény alapjául szolgáló*

8. *felé nyúlni és egy gyors és határozott mozdulattal*

9. *kinyomni a fenébe. Mert tapasztalataim szerint nincs*

10. *pótselejtezők menetrendjét A négyszeres aranyérmes*

11. *a walesi bosnyák továbbjutója otthonában harcolhatja*

12. *akkor az esetek többségében érdemes gyorsan a távirányító*

13. *Olaszország Észak Írországot fogadja majd*

14. *amely a kilencvenes évek vérben és erőszakban fürdő*

15. *maffia működéséről és brutalitásáról*

Appendix B

Detailed Per-Class Results

This appendix reports the full set of per-class evaluation metrics that were only summarised in the main text. For each experimental condition, we provide word error rate (WER), character error rate (CER), phoneme error rate (PER), and, where applicable, the average search cost for all phonological classes.

The tables in this appendix serve three purposes:

1. **Transparency and reproducibility:** they allow readers to inspect the raw per-class behaviour underlying the aggregate results reported in Sections 4.2
2. **Fine-grained error analysis:** they make it possible to identify which specific phonological classes (e.g. affricates, front-rounded vowels, palatals, clusters) dominate the error profile for a given decoder or search configuration.
3. **Future comparison:** they provide a reference for subsequent work to compare against, either at the aggregate level or for individual classes of interest.

Unless noted otherwise, the class labels follow the taxonomy introduced in Section 4.5, and all metrics are computed on the same held-out test set as in the main experiments.

Table B.1: Selected per-class scores, DP (v1), Zipf ≥4.5.

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	1.083333	0.691612	0.984886	10.672000
affricate_cs_onset	1.500000	1.230769	2.100000	8.400000
affricate_dzs	1.000000	0.750000	1.214286	10.760000
affricates	1.333333	0.687500	0.911765	16.075000
clusters	1.083333	0.887432	1.451389	6.480500
final_long_e	1.666667	1.000000	1.217391	15.029000
front_rounded	1.190476	0.793140	1.293844	12.663000
front_rounded_long	1.285714	1.151215	1.680952	9.208143
front_rounded_short	1.500000	1.244444	1.916667	9.068000
geminate	1.400000	0.718750	1.318182	14.934000
geminate_dd	1.000000	0.750000	1.000000	5.040000
hiatus	1.666667	1.000000	1.217391	15.029000
i_vs_long_i	1.000000	0.750000	1.000000	5.040000
i_vs_long_i_rhyme	1.000000	1.000000	1.285714	6.448000
laterals	1.000000	0.523810	0.588235	8.267000
long_a	1.000000	1.000000	1.285714	6.448000
long_i	1.333333	0.687500	0.911765	16.075000
long_o	1.000000	0.730769	1.000000	14.013000
long_u	1.333333	0.857143	1.555556	7.358000
long_vs_short_vowels	0.666667	0.900000	0.875000	8.024000
palatal_gy	1.166667	0.895238	1.736111	9.288000
palatal_gy_clusters	1.333333	0.857143	1.555556	7.358000
palatal_ny	1.200000	0.859375	1.492424	11.927000
palatal_ty	1.200000	0.743990	1.214646	9.987000
plosives	1.202381	0.777611	0.948065	13.518750
rhythm	1.380952	0.828482	1.097420	16.626667
s_vs_sh	1.000000	0.846154	1.181818	10.520000
sibilant_clusters	1.333333	0.687500	0.911765	16.075000
sibilants	1.125000	0.903801	1.257329	8.864750
voiced_coda_g	1.500000	1.230769	2.100000	8.400000
voicing_assimilation_target	0.666667	0.625000	0.500000	4.195000
vowel_harmony	1.000000	0.673913	0.918919	24.524000
vowel_length	1.168254	0.895760	1.303836	10.871407
word_initial	1.000000	0.750000	1.214286	10.760000
z_vs_zh	1.000000	0.875000	1.000000	9.160000

Table B.2: Selected per-class scores, DP (v2), Zipf ≥4.5.

