

# Creating ConLangs to Probe the Metalinguistic Grammatical Knowledge of LLMs

**Chihiro Taguchi**  
University of Notre Dame  
Notre Dame, IN, USA  
ctaguchi@nd.edu

**Richard Sproat**  
Sakana AI  
Tokyo, Japan  
rws@sakana.ai

## Abstract

We present a system that uses LLMs as a tool in the development of Constructed Languages—ConLangs, which we call  $1\Delta SC$  (Interactive Agentic System for ConLangs). The system is modular in that it creates each of the components—phonology, morphology and syntax, lexicon, orthography, and grammatical handbook, using module-specific sets of prompts. The approach is agentic in that various modules allow for refining the output given automatically-generated commentary on a previous step. Our main goals are twofold. First, we aim to provide tools that facilitate an engaging and enjoyable experience in creating artificially constructed languages. Second, the focus of this paper is on using our ConLang framework as a novel way to explore what LLMs ‘know’ about language—not what they know about any particular language or encyclopedic facts, but how much they know about and understand language and linguistic concepts. In the experiments, we particularly focus on the morphosyntax module and show that there is a fairly wide gulf in capabilities both among different LLMs and among different linguistic specifications, with it being notably easier for systems to deal with more typologically common patterns than rarer ones. All code is released: <https://github.com/SakanaAI/IASC>.

## 1 Introduction

The term ‘Constructed Language’—often shortened to ‘ConLang’—is used to refer to any artificially created language that is intended, in principle, to be as expressive as naturally evolved human languages. The latter restriction is important since ConLangs are to be distinguished from artificial languages such as mathematical formulae and programming languages, which are artificial languages (at least in the formal sense of language) to be sure, but which are much more limited in the kinds of messages they can convey.

This paper introduces  $1\Delta SC$ ,<sup>1</sup> an *Interactive Agentic System for ConLangs*, a system that enables users to customize linguistic parameters to design their own ConLang with a large language model (LLMs). The system consists of modules for phonology, morphosyntax, lexicon creation, orthography, and grammatical handbook writing, with each module being responsible for generating linguistic rules or transformed output that follow high-level specifications given by the user. For some modules, the output can be refined agentically using automatically generated feedback on a previous step. The system’s workflow is illustrated in Figure 1.

$1\Delta SC$  presents a unique challenge for LLMs as the workflow requires deep metalinguistic knowledge for each module. That is, the model needs not only to know encyclopedic facts about languages and linguistic features but also to explicitly understand abstract linguistic concepts and manipulate language based on grammatical rules. For example, suppose that a user wants to construct a language with a Verb-Subject-Object word order and has an input sentence “the cat caught the fish.” Then, the morphosyntax module of the system should ideally output “caught the cat the fish.” This sort of linguistic transformation requires strong metalinguistic reasoning skills beyond mere encyclopedic knowledge about languages.

Thus, the purpose of this paper is twofold. First, we present  $1\Delta SC$  as an engaging, flexible, and reproducible tool for designing and generating unique ConLangs.<sup>2</sup> Second, given the challenging nature of  $1\Delta SC$ ’s linguistic transformations, we use the morphosyntax module of the system as

<sup>1</sup>IASC is to be pronounced as /i:ask/ in English. The word *iasc* means ‘fish’ in Modern Irish.

<sup>2</sup>In fact online tools to help with some of the steps already exist, including <http://VulgarLang.com>, as well as instruction manuals such as Rosenfelder (2012). Our system can thus be thought of as a much broader, AI-driven version of these.

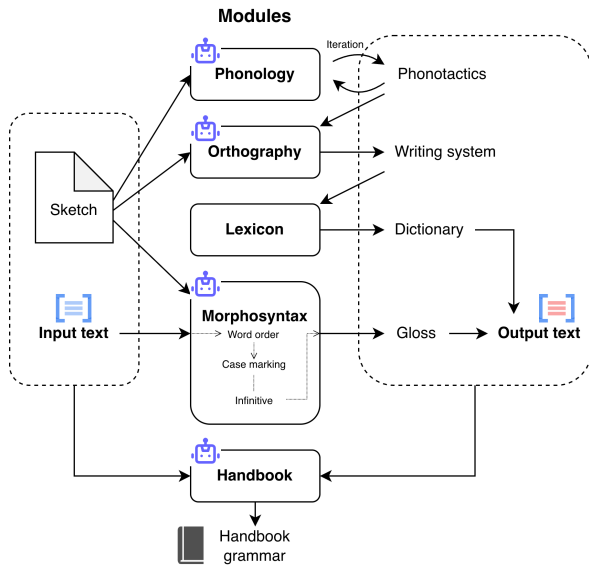


Figure 1: The workflow of 1ASC.

a benchmark to probe and evaluate the metalinguistic grammatical knowledge of LLMs. Breaking down the problem into targeted tasks, such as setting word order and case marking, makes it easy to evaluate whether LLMs can be induced to produce outputs that conform to expectations. For this task, we construct an evaluation dataset that tests nine different morphosyntactic configurations with typologically diverse sets of features. As we will see, LLMs show a range of abilities that to some degree seem to relate to how frequent and well-documented a linguistic phenomenon is in the published literature, and how well the feature is likely to be supported in the system’s training data.

## 2 Related work

### 2.1 ConLangs

When people think of ConLangs, the first things that come to mind may be languages like Esperanto (Zamenhof, 1887) or Interlingua (Gode and Blair, 1951), which are artificial languages designed to be like, and replace, natural languages, though with fewer of the grammatical complexities one often finds in real natural languages. One might also think of fantasy languages, such as the languages of Tolkien’s Middle-earth like Elvish (Quenya) (Tolkien, 1968), or Science Fiction ‘alien’ languages like Klingon (Okrand, 1985) or the Hep-tapod language from *Arrival* (Villeneuve, 2016). Finally, there is a third category of languages that are supposed to be based on ‘logical’ principles, such as LogLan or John Wilkins’ ‘philosophical

language’ (Wilkins, 1668). In 1ASC, we are interested in discovering to what degree Large Language Models (LLMs) can help in the creation of ConLangs that are much like real human languages in their properties, but are not intended to replace naturally evolved languages. Thus the ConLangs created here fit more into the fantasy-SF theme.

### 2.2 LLMs’ metalinguistic abilities

A large amount of research has been devoted to investigating the linguistic abilities of LLMs and their implications for linguistic theory. The latter point in particular has included some strong claims, with Chomsky et al. (2023) arguing in an OpEd that LLMs are completely irrelevant to the study of language, and Piantadosi (2024) arguing the opposite, namely that LLMs completely refute linguistic theories positing the innateness of grammatical knowledge.

Of most relevance to our work are studies on LLMs’ metalinguistic knowledge, in particular how good language models are at performing tasks that linguists might perform, such as doing syntactic annotation, or correctly marking syntactic arguments in a language that has a particular case-marking pattern. Studies along these lines include (Blevins et al., 2023; Ginn et al., 2024; Buhnla, 2025; Suvarna et al., 2024; Jang et al., 2025; Merullo et al., 2025; Dinu et al., 2025; Saeed et al., 2025; Pantelidou et al., 2025; Beguš et al., 2025; Goyal and Dan, 2025; Choudhary et al., 2025).

Studies of LLMs’ abilities at constructing ConLangs have been more limited, though artificial languages have figured in various studies of other aspects of LLMs’ linguistic abilities (White and Cotterell, 2021; Kallini et al., 2024; Marmonier et al., 2025; Swain et al., 2025). Of more direct relevance to our work, Diamond (2023) investigated the use of ChatGPT to construct languages via prompting, and studied properties of the languages that the system came up with, and whether the constructed languages obeyed the standard ‘Zipfian’ power law distribution. He concludes that the texts in the constructed languages show broadly similar word frequency distributions to their English translations.

While we were conducting our research, Alper et al. (2025)’s paper on ConlangCrafter appeared. Their model is similar in some respects to our approach in that, unlike Diamond (2023), it breaks the problem down into separate modules of phonology, morphology, syntax, lexicon generation and

translation, each of which is controlled by a high-level description of the properties the target language should have. Alper et al. (2025)’s work is obviously related to ours, but there are also important differences, particularly when it comes to morphosyntactic feature generation. In particular, we adopt a very fine-grained multistage approach to generating morphosyntax, which has the advantage that we can investigate the question of how well LLMs ‘know’ specific morphosyntactic features, abstracting away from other features.

The interested reader can find a more detailed discussion of prior work in Appendix A.1.

### 3 Stages of ConLang Construction

IASC is a full system for creating ConLangs, and as such consists of several components that each deal with a different portion of the system. For reasons of space, we mainly focus in this paper on the morphosyntactic component, which has the most implications for what LLMs know about language. Other components, on which the interested reader can find many more details in the Appendix, are phonotactics (A.3), lexicon generation (A.5), orthography creation (A.6), the writing of a grammatical handbook of the language (A.7), and the translation of further texts into the language (A.8). The system also depends on a set of (English) texts that serve as the input from which target language data is constructed: construction of this data is described in A.4. In addition, in A.11 we discuss a novel agentic approach to generating phonological rules based on attested sets of sound changes. And in A.9 we discuss possible applications of our approach to morphosyntax to low-resource languages. Finally, A.2 outlines the linguistic background that we assume for our discussion.

However, to set the stage for the ensuing discussion, it is useful to present briefly how the system constructs languages, and what an example language looks like. Suppose we want a language with Welsh-like phonotactics, an Ergative-Absolutive case system with a typologically unusual Object-Verb-Subject word order like that of Mizo (see Section 4), and an orthography based on the Latin script. The system first creates a Python program to generate morphemes that attempt to approximate the phonemes and their phonotactic distribution for the target language. This is done iteratively, with the system examining its own previous output, and possibly improving upon its previous work in

an agentic fashion. Next, the morphosyntax is developed, as described in detail in the next section. Once this is done, and based on the morphosyntactic glosses of the source sentences for which we wish to create target-language equivalents, the system will then create the needed target-language morphemes using the phonotactic grammar to generate the forms. The system then develops an orthography in the desired script, based on the phonotactic grammar developed previously, and using a similar approach. Finally, the system writes a grammatical handbook for the language, and invents a name for the language.

An example sentence from the Welsh-Mizo-Latin ConLang ‘Thrlangai’, is as follows:

- (1) *Chacho shitho krsüb nükhai næb*  
 climate change country we ERG  
*rékólh there nonü noli snpor*  
 together address needs to with  
*grnaim lauben.*  
 third singular

‘Our country needs to address climate change together’

Important features include ergative (ERG) case marking (*næb*) on *krsüb nükhai* ‘our country’, the placement of the object *Chacho shitho* ‘climate change’ before the subject, conforming to the specifications of the desired morphosyntactic structure, and the occurrence of the lateral fricative /ɬ/, a very characteristic sound of Welsh, here written as ⟨lh⟩ in the word *rékólh*.

For another example, consider a language that has Japanese-like phonology, Turkish-like morphosyntax—including Subject-Object-Verb word order and a negation suffix on verbs—and a Cyrillic-based orthography. An example from this language, called Koshanese, is as follows:

- (2) Кешизо дате шукутуа курауоотенсе  
*ke.fi-zo da.te fu.ku-tu-a*  
 she-NOM spicy food-SING-ACC  
*ku.ra-u-o-o-ten-se*  
 like-NEG-PRES-IND-3SG-ACT

‘She doesn’t like spicy food’

Examples of other generated languages can be found in the Appendix and at <https://github.com/SakanaAI/IASC/tree/main/handbooks>.

## 4 Morphosyntax

This section details the Morphosyntax module of  $\Delta$ ASC as well as experiments that probe LLMs’ metalinguistic knowledge. The term *morphosyntax* denotes the subset of grammar that relates to how words are arranged in a sentence (syntax) and how grammatical information is marked on words (morphology). Morphology addresses how a word is structured, and syntax addresses how words form a phrase and a sentence. For example, a form *dogs* consists of two *morphemes* (minimum meaningful units): *dog* and the plural suffix *-s*. In this case, we can analyze the form morphologically as *dog-s*, and annotate it as *dog-PLUR*. This morpheme-to-morpheme annotation scheme, where morphemes like PLUR (plural marker) are expressed in an abbreviated form, is called a *gloss*. Syntax, on the other hand, is concerned with how words combine to form a larger phrase, such as a noun phrase, prepositional phrase, or an entire sentence. For example, in English, a language with a relatively rigid word order, a canonical declarative sentence is formed by arranging the constituents in the order of subject, verb, and object (SVO), as in *the dog chased the cat*.

What features are handled in morphology and syntax depend on languages. Some languages employ rich morphology to encode various grammatical features. For example, in Latin, nominal morphology encodes grammatical gender, number, and case, and verbal morphology encodes subject person, subject number, tense–aspect, mood, and voice. On the other hand, other languages have simpler morphology and depend on syntactic constraints to encode grammatical features. For example, though Vietnamese morphology does not encode any of the features found in Latin morphology, at least some such information is nonetheless encoded in syntax by means of uninflected lexical items and rigid word order. Given this cross-linguistic variability and the intertwined interactions in the realms of morphology and syntax, we treat them as one stage, *i.e.* morphosyntax, in our ConLang construction pipeline.

### 4.1 Research questions

Of the modules in  $\Delta$ ASC, we choose the Morphosyntax module as a probing system for evaluating the metalinguistic ability of LLMs since (1) this module requires a large amount of abstract understanding and reasoning about linguistic concepts,

and (2) the fact that LLMs are trained primarily on text makes morphosyntactic knowledge a natural first thing to investigate over, say, phonology.

Morphosyntactic transformation of natural language text by a language model poses the following two research questions:

1. **Metalinguistic ability.** It is already known that LLMs have linguistic knowledge (Beguš et al., 2025) and are able to perform translation to/from an unseen low-resource language (Tanzer et al., 2024; Zhu et al., 2025). It has also been reported that LLMs have the creative capacity to generate ConLangs on the fly (Alper et al., 2025). However, these findings still do not demonstrate the metalinguistic knowledge and ability of LLMs concerning morphosyntax. For instance, it is a straightforward task for linguistically-trained humans to transform an English sentence based on an instruction “Change the English sentence to OSV word order” or “Change the English sentence so that the past tense is expressed by a prefix PAST-”. How well can an LLM perform such tasks?
2. **Robustness to atypical typology.** How do atypical morphosyntactic features affect the ability of the LLM to perform the task? In principle, it is no harder to transform a sentence in SVO word order into, say OVS word order than it is to transform it to SOV word order. But the former is much less widely attested. Does this make a difference in the performance of LLMs on this task?