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	3.062500	2.948212	3.794118	9.278500
affricate_cs_onset	1.000000	0.846154	1.300000	8.000000
affricate_dzs	1.000000	0.750000	0.714286	7.800000
affricates	2.166667	0.833333	1.176471	14.082000
clusters	2.812500	3.407482	4.444444	5.651500
final_long_e	2.000000	1.190476	0.913043	5.556000
front_rounded	2.315476	2.313723	2.873926	9.652333
front_rounded_long	1.166667	1.019347	1.372222	7.510286
front_rounded_short	1.444444	1.166667	1.500000	7.035000
geminate	1.400000	0.750000	1.363636	12.611000
geminate_dd	1.000000	0.750000	0.818182	5.040000
hiatus	2.000000	1.190476	0.913043	5.556000
i_vs_long_i	1.000000	0.750000	0.818182	5.040000
i_vs_long_i_rhyme	1.000000	1.000000	1.714286	4.610000
laterals	1.200000	0.571429	0.352941	3.291000
long_a	1.000000	1.000000	1.714286	4.610000
long_i	2.166667	0.833333	1.176471	14.082000
long_o	0.833333	0.692308	0.888889	11.102000
long_u	1.000000	0.928571	1.222222	4.912000
long_vs_short_vowels	1.000000	0.600000	0.625000	4.181000
palatal_gy	1.000000	0.830952	1.236111	6.961500
palatal_gy_clusters	1.000000	0.928571	1.222222	4.912000
palatal_ny	1.200000	0.763889	1.681818	9.765500
palatal_ty	1.200000	0.798077	1.237374	8.825500
plosives	1.160714	0.805946	0.938114	10.342000
rhythm	1.214286	0.803762	0.945263	12.617667
s_vs_sh	1.500000	0.692308	0.818182	6.237000
sibilant_clusters	2.166667	0.833333	1.176471	14.082000
sibilants	2.135417	2.054937	2.732255	6.949750
voiced_coda_g	1.000000	0.846154	1.300000	8.000000
voicing_assimilation_target	1.000000	0.812500	0.916667	3.515000
vowel_harmony	1.000000	0.630435	0.729730	15.774000
vowel_length	1.456526	1.194472	1.573473	8.067815
word_initial	1.000000	0.750000	0.714286	7.800000
z_vs_zh	1.000000	1.062500	1.833333	5.337000

Table B.3: Selected per-class scores, DP (v1), Zipf ≥5.

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	1.000000	0.709240	1.054330	10.460625
affricate_cs_onset	1.000000	0.846154	1.200000	7.771500
affricate_dzs	1.000000	0.875000	1.714286	10.760000
affricates	1.000000	0.604167	0.911765	15.411000
clusters	1.000000	0.887432	1.451389	6.024125
final_long_e	1.333333	0.809524	1.086957	13.669500
front_rounded	1.398810	1.203139	1.241676	11.274750
front_rounded_long	0.964286	0.834366	1.290476	8.676000
front_rounded_short	0.916667	0.722222	1.314815	8.334333
geminate	1.400000	0.750000	1.318182	14.907000
geminate_dd	1.000000	0.625000	0.636364	5.040000
hiatus	1.333333	0.809524	1.086957	13.669500
i_vs_long_i	1.000000	0.625000	0.636364	5.040000
i_vs_long_i_rhyme	1.000000	1.000000	1.571429	6.424000
laterals	1.400000	1.095238	1.882353	8.253500
long_a	1.000000	1.000000	1.571429	6.424000
long_i	1.000000	0.604167	0.911765	15.411000
long_o	1.000000	0.730769	1.055556	13.986500
long_u	1.000000	0.857143	1.777778	6.446500
long_vs_short_vowels	0.666667	0.400000	0.375000	6.687500
palatal_gy	1.000000	0.961905	1.888889	8.827750
palatal_gy_clusters	1.000000	0.857143	1.777778	6.446500
palatal_ny	1.200000	0.763889	1.159091	11.913500
palatal_ty	1.200000	0.836538	1.325758	9.896000
plosives	1.764881	1.349903	0.902965	11.707375
rhythm	1.797619	1.529037	0.926175	14.224000
s_vs_sh	2.000000	1.076923	1.545455	9.209000
sibilant_clusters	1.000000	0.604167	0.911765	15.411000
sibilants	1.125000	0.824313	1.216582	8.314750
voiced_coda_g	1.000000	0.846154	1.200000	7.771500
voicing_assimilation_target	1.666667	0.812500	0.833333	4.157500
vowel_harmony	1.000000	0.717391	0.972973	24.402000
vowel_length	1.146032	0.897678	1.218064	10.180630
word_initial	1.000000	0.875000	1.714286	10.760000
z_vs_zh	1.000000	0.875000	1.166667	9.160000