### 4.2 Methods

In the Morphosyntax module, our goal is to manipulate the morphosyntactic structure of a source sentence with a language model  $\mathcal{M}$  such that the output follows the grammatical specifications of a given ConLang. In particular, the input is a triple of a source sentence  $s_{\text{src}}$ , the morphosyntactic parameters  $G$  of our ConLang grammar, and a prompt text  $t$ , and we wish to obtain the target sentence  $s_{\text{tgt}} = \mathcal{M}(s_{\text{src}}, G, t)$ . In this Morphosyntax module, the output  $s_{\text{tgt}}$  is not the final translation of the source sentence but is given in the form of a gloss. For simplicity, the initial source sentence  $s_{\text{src}}$  and the prompt  $t$  are given in English.

Roughly speaking, there are two possible ways to implement the Morphosyntax module: (1) to transform a source sentence in one pass, and (2)

to cumulatively transform a source sentence with iterative prompting. Our preliminary experiments with the first approach showed that the language models struggle to follow the grammar specifications, likely because the needed prompts became too long and complicated. For this reason, the implementations of our main experiments follow the cumulative morphosyntax approach.

In the Cumulative Morphosyntax approach, the source sentences are transformed through iterative prompting. In each prompting step, the language model  $\mathcal{M}$ 's task is to transform the sentence of the previous output  $s_{i-1}$  such that it follows the grammar specifications  $G$  and the prompt text  $t_i$  corresponding to a specific grammatical feature:  $s_i = \mathcal{M}(s_{i-1}, G, t_i)$ . For example, suppose that we want to modify the case-marking system at the  $i$ th stage. Then, the placeholders of the prompt text  $t_i$  are filled based on the provided grammar  $G$ , and the model is expected to transform the input sentence  $s_{i-1}$  in such a way that only the case-marking system of the sentence is modified in the output.

The features used in the experiments can be seen in Table 1.

### 4.3 Evaluation data

To quantify the performance of the LLM's morphosyntactic manipulation, we constructed an evaluation dataset.<sup>3</sup> The dataset consists of 45 source sentences and nine ConLang feature sets, amounting to 405 sentences. These source sentences are designed to test a variety of grammatical features. For the ConLang feature sets, typologically diverse combinations of features were selected. To reflect the real-world typological diversity, we prepare sets of features inspired by eight existing languages: Arabic, Fijian, French, Hixkaryana, Mizo, Turkish, Vietnamese, and Welsh. In this paper, these feature sets are referred to by their quoted lowercase counterparts (*e.g.*, 'arabic'), but note that these names are only used for convenience and *do not strictly reflect all morphosyntax details of the named language*. We also prepare a feature set with a typologically extremely rare combination of the features, which we call the 'hard' language. This language contains an atypical mixture of head-initialness and head-finalness as well as complex and unusual morphology. See Appendix B for the

<sup>3</sup>The dataset is publicly available under the CC BY-SA 4.0 license at [https://huggingface.co/datasets/ctaguchi/conlang\\_eval\\_dataset](https://huggingface.co/datasets/ctaguchi/conlang_eval_dataset).

full details of the morphosyntax feature parameters. The gold data was created by one of the authors—a trained linguist—by translating source sentences to the targets in the form of glosses.

### 4.4 Evaluation metrics

The evaluation stage of the Morphosyntax module receives the gloss transformed from the source sentence as the input and measures the correctness. Since there can be small non-deterministic variations in the output formats provided by the language models, the output translations are fed into another lightweight language model to obtain a unified structured output in JSON format. In our experiments, GPT 4.1 mini was used for this structuring step.

**Translation Edit Rate (TER).** TER was proposed as a metric of translation quality by (Snover et al., 2006) and measures the correctness of a translation output by simply dividing the number of edits by the reference length:

$$\text{TER} = \frac{\text{Number of edits}}{\text{Reference length}}$$

In contrast to edit distance-based metrics such as WER, in TER a shift of a group of words by any number of positions counts as 1 edit. For example, compare the reference and hypothesis sentences given below:

- Reference: the cat caught the fat rat
- Hypothesis: the fat rat caught the cat

In this case, TER counts two edit operations to get the reference from the hypothesis: the fat rat to the right and the cat to the left. Therefore, the TER score for the hypothesis is  $2/6 \simeq 0.167$ .

**Stem Edit Rate (SER).** SER is an extension of TER to measure the correctness of the positions of the *stems* in the hypothesis glosses. This metric is useful to measure how well the language model is able to manipulate syntax, regardless of morphological changes in the output. In our implementation, SER is calculated by (1) obtaining the stems through removing the morphological features, which are expressed using capital letters and (2) calculating the TER of the stems. That is, TER counts the edits of the exact forms including the morphological feature tags, and SER only counts the edits of the stems, ignoring the morphological features.

Submodule	Features	Subfeatures	Values
Syntax	main_word_order		Literal["SOV", "SVO", "VSO", "VOS", "OSV", "OVS"]
	oblique_word_order		Literal["VOX", "VXO", "XOV", "XVO", "OVX", "OXV"]
	adj_noun_word_order		Literal["NA", "AN"]
	posspron_noun_word_order		Literal["PossN", "NPoss"]
	num_noun_word_order		Literal["NumN", "NNum"]
	adposition_noun_word_order		Literal["PN", "NP"]
Morphology	case	case_marking	List[Literal["nominative", "accusative", "dative", "genitive", "ablative", "locative", "instrumental", "ergative", "absolutive"]]
		case_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
		oblique_case_marking	Literal["nominative", "accusative", "dative", "genitive", "ablative", "locative", "instrumental", "ergative", "absolutive"]
	definiteness	definiteness	List[Literal["definite", "indefinite"]]
		definiteness_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	adjective_agreement	adjective_agreement	List[Literal["number", "case", "definiteness"]]
		adjective_agreement_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	comparative	comparative	List[Literal["comparative", "superlative", "equative"]]
		comparative_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	tense_aspect	tense_aspect	List[Literal["present", "past", "future", "perfect", "imperfect", "immediate past", "recent past", "remote past", "nonpast"]]
		nominal_number	Literal[List["singular", "plural", "dual", "paucal"]]
	person	nominal_number_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
		tense_aspect_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
		person_agreement	List[Literal["first", "second", "third"]]
		person_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	voice	verbal_number_agreement	List[Literal["singular", "plural", "dual", "paucal"]]
		verbal_number_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	mood	voice	List[Literal["active", "passive"]]
		voice_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	relativization	mood	List[Literal["indicative", "subjunctive", "imperative", "conditional"]]
		mood_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
	infinitive	relativization_order	Literal["head-initial", "head-final"]
		relativization_marking	Literal["head-marking", "dependent-marking"]
		relativizer_position	Literal["prepositional", "postpositional"]
		relativizer_morpheme	Literal["affix", "word"]
	nominal_number	infinitive	Literal["infinitive"]
		infinitive_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]
inclusive_exclusive	nominal_number	List[Literal["singular", "plural", "dual", "paucal"]]	
	nominal_number_marking_strategy	Literal["prefix", "suffix", "prepositional word", "postpositional word"]	
			bool

Table 1: The details of the Morphosyntax data structure. `Literal[...]` means that one value must be chosen from the list, and `List[Literal[...]]` means that multiple values from the list are accepted.

### Morphological Feature Error Rate (MFER).

In contrast to SER, MFER measures how well the language model can gloss morphological features in the correct form in the correct position. Thus, this metric does not take word order into account; rather, for each reference word, it searches for a word in the hypothesis that minimally deviates from the reference word. The morpheme-level edit distance is then used to compute the error rate. Note that, while we take into account whether a morpheme is a prefix or a suffix, we do not take into account the order within the sequence of prefixes or suffixes. In other words, sequences of suffixes like `-PRES-3SGNOM` and `-3SGNOM-PRES` are considered identical when computing MFER. It is implemented this way because the order within a sequence of affixes is not explicitly specified in the ConLang prompts.

### Morphosyntactic Error Rate (MSER).

MSER is proposed to take both morphological errors and syntactic errors into account by calculating a weighted average of SER and MFER ( $\alpha = 0.5$  by default):

$$\text{MSER} = \alpha \cdot \text{SER} + (1 - \alpha)\text{MFER}.$$

### Lemma F1 score (Lem F1).

This metric measures how well the model is capable of lemmatizing the input sentences. Lemmatization is important, since if one is to transform the source sentences into target glosses, one needs the base form of the original words, not the inflected form: thus `dog-PL-ERG` rather than `dogs-PL-ERG`. In preliminary experiments we found that several language models failed to lemmatize correctly, thereby negatively affecting other metric scores such as SER.

Given a predicted sequence  $s_p$  and its reference sequence  $s_r$ , we obtain the frequencies of the lemmata  $l_p$  and  $l_r$  expressed as multisets (bags-of-words). We then count the true positives  $\text{TP} = |l_p \cap l_r|$  (i.e., the predicted lemmata also included in the reference lemmata), false positives  $\text{FP} = l_p \setminus l_r$ , and false negatives  $\text{FN} = l_r \setminus l_p$ . After this, the lemma precision (LemPrec) and the lemma recall (LemRec) are calculated as:

$$\text{LemPrec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{LemRec} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The lemma F1 score (LemF1) is the harmonic mean of the lemma precision and the lemma re-

Model	Version
gpt-4o-mini	gpt-4o-mini-2024-07-18
gpt-4.1-mini	gpt-4.1-mini-2025-04-14
gpt-5-mini	gpt-5-mini-2025-08-07
gpt-5	gpt-5-2025-08-07
gemini-2.5-flash	gemini-2.5-flash (released: June 17, 2025)
gemini-2.5-pro	gemini-2.5-pro (released: June 17, 2025)
claude-3-5-sonnet	claude-3-5-sonnet-20240620

Table 2: Models and their snapshots.

call; namely,

$$\text{LemF}_1 = \frac{2 \cdot \text{LemPrec} \cdot \text{LemRec}}{\text{LemPrec} + \text{LemRec}}.$$

In the evaluation below, we utilize the macro lemma F1 scores (*i.e.*, averaging over the F1 scores of each prediction).

#### 4.5 Evaluation results

We evaluated the morphosyntactic transformation performance across six different language models of varying quality: Claude 3.5 Sonnet, Gemini 2.5 Flash, Gemini 2.5 Pro, GPT-4o-mini, GPT-5-mini, and GPT-5. The temperature is set to 0 where it can be specified, and thinking is enabled for the reasoning models. For snapshots of the specific versions, see Table 2. Figure 2 summarizes a comparison of MSER scores across the 9 feature sets and 6 LLMs. For full results see Table 18 in Appendix C. Each result is obtained from a single run.

**Typological unusualness.** As the results in Table 18 demonstrate, all of the language models struggled to transform the sentences into the ‘hard’ language. The ‘hard’ language has a rich morphology, and typologically unusual features such as case prefixation. In addition, we can observe higher SERs in languages with a typologically unusual word order such as ‘fijian’ (VOS), ‘mizo’ (OSV), and ‘hard’ (OSV). In contrast, languages with a typologically more common word order, such as ‘french’ (SVO), ‘arabic’ (VSO), and ‘welsh’ (VSO), yielded lower errors in general. Notably, Claude 3.5 Sonnet, Gemini 2.5 Pro, and GPT-5-mini yielded perfect SER scores (0.00) in arabic and welsh, both of which have a VSO word order. When we classify SOV, SVO, and VSO as common word orders and the others as uncommon word orders, there is a weak significant correlation between word order and SER (point-biserial correlation coefficient  $r_{pb} = 0.29$ ,  $p = 0.0026 < 0.05$ ).

#### Lemmatization: analytic vs. synthetic language.

Another interesting observation is that some high-performance LLMs struggled to correctly lemmatize analytic languages, while performing better in synthetic languages with heavy morphological inflection. Let us suppose that ‘fijian’, ‘mizo’, and ‘vietnamese’ are analytic, and the other languages are synthetic, based on the description given in (Bickel and Nichols, 2013). Then, we observe a weak significant correlation between the inflectional degrees and the Lemma F1 ( $r_{pb} = 0.32$ ,  $p = 0.00082 < 0.05$ ). In other words, languages that rely more on word order and function words rather than inflectional morphology generally posed greater challenges for lemma recognition by LLMs. In the output for the analytic languages, we observed that some inflected nouns and verbs were left inflected. Overall, the language that the models struggled the most is ‘hard’ with an average MSER score of 85.21, and the easiest language was ‘welsh’ with an average MSER score of 28.26.

**Effect of few-shot in-context learning.** Given the difficulty of the task, we have also tested the models’ performance with an additional stage called *review*, where a model is provided with a prompt that contains a summary of the grammar and four example sentences with label glosses at the end of the Morphosyntax module. This allows the model to digest the grammar with a few-shot in-context learning (ICL) setting. In this case, as shown in Table 3, Claude 3.5 Sonnet, Gemini 2.5 Flash, Gemini 2.5 Pro, and GPT-5 showed an improvement for most metrics.

**Model ‘intelligence’ and performance.** The results demonstrate that there are differences in the ability of morphosyntactic manipulation across different feature sets and language models. It can be observed that the LLMs are generally able to transform the input into ConLang grammars to some extent. However, there is a significant difference between larger LLMs (Claude 3.5 Sonnet, Gemini 2.5 Pro, and GPT-5) and the others ( $r_{pb} = -0.31$ ,  $p = 0.001 < 0.05$ ). As the results so far suggest, not all models were able to perform the task satisfying the conditions given in the prompt instructions. In particular, GPT-4o-mini and GPT-5-mini struggled with the task even when the few-shot examples were given. The model that performed the best in this task overall was Gemini-2.5-Pro with few-shot ICL, outperforming the other models in all the