Table B.4: Selected per-class scores, DP Top8k (v1)

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	1.000000	0.709240	1.054330	10.460625
affricate_cs_onset	1.000000	0.846154	1.200000	7.771500
affricate_dzs	1.000000	0.875000	1.714286	10.760000
affricates	1.000000	0.604167	0.911765	15.411000
clusters	1.000000	0.887432	1.451389	6.024125
final_long_e	1.333333	0.809524	1.086957	13.669500
front_rounded	1.398810	1.203139	1.241676	11.274750
front_rounded_long	0.964286	0.834366	1.290476	8.676000
front_rounded_short	0.916667	0.722222	1.314815	8.334333
geminate	1.400000	0.750000	1.318182	14.907000
geminate_dd	1.000000	0.625000	0.636364	5.040000
hiatus	1.333333	0.809524	1.086957	13.669500
i_vs_long_i	1.000000	0.625000	0.636364	5.040000
i_vs_long_i_rhyme	1.000000	1.000000	1.571429	6.424000
laterals	1.400000	1.095238	1.882353	8.253500
long_a	1.000000	1.000000	1.571429	6.424000
long_i	1.000000	0.604167	0.911765	15.411000
long_o	1.000000	0.730769	1.055556	13.986500
long_u	1.000000	0.857143	1.777778	6.446500
long_vs_short_vowels	0.666667	0.400000	0.375000	6.687500
palatal_gy	1.000000	0.961905	1.888889	8.827750
palatal_gy_clusters	1.000000	0.857143	1.777778	6.446500
palatal_ny	1.200000	0.763889	1.159091	11.913500
palatal_ty	1.200000	0.836538	1.325758	9.896000
plosives	1.764881	1.349903	0.902965	11.707375
rhythm	1.797619	1.529037	0.926175	14.224000
s_vs_sh	2.000000	1.076923	1.545455	9.209000
sibilant_clusters	1.000000	0.604167	0.911765	15.411000
sibilants	1.125000	0.824313	1.216582	8.314750
voiced_coda_g	1.000000	0.846154	1.200000	7.771500
voicing_assimilation_target	1.666667	0.812500	0.833333	4.157500
vowel_harmony	1.000000	0.717391	0.972973	24.402000
vowel_length	1.146032	0.897678	1.218064	10.180630
word_initial	1.000000	0.875000	1.714286	10.760000
z_vs_zh	1.000000	0.875000	1.166667	9.160000

Table B.5: Selected per-class scores, DP (v2), Zipf ≥5.