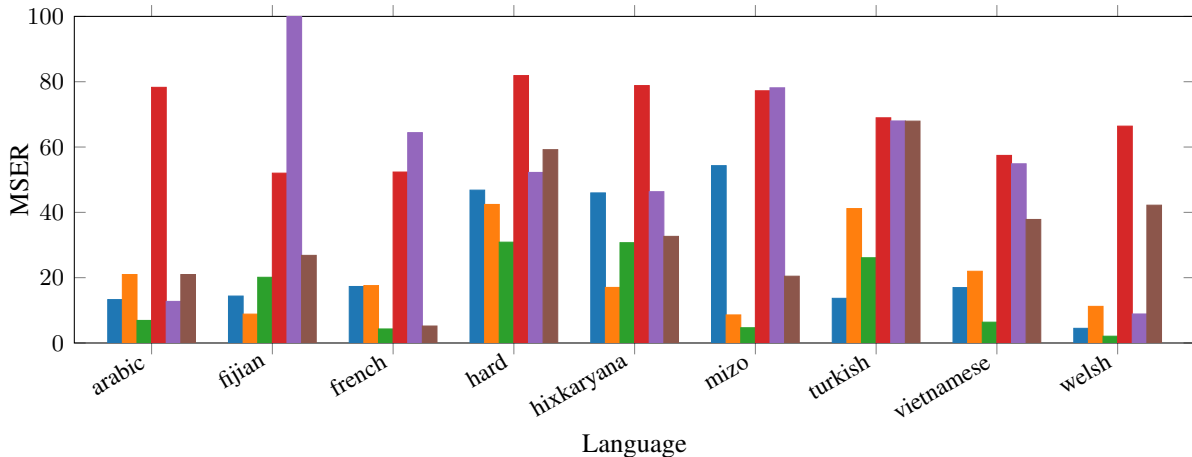


Figure 2: MSER by model and language. All scores are the results with the few-shot in-context examples. ■ Claude 3.5 Sonnet, ■ Gemini 2.5 Flash, ■ Gemini 2.5 Pro, ■ GPT-4o-mini, ■ GPT-5-mini, ■ GPT-5.

Model	ICL	TER	SER	MFER	MSER	LemF1
claude-3-5-sonnet	No	77.78	33.37	41.73	37.55	76.17
	Yes	<b>55.22</b>	<b>26.29</b>	<b>24.18</b>	<b>25.23</b>	<b>89.13</b>
gemini-2.5-flash	No	74.86	43.07	47.64	45.36	73.11
	Yes	<b>48.35</b>	<b>18.02</b>	<b>24.14</b>	<b>21.08</b>	<b>89.99</b>
gemini-2.5-pro	No	83.00	60.80	47.57	54.18	74.38
	Yes	<b>40.94</b>	<b>12.64</b>	<b>16.71</b>	<b>14.68</b>	<b>89.30</b>
gpt-4o-mini	No	<b>87.58</b>	71.64	66.67	69.16	52.79
	Yes	88.83	<b>69.93</b>	<b>66.36</b>	<b>68.14</b>	<b>53.49</b>
gpt-5-mini	No	<b>80.80</b>	<b>62.70</b>	47.80	55.25	68.28
	Yes	84.76	62.79	<b>46.01</b>	<b>54.40</b>	<b>69.13</b>
gpt-5	No	78.53	45.58	46.26	45.92	61.76
	Yes	<b>57.57</b>	<b>35.70</b>	<b>33.90</b>	<b>34.80</b>	<b>79.90</b>

Table 3: Average scores over languages by models comparing performance with and without a few-shot in-context learning (ICL) stage.

metrics but Lemma F1 by a significant margin as demonstrated in Table 3.

Table 4 presents a comparison of the actual outputs by the different models for a sentence in the ‘hard’ language with few-shot ICL. Overall, Gemini 2.5 Pro is able to follow the grammar specifications of the ‘hard’ language, including the OSV word order, ergative-absolutive alignment, and prefixation. On the other hand, GPT-4o-mini completely ignores morphology even with the few-shot examples. These results suggest that larger state-of-the-art LLMs do have relevant metalinguistic knowledge, though capabilities still greatly vary across models.

## 5 Conclusion

This study introduced a novel agentic LLM system for ConLang generation. With this system, we examined how effectively LLMs can manipulate and generate morphosyntactic features in response to explicit instructions. We found that larger SOTA LLMs are mostly able to follow the grammatical instructions correctly to generate appropriate target-language glossing, while smaller models struggled to perform the task. The performance of the larger LLMs can be further improved by providing few-shot in-context examples before generating the final output. We observed correlations between the LLMs’ performance and typological features, LLMs struggling more with ConLangs that have a unusual word order or have no morphological inflection.

To specifically answer our research questions:

1. **Metalinguistic ability.** Larger LLMs are generally capable of word order permutations, though there is a definite performance drop with typologically unusual orders.
2. **Robustness to atypical typology.** Larger LLMs performed well at many of the tasks, though even some large LLMs struggled with the ‘hard’ language, which combined several typologically unusual configurations. Oddly, LLMs performed worse on a crucial processing step—lemmatization—if the target language was more ‘analytic’.

For the large proprietary models in particular, we cannot unfortunately know *why* the models performed as they did. However, it is unlikely that

Model	ICL	Output
Source Gold		The children played in the park. DEF-ABS-PLUR-child DEF-LOC-SING-park RECPAST-3PLABS-IND-play.
claude-3-5-sonnet	No	DEF-PLUR-ABS-child DEF DEF-SING-park-LOC DEF in RECPAST-ACT-play-3PLNOM.
	Yes	ABS-DEF-PLUR-child LOC-DEF-SING-park the in RECPAST-3PLABS-IND-ACT-play.
gemini-2.5-flash	No	abs-DEF-PLUR-child active-plural-play-third LOC-DEF-SING-park in.
	Yes	LOC-DEF-SING-park in ERG-DEF-PLUR-child RECPAST-3PLERG-IND-ACT-play.
gemini-2.5-pro	No	abs-DEF-plur-child loc-DEF-sing-park in plural-RECPAST-active-play-third.
	Yes	DEF-ABS-PLUR-child DEF-LOC-SING-park in RECPAST-3PLABS-IND-ACT-play.
gpt-4o-mini	No	park the in played children the.
	Yes	park the in played children the.
gpt-5-mini	No	ABS-the DEF-PLUR-child PL-ACT-PL-REMPAST-IND-play INS-DEF-SING-park the in.
	Yes	INS-DEF-SING-park in ABS-DEF-PLUR-child ACT-PLUR-REMPAST-IND-play-THIRD.
gpt-5	No	(NA)
	Yes	RECPAST-3PLABS-IND-ACT-play DEF-ABS-PLUR-child DEF-INS-SING-park in.

Table 4: A comparison of example model outputs in the ‘hard’ language. The most plausible output is that of Gemini 2.5 Pro, which correctly follows the word order, ergative-absolutive alignment, and most of the morphology. GPT-5 without in-context examples did not generate any output for the sentence.

they would have been exposed to much text in languages that have, say, VOS orders. Similarly, while the models have presumably seen grammatical information (*e.g.* on Wikipedia) about such languages, such grammars will be rare compared to grammars for languages with more ‘typical’ grammatical properties. But in order to perform the transformation to that word order, it is not necessary that the model have seen real life instances: it suffices that it understands concepts like *subject*, *object*, and *verb*, and that it understands what it means to put phrases in a particular order. For this sort of metalinguistic task, the ‘knowledge’ of the LLMs seems to be mostly sufficient, though once again, we observe that the more unusual the target, the more the model struggles with the task.

## 6 Limitations

There are many obvious limitations to our work.

The first and most obvious one is that in developing the morphosyntactic properties of the target ConLang, we assume a model where one translates in successive steps into the target language. The result is that the target language will likely retain properties of the source language—English, in our experiments. In the case of English this means that one will often find residues of English-specific syntax, such as *do*-support, use of auxiliary *be* in passives, along with direct translations of English expressions, in the target language. One possible approach to improve this would be to explicitly instruct the model to avoid direct translation of particular terms (especially things like auxiliary verbs) and be creative in its choice of target expressions.

The morphosyntax is currently limited to the set of features that we have selected as interesting, and there are many features that we have not treated, that are grammatically expressed in at least some languages. One example is diminutives and augmentatives, found in many languages. Some of the features that we do in principle cover, such as gender, remain problematic. If one is creating a ConLang, except in the case of natural gender, it is not obvious how to assign gender to a given target word. Beyond that, the current system only supports masculine, feminine and neuter genders, which reflects just one kind of gender system. Obviously this could be extended to cover, for example, Bantu-like gender systems, but the question of how the system would decide what gender to assign to a given word remains.

Our results are of course also limited by our evaluation set which, while we attempted to make it comprehensive, obviously could be expanded to cover a wider range of examples that test the systems' abilities.

Another issue is that the morphology is currently limited to prefixation and suffixation. Furthermore, once an affix has been chosen to express a particular morphosyntactic feature, the same affix will be used to express that feature in all cases. Sequences of morphosyntactic features such as SING-NOM are expressed by sequences of affixes. The current system is thus biased towards a strictly affixal, single-exponent, agglutinative strategy, where furthermore there is no paradigmatic variation.

The phonotactic, phonological and orthographic components of our system, along with the genera-

tion of handbooks as documented in the Appendix also have limitations, but we omit discussion of these since they are not part of the main paper.

Finally, we compared several LLMs in their ability to carry out the tasks required of them. On balance one system that fared very well over a variety of tasks was Gemini 2.5 Pro. But we hasten to add that our coverage is not comprehensive. In any case, the abilities of LLMs are constantly changing and improving, so it is likely that the capability of systems to carry out the detailed instructions required of them in our tasks will only improve over time.

## 7 Ethical considerations

While it is hard to think of any technology that could not be used for harm, and while it is certainly the case that LLMs and Generative AI more generally have raised serious red flags when it comes to ethical and safety issues, we believe that our work does not present any obvious ethical problems. All of the data used in this project are either public data or synthetic, and all of the data created are, by definition, synthetic.

Since the process of ConLang creation sketched here is in part random, it is possible that the system might generate, unintentionally, outputs that could be deemed offensive. The most obvious example would be words that are homophones or close homophones with offensive terms in some language. In that situation, a user of our system could easily intervene and change the offensive item, if so desired.

During the preparation of this paper, AI assistants were used to correct grammatical errors and improve readability, as one of the authors is not a native speaker of English.

## 8 Acknowledgments

We thank various colleagues at Sakana.ai for feedback, in particular Robert Lange for some in-depth discussions about agentic systems. We also thank three anonymous reviewers and the metareviewer for helpful feedback.

## References

- Adrian Akmajian, Richard Demers, Ann Farmer, and Robert Harnish. 2001. *Linguistics: An Introduction to Language and Communication*, fifth edition. MIT Press, Cambridge, MA.
- Morris Alper, Moran Yanuka, Raja Giryes, and Gašper Beguš. 2025. [ConlangCrafter: Constructing languages with a multi-hop LLM pipeline](#). *Preprint*, arXiv:2508.06094.
- Seth Aycock, David Stap, Di Wu, Christof Monz, and Khalil Sima'an. 2025. [Can LLMs really learn to translate a low-resource language from one grammar book?](#) *Preprint*, arXiv:2409.19151.
- Marcin Badurowicz, Stanisław Skulimowski, and Maciej Laskowski. 2024. Feasibility of using low-parameter local LLMs in answering questions from enterprise knowledge base. *Applied Computer Science*, 20(4):175–191.
- Gašper Beguš, Maksymilian Dabkowski, and Ryan Rhodes. 2025. [Large linguistic models: Investigating LLMs' metalinguistic abilities](#). *IEEE Transactions on Artificial Intelligence*, page 1–15.
- Katrien Beuls and Paul Van Eecke. 2024. [Humans learn language from situated communicative interactions. What about machines?](#) *Computational Linguistics*, 50(4):1277–1311.
- Balthasar Bickel and Johanna Nichols. 2013. [Inflectional synthesis of the verb \(v2020.4\)](#). In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Zenodo.
- Juliette Blevins. 2004. *Evolutionary Phonology: The Emergence of Sound Patterns*. Cambridge University Press.
- Terra Blevins, Hila Gonen, and Luke Zettlemoyer. 2023. [Prompting language models for linguistic structure](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6649–6663, Toronto, Canada. Association for Computational Linguistics.
- Ioana Buhnila. 2025. Semantic and pragmatic annotation of paraphrases in the era of large language models. In *Linguistic Society of Korea*, pages 61–62, Seoul.
- Lalnunthangi Chhangte. 1989. The grammar of simple clauses in Mizo. *South-East Asian Syntax*, pages 93–174.
- Hyesun Cho. 2025a. Comparing LLMs and humans in the gender classification of Korean names. In *Linguistic Society of Korea*, pages 105–106, Seoul.
- Ye-eun Cho. 2025b. Do large language models possess pragmatic competence? A methodological approach. In *Linguistic Society of Korea*, pages 63–64, Seoul.
- Noam Chomsky, Ian Roberts, and Jeffrey Watumull. 2023. The false promise of ChatGPT. *New York Times*.
- Mukund Choudhary, KV Aditya Srivatsa, Gaurja Aeron, Antara Raaghavi Bhattacharya, Dang Khoa Dang Dinh, Ikhlusal Akmal Hanif, Daria Kotova, Ekaterina Kochmar, and Monojit Choudhury. 2025. [Unveiling: What makes Linguistics Olympiad puzzles tricky for LLMs?](#) *Preprint*, arXiv:2508.11260.
- Jaime Collado-Montañez, L. Alfonso Ureña-López, and Arturo Montejo-Ráez. 2025. [Towards fundamental language models: Does linguistic competence scale with model size?](#) *Preprint*, arXiv:2509.02225.
- Bernard Comrie. 2013. [Numeral bases \(v2020.4\)](#). In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Zenodo.
- Marianne de Heer Kloots and Willem Zuidema. 2024. Human-like linguistic biases in neural speech models: Phonetic categorization and phonotactic constraints in Wav2Vec2.0. In *Interspeech 2024*, page 4593–4597. ISCA.
- Desmond C. Derbyshire. 1985. *Hixkaryana and Linguistic Typology*. Summer Institute of Linguistics and the University of Texas at Arlington.
- Justin Diamond. 2023. [“Genlangs” and Zipf’s law: Do languages generated by ChatGPT statistically look human?](#) *Preprint*, arXiv:2304.12191.
- Anca Dinu, Andra-Maria Florescu, and Alina Resceanu. 2025. [A comparative approach to assessing linguistic creativity of large language models and humans](#). *Preprint*, arXiv:2507.12039.
- Raiomond Doctor, Alexander Gutkin, Cibu Johny, Brian Roark, and Richard Sproat. 2022. [Graphemic normalization of the Perso-Arabic script](#). In *Grapholinguistics in the 21st Century*, Télécom Paris, Palaiseau, France.
- Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online (v2020.4)*. Zenodo.
- Richard Futrell and Kyle Mahowald. 2025. [How linguistics learned to stop worrying and love the language models](#). *Preprint*, arXiv:2501.17047.
- Michael Ginn, Mans Hulden, and Alexis Palmer. 2024. [Can we teach language models to gloss endangered languages?](#) *Preprint*, arXiv:2406.18895.
- Alexander Gode and Hugh Blair. 1951. *A Grammar of Interlingua*. International Auxiliary Language Association.
- Kyle Gorman and Richard Sproat. 2022. [On the persistent conflation of writing and language](#). In *Grapholinguistics in the 21st Century*, Télécom Paris, Palaiseau, France.