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	1.000000	0.709240	1.054330	10.460625
affricate_cs_onset	1.000000	0.846154	1.200000	7.771500
affricate_dzs	1.000000	0.875000	1.714286	10.760000
affricates	1.000000	0.604167	0.911765	15.411000
clusters	1.000000	0.887432	1.451389	6.024125
final_long_e	1.333333	0.809524	1.086957	13.669500
front_rounded	1.398810	1.203139	1.241676	11.274750
front_rounded_long	0.964286	0.834366	1.290476	8.676000
front_rounded_short	0.916667	0.722222	1.314815	8.334333
geminate	1.400000	0.750000	1.318182	14.907000
geminate_dd	1.000000	0.625000	0.636364	5.040000
hiatus	1.333333	0.809524	1.086957	13.669500
i_vs_long_i	1.000000	0.625000	0.636364	5.040000
i_vs_long_i_rhyme	1.000000	1.000000	1.571429	6.424000
laterals	1.400000	1.095238	1.882353	8.253500
long_a	1.000000	1.000000	1.571429	6.424000
long_i	1.000000	0.604167	0.911765	15.411000
long_o	1.000000	0.730769	1.055556	13.986500
long_u	1.000000	0.857143	1.777778	6.446500
long_vs_short_vowels	0.666667	0.400000	0.375000	6.687500
palatal_gy	1.000000	0.961905	1.888889	8.827750
palatal_gy_clusters	1.000000	0.857143	1.777778	6.446500
palatal_ny	1.200000	0.763889	1.159091	11.913500
palatal_ty	1.200000	0.836538	1.325758	9.896000
plosives	1.764881	1.349903	0.902965	11.707375
rhythm	1.797619	1.529037	0.926175	14.224000
s_vs_sh	2.000000	1.076923	1.545455	9.209000
sibilant_clusters	1.000000	0.604167	0.911765	15.411000
sibilants	1.125000	0.824313	1.216582	8.314750
voiced_coda_g	1.000000	0.846154	1.200000	7.771500
voicing_assimilation_target	1.666667	0.812500	0.833333	4.157500
vowel_harmony	1.000000	0.717391	0.972973	24.402000
vowel_length	1.146032	0.897678	1.218064	10.180630
word_initial	1.000000	0.875000	1.714286	10.760000
z_vs_zh	1.000000	0.875000	1.166667	9.160000

Table B.6: Selected per-class scores, Morph Dictionary (n>3) (v1)

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	0.916667	0.813154	1.419935	7.121165
affricate_cs_onset	1.000000	1.000000	2.100000	4.050000
affricate_dzs	1.000000	0.833333	1.642857	6.666667
affricates	0.833333	0.770833	1.235294	13.143826
clusters	1.000000	0.817769	1.500000	3.687500
final_long_e	1.000000	0.952381	1.478261	9.408333
front_rounded	1.000000	0.829780	1.512766	7.797415
front_rounded_long	1.071429	0.815085	1.494444	5.210374
front_rounded_short	1.222222	0.788889	1.342593	5.054683
geminate	7.400000	7.781250	4.136364	8.820833
geminate_dd	1.000000	0.750000	1.545455	4.370000
hiatus	1.000000	0.952381	1.478261	9.408333
i_vs_long_i	1.000000	0.750000	1.545455	4.370000
i_vs_long_i_rhyme	1.000000	0.777778	1.714286	3.267143
laterals	1.000000	0.761905	1.352941	7.083750
long_a	1.000000	0.777778	1.714286	3.267143
long_i	0.833333	0.770833	1.235294	13.143826
long_o	0.833333	0.769231	1.555556	7.570000
long_u	1.000000	0.928571	1.777778	3.383333
long_vs_short_vowels	1.000000	0.900000	1.500000	3.455000
palatal_gy	1.000000	0.897619	1.722222	4.630238
palatal_gy_clusters	1.000000	0.928571	1.777778	3.383333
palatal_ny	4.200000	4.279514	2.734848	6.015417
palatal_ty	4.200000	4.352163	2.845960	6.340000
plosives	0.916667	0.805635	1.313145	8.795865
rhythm	1.000000	0.907514	1.473082	10.637820
s_vs_sh	1.000000	0.538462	1.000000	4.043333
sibilant_clusters	0.833333	0.770833	1.235294	13.143826
sibilants	1.083333	0.777308	1.338466	4.935506
voiced_coda_g	1.000000	1.000000	2.100000	4.050000
voicing_assimilation_target	0.666667	0.500000	0.833333	3.270000
vowel_harmony	0.900000	0.673913	1.162162	13.850833
vowel_length	1.245679	1.073126	1.561036	6.468326
word_initial	1.000000	0.833333	1.642857	6.666667
z_vs_zh	1.000000	0.687500	1.166667	6.477143

Table B.7: Selected per-class scores, Morph Dictionary (n>3) (v2)

Class	WER	CER	PER	Avg. Search Cost
affricate_cs	0.958333	0.780765	1.316585	6.911478
affricate_cs_onset	1.000000	1.000000	2.100000	3.785000
affricate_dzs	0.800000	0.916667	2.214286	6.241389
affricates	1.000000	0.770833	1.294118	12.942857
clusters	1.000000	0.850066	1.562500	3.643750
final_long_e	1.000000	0.809524	1.304348	8.420000
front_rounded	0.998413	0.821773	1.436189	7.261713
front_rounded_long	0.976190	0.813953	1.514286	4.702602
front_rounded_short	1.000000	0.722222	1.324074	4.392183
geminate	1.000000	0.812500	1.500000	8.019167
geminate_dd	1.000000	0.750000	1.181818	4.065000
hiatus	1.000000	0.809524	1.304348	8.420000
i_vs_long_i	1.000000	0.750000	1.181818	4.065000
i_vs_long_i_rhyme	1.000000	1.111111	1.857143	1.816667
laterals	1.000000	0.857143	1.647059	7.186250
long_a	1.000000	1.111111	1.857143	1.816667
long_i	1.000000	0.770833	1.294118	12.942857
long_o	0.833333	0.846154	1.500000	7.660000
long_u	1.000000	0.928571	1.777778	3.783333
long_vs_short_vowels	1.000000	1.000000	1.250000	3.355000
palatal_gy	1.166667	0.864286	1.555556	4.355000
palatal_gy_clusters	1.000000	0.928571	1.777778	3.783333
palatal_ny	1.000000	0.795139	1.416667	5.614583
palatal_ty	1.000000	0.790865	1.361111	5.850278
plosives	1.214286	1.359213	1.931146	8.237222
rhythm	0.952381	0.833117	1.380417	10.282963
s_vs_sh	1.000000	0.461538	0.909091	2.416667
sibilant_clusters	1.000000	0.770833	1.294118	12.942857
sibilants	1.000000	0.717174	1.303854	4.134688
voiced_coda_g	1.000000	1.000000	2.100000	3.785000
voicing_assimilation_target	2.000000	2.937500	3.583333	2.100000
vowel_harmony	0.800000	0.630435	1.081081	11.242083
vowel_length	0.986067	0.807203	1.417168	5.874762
word_initial	0.800000	0.916667	2.214286	6.241389
z_vs_zh	1.000000	0.687500	1.166667	4.976667

Table B.8: Selected per-class scores, Gemini

Class	WER	CER	PER
affricate_cs	1.145833	0.673646	1.105392
affricate_cs_onset	1.500000	0.846154	1.300000
affricate_dzs	1.000000	0.500000	1.214286
affricates	1.666667	0.437500	0.588235
clusters	2.354167	2.653765	1.555556
final_long_e	1.000000	0.428571	0.608696
front_rounded	1.317460	0.696507	1.033275
front_rounded_long	2.142857	2.267649	1.245238
front_rounded_short	1.333333	0.927778	1.416667
geminate	1.400000	0.625000	1.136364
geminate_dd	1.000000	0.687500	1.181818
hiatus	1.000000	0.428571	0.608696
i_vs_long_i	1.000000	0.687500	1.181818
i_vs_long_i_rhyme	1.333333	2.444444	0.285714
laterals	1.000000	0.619048	0.647059
long_a	1.333333	2.444444	0.285714
long_i	1.666667	0.437500	0.588235
long_o	1.166667	0.807692	1.277778
long_u	7.000000	8.785714	2.666667
long_vs_short_vowels	0.666667	0.400000	0.875000
palatal_gy	4.166667	4.692857	1.750000
palatal_gy_clusters	7.000000	8.785714	2.666667
palatal_ny	2.450000	3.201389	4.484848
palatal_ty	1.200000	0.774038	1.345960
plosives	1.315476	0.660057	0.775146
rhythm	1.642857	0.817576	0.950194
s_vs_sh	1.000000	0.384615	0.727273
sibilant_clusters	1.666667	0.437500	0.588235
sibilants	1.867560	1.887683	0.970589
voiced_coda_g	1.500000	0.846154	1.300000
voicing_assimilation_target	0.333333	0.187500	0.250000
vowel_harmony	0.900000	0.413043	0.486486
vowel_length	1.712169	1.585792	1.289768
word_initial	1.000000	0.500000	1.214286
z_vs_zh	1.000000	0.437500	0.750000

Appendix C

Archive and Negative Results

WFST experiment. A composition-based search using Pynini/OpenFst was attempted but ran out of resources on the local machine; the run was excluded from final products. The approach remains promising with on-the-fly composition and tighter pruning.[1]

[1] See scripts under `/archive`.