- Satyam Goyal and Soham Dan. 2025. [IOL-BENCH: Benchmarking LLMs on linguistic reasoning](#). *Preprint*, arXiv:2501.04249.
- Jennifer Hu, Kyle Mahowald, Gary Luyuan, Anna Ivanova, and Roger Levy. 2024. [Language models align with human judgments on key grammatical constructions](#). *Proceedings of the National Academy of Sciences*, 121(36).
- Yusuke Ide, Yuto Nishida, Justin Vasselli, Miyu Oba, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2025. [How to make the most of LLMs’ grammatical knowledge for acceptability judgments](#). *Preprint*, arXiv:2408.09639.
- Dongjun Jang, Youngchae Ahn, and Hyopil Shin. 2025. [P-CoT: A pedagogically-motivated participatory chain-of-thought prompting for phonological reasoning in LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21958–21979, Vienna, Austria. Association for Computational Linguistics.
- Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. 2024. [Mission: Impossible language models](#). *Preprint*, arXiv:2401.06416.
- Anisia Katinskaia and Roman Yangarber. 2024. [Probing the category of verbal aspect in transformer language models](#). *Preprint*, arXiv:2406.02335.
- Jordan Kodner, Sarah Payne, and Jeffrey Heinz. 2023. [Why linguistics will thrive in the 21st century: A reply to Piantadosi \(2023\)](#). *Preprint*, arXiv:2308.03228.
- Shalom Lappin. 2024. [Assessing the strengths and weaknesses of large language models](#). *Journal of Logic, Language and Information*, 33:9–20.
- Miguel López-Otal, Jorge Gracia, Jordi Bernad, Carlos Bobed, Lucía Pitarch-Ballesteros, and Emma Anglés-Herrero. 2025. [Linguistic interpretability of transformer-based language models: a systematic review](#). *Preprint*, arXiv:2504.08001.
- Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. [The AI Scientist: Towards fully automated open-ended scientific discovery](#). *Preprint*, arXiv:2408.06292.
- Malik Marmonier, Rachel Bawden, and Benoît Sagot. 2025. [Explicit learning and the LLM in machine translation](#). *Preprint*, arXiv:2503.09454.
- Jack Merullo, Arjun Khurana, and Oliver McLaughlin. 2025. [I have no mouth, and I must rhyme: Uncovering internal phonetic representations in LLaMA 3.2](#). *Preprint*, arXiv:2508.02527.
- Steven Moran and Daniel McCloy. 2019. [PHOIBLE 2.0](#).
- Hiroshi Nakagawa. 2013. *Nyū Ekusupuresu Ainu-go (New Express Ainu)*. Hakuuisha, Tokyo.
- Hiroshi Nakagawa. 2024. *Ainu-go Kōbunten (A Comprehensive Grammar of the Ainu Language)*. Hakuuisha, Tokyo.
- Dmitry Nikolaev and Sebastian Padó. 2023. [Investigating semantic subspaces of transformer sentence embeddings through linear structural probing](#). *Preprint*, arXiv:2310.11923.
- Marc Okrand. 1985. *The Klingon dictionary*. Pocket Books.
- Arika Okrent. 2009. *In the Land of Invented Languages*. Spiegel & Grau, New York.
- Nikoleta Pantelidou, Evelina Leivada, and Paolo Morosi. 2025. [Resource-sensitive but language-blind: Community size and not grammatical complexity better predicts the accuracy of large language models in a novel wug test](#). *Preprint*, arXiv:2510.12463.
- Iñigo Parra. 2025. [Neural correlates of language models are specific to human language](#). *Preprint*, arXiv:2510.03156.
- Steven Piantadosi. 2024. [Modern language models refute Chomsky’s approach to language](#). In Edward Gibson and Moshe Poliak, editors, *From Fieldwork to Linguistic Theory: A Tribute to Dan Everett*, number 15 in Empirically Oriented Theoretical Morphology and Syntax, pages 353–414. Language Science Press.
- Aleksandr Piperski. 2017. *Конструирование языков: От эсперанто до дотракийского (Language Construction: from Esperanto to Dothraki)*. Alpina Non-Fiction, Moscow.
- F. Pucci. 1931. [Il traduttore meccanico ed il metodo per corrispondersi fra europei conoscendo ciascuno solo la propria lingua](#). Technical Report I, Salerno.
- Zhuang Qiu, Xufeng Duan, and Zhenguang G. Cai. 2024. [Grammaticality representation in ChatGPT as compared to linguists and laypeople](#). *Preprint*, arXiv:2406.11116.
- Jean-Marie Le Ray. 2025. [The forgotten code: Validating a century-old translation system with AI](#). *Preprint*, arXiv:2509.02506.
- Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. [The OpenGrm open-source finite-state grammar software libraries](#). In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, Jeju Island, Korea. Association for Computational Linguistics.
- Mark Rosenfelder. 2012. *The Language Construction Kit*.
- Mohammed J. Saeed, Tommi Vehvilainen, Evgeny Fedoseev, Sevil Caliskan, and Tatiana Vodolazova. 2025. [Impact: Inflectional morphology probes across complex typologies](#). *Preprint*, arXiv:2506.23929.

- Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. [A study of translation edit rate with targeted human annotation](#). In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.
- Ashima Suvarna, Harshita Khandelwal, and Nanyun Peng. 2024. [PhonologyBench: Evaluating phonological skills of large language models](#). *Preprint*, arXiv:2404.02456.
- Sankalp Tattwadarshi Swain, Anshika Krishnatray, Dhruv Kumar, and Jagat Sesh Challa. 2025. [Talking with Oompa Loompas: A novel framework for evaluating linguistic acquisition of LLM agents](#). *Preprint*, arXiv:2509.07389.
- Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. 2024. [A benchmark for learning to translate a new language from one grammar book](#). *Preprint*, arXiv:2309.16575.
- Lindia Tjauatja, Graham Neubig, Tal Linzen, and Sophie Hao. 2025. [What goes into a LM acceptability judgment? rethinking the impact of frequency and length](#). *Preprint*, arXiv:2411.02528.
- John Ronald Reuel Tolkien. 1968. *The Lord of the Rings*. Allen & Unwin, London.
- Tiago Timponi Torrent, Thomas Hoffmann, Arthur Lorenzi Almeida, and Mark Turner. 2024. *Copilots for Linguists: AI, Constructions, and Frames*. Elements in Construction Grammar. Cambridge University Press.
- Denis Villeneuve. 2016. *Arrival*. Film. Distributed by Paramount Pictures.
- Alex Warstadt and Samuel R. Bowman. 2024. [What artificial neural networks can tell us about human language acquisition](#). *Preprint*, arXiv:2208.07998.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. [Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.
- Jennifer C. White and Ryan Cotterell. 2021. [Examining the inductive bias of neural language models with artificial languages](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 454–463, Online. Association for Computational Linguistics.
- John Wilkins. 1668. *An Essay Towards a Real Character, and a Philosophical Language*. Royal Society, London.
- Ludoviko Lazaro Zamenhof. 1887. *Международный языкъ (International language)*. Типо-Литографія Х. Кельтера, Warsaw.
- Hongpu Zhu, Yuqi Liang, Wenjing Xu, and Hongzhi Xu. 2025. [Evaluating large language models for in-context learning of linguistic patterns in unseen low resource languages](#). In *Proceedings of the First Workshop on Language Models for Low-Resource Languages*, pages 414–426, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