Bibliography

[1] Jonathan Fiscus and NIST. *SCTK: The NIST Scoring Toolkit (sclite, sc_stats) Documentation*. https://github.com/usnistgov/SCTK/tree/master/doc. 2024.

[2] Thennal D K et al. *Advocating Character Error Rate for Multilingual ASR Evaluation*. 2024. arXiv: 2410.07400 [cs.CL]. URL: https://arxiv.org/abs/2410.07400.

[3] Arun Baby et al. "An ASR Guided Speech Intelligibility Measure for TTS Model Selection". In: *arXiv:2006.01463* (2020).

[4] Carnegie Mellon University. *CMU Pronouncing Dictionary v0.7b*. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. 2014.

[5] Alec Radford et al. "Robust Speech Recognition via Large-Scale Weak Supervision". In: *arXiv preprint arXiv:2212.04356* (2022).

[6] Alexei Baevski et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *NeurIPS* (2020).

[7] Péter Siptár and Miklós Törkenczy. *The Phonology of Hungarian*. Oxford University Press, 2000.

[8] Virág Kulcsár and Dániel Lévai. "emPhon: Morphologically sensitive open-source phonetic transcriber". In: *XVII. Conference on Hungarian Computational Linguistics (MSZNY2021)*. Szeged, 2021.

[9] David Mortensen, Siddharth Dalmia, and Patrick Littell. "PanPhon: A Resource for Mapping IPA Segments to Articulatory Feature Vectors". In: *COLING*. 2016.

[10] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd. Stanford / draft online, 2023. URL: https://web.stanford.edu/~jurafsky/slp3/.

[11] Mehryar Mohri, Fernando Pereira, and Michael Riley. "Speech Recognition with Weighted Finite-State Transducers". In: *Springer Handbook of Speech Processing*. 2008.

[12] Kyle Gorman. *Pynini: Weighted Finite-State Grammar Library*. `https://github.com/kylebgorman/pynini`. 2016.

[13] Coqui.ai. *Coqui TTS: A deep learning toolkit for Text-to-Speech*. `https://github.com/coqui-ai/TTS`. 2021.

[14] Attila Novák, Borbála Siklósi, and Csaba Oravecz. "A New Integrated Open-source Morphological Analyzer for Hungarian". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari et al. Portorož, Slovenia: European Language Resources Association (ELRA), 2016. ISBN: 978-2-9517408-9-1.

[15] Google AI. *Migrate to the Google GenAI SDK*. `https://ai.google.dev/gemini-api/docs/migrate`. Accessed 2025-09-11. 2025.

[16] Google AI. *Gemini API: Text generation (Python with `from google import genai`)*. `https://ai.google.dev/gemini-api/docs/text-generation`. Accessed 2025-09-11. 2025.

[17] Omniglot. *Hungarian tongue twisters*. `https://omniglot.com/language/tonguetwisters/hungarian.htm`. Accessed: 2025-09-11. 2025.

[18] alphaDictionary. *Hungarian Tongue Twisters*. `https://www.alphadictionary.com/fun/tongue_twisters/hungarian_tongue_twisters.html`. Accessed: 2025-09-11. 2025.

[19] International Tongue Twister Collection. *Hungarian Tongue Twisters*. `http://www.uebersetzung.at/twister/hu.htm`. Accessed: 2025-09-11. 2025.

[20] SpeakingTongue. *Hungarian Tongue Twisters*. `https://speakingtongue.com/hungarian-tongue-twisters/`. Accessed: 2025-09-11. 2025.

List of Figures

List of Tables