## A Appendix

### A.1 A more detailed discussion of related work

#### A.1.1 Prior work on probing LLMs’ linguistic abilities

Since the advent of transformer-based large language models, there has been a large amount of work probing the linguistic abilities of these models. A lot of this work investigates the abilities of the models on linguistic constructions in particular languages, or probing LLMs to discover how and what linguistic information is represented; some recent work along these lines includes (Nikolaev and Padó, 2023; Katinskaia and Yangarber, 2024; Qiu et al., 2024; Ide et al., 2025; Tjautja et al., 2025). Other work makes bolder claims about LLMs’ linguistic abilities, with Piantadosi (2024) in particular arguing that these models refute Chomsky’s theory of the innateness of grammatical knowledge—and see Kodner et al. (2023) for a response. Given that one clear difference between humans’ and LLMs’ learning of language is the amount of data involved, of particular relevance is work in the BabyLM challenge task (Warstadt et al., 2023) (and see also Warstadt and Bowman (2024)), where language models have been shown to produce competitive performance with much more human-sized amounts of data. Collado-Montañez et al. (2025) argue linguistic competence of LLMs does not scale as steeply with model size as does factual knowledge, and that linguistic competence can be decoupled from LLMs’ more general knowledge, and handled with a smaller model, decoupled from the model of general knowledge. Also related is work such as that of Beuls and Van Eecke (2024) that investigates more plausible language-learning scenarios that involve agents communicating with each other in a simulated social environment more akin to the way humans learn language. Kallini et al. (2024), add further fuel to the fire by arguing that structurally ‘impossible’ languages are much less easily learned by LLMs than are structurally human-like languages, countering a claim by Chomsky et al. (2023) that such models do not distinguish between observed and unobserved linguistic structure, and suggesting that whatever learning biases humans may have, LLMs seem to have them too. Hu et al. (2024) argue that LLMs’ judgments align well with those of humans on some important grammatical constructions (and see Cho (2025b) for recent work on LLMs’

pragmatic competence). López-Otal et al. (2025) present a literature review of 160 research works that seek to probe how linguistic knowledge of various kinds is represented in transformer-based models. There has also been interest in investigating to what degree the representations arrived at by LLMs mirror what one can infer from neurology, for example via fMRI: see Parra (2025) for a recent example. Finally, see Lappin (2024) for a balanced assessment of arguments that have been made about LLMs’ abilities, and lack thereof, and Futrell and Mahowald (2025) for an equally balanced assessment of the implications of LLMs for the field of linguistics.

Of more direct relevance to our work are studies that investigate metalinguistic knowledge, in particular how good language models are at performing tasks that linguists might perform, such as performing syntactic annotation, or correctly marking syntactic arguments in a language that has a particular case-marking pattern. Thus Blevins et al. (2023) investigate the abilities of language models to perform various sorts of structural analysis. Ginn et al. (2024) investigate LLMs’ ability to gloss low-resource languages, showing that without any further training or fine-tuning, such models perform better than transformers trained specifically for these tasks. Buhnla (2025) investigates the ability of smaller (~3B-parameter) models for tasks related to pragmatic implicature. On the phonological side, Suvarna et al. (2024) present the PhonologyBench benchmark that tests LLMs’ abilities at such tasks as rhyme generation or syllable counting, and demonstrate that LLMs show striking performance on many tasks, despite receiving no explicit exposure to phonology, yet at the same time fall short of human abilities in this area. See also Jang et al. (2025), who refine prompting to improve LLMs’ performance on the PhonologyBench benchmark. A related study is Merullo et al. (2025), who investigate the rhyming abilities of Llama-3.2-1B-Instruct, suggesting that the LLM forms an internal representation of phonology that, however, differs in some notable ways from those that humans seem to construct.<sup>4</sup> Cho (2025a) shows that LLMs seem to use phonological cues similar to those used by humans in determining the gender in Korean given names. Dinu et al. (2025) investi-

<sup>4</sup>Needless to say, phonological knowledge can benefit from having access to speech. See de Heer Kloots and Zuidema (2024) for a study of human-like biases in neural speech models.

gate LLMs’ ability to engage in tasks that involve linguistic creativity, such as creating new words based on existing derivational and compounding mechanisms, and argue that LLMs actually outperform humans in many of these tasks. [Saeed et al. \(2025\)](#) investigate LLMs’ abilities at inflection morphology across a wide range of languages showing, not surprisingly, that while such models perform well on English, they do not deal so well with uncommon morphological patterns. They further show Chain-of-Thought and other ‘thinking’ strategies of recent models can actually degrade performance on these tasks. However, whether it is the morphological patterns per se, or merely the frequency with which they occur in the data is unclear: [Pantelidou et al. \(2025\)](#), in a controlled study involving predicting inflected forms of novel words in a variety of languages, argue that the main driver of differences in performance is the amount of data available for the language. Finally, [Beguš et al. \(2025\)](#) investigate LLMs’ abilities at various metalinguistic tasks such as determining if a sentence is ambiguous, or identifying syntactic recursion. They particularly single out OpenAI’s o1 model for vastly outperforming other models, and point to its Chain-of-Thought reasoning abilities as one possible reason, suggesting that such reasoning approaches may be of help in at least some metalinguistic tasks.

[Goyal and Dan \(2025\)](#) and [Choudhary et al. \(2025\)](#) study the performance of LLMs on Linguistics Olympiad puzzles. [Goyal and Dan](#) show that LLMs struggle to handle many types of Olympiad problems, in particular ones that require the participant to extract rules. [Choudhary et al.](#) observe, again unsurprisingly, that the models show a strong English bias in that they do better on Olympiad questions that probe features that happen to coincide with properties of English. Complex morphology also posed problems for models, with the models performing worse on problems that involved larger numbers of morphological features. Models were helped in such problems by making the morpheme boundaries in examples explicit. Some of these conclusions relate to results we report in this paper. However one important difference between what we present and prior work is that our approach to generating ConLangs involves generating abstract English-glossed morphosyntactic annotations, rather than surface forms directly. This allows us to probe the question of how well LLMs understand the various *concepts* underlying differ-

ent morphosyntactic feature settings, abstracting away from particulars of how those settings are marked.

### A.1.2 Prior work on ConLangs

Prior work on testing the linguistic abilities of LLMs has made use of artificially constructed languages. For example [White and Cotterell \(2021\)](#) construct artificial languages specifically designed to allow one to investigate the sensitivity of the LLM to the variation of one feature, such as word order. [Kallini et al. \(2024\)](#) construct ‘impossible’ languages—languages that have features that would be exceedingly unusual in any natural language—to test whether these unusual features affect LLMs’ abilities to learn the languages. And [Marmonier et al. \(2025\)](#) use artificially perturbed versions of real languages, along with grammatical descriptions of those languages to test whether LLMs can benefit from descriptions contained in grammar books, inspired by work of [Tanzer et al. \(2024\)](#) and others. Very recent work by [Swain et al. \(2025\)](#) uses a simple artificial constructed formal language ‘Tinkatongue’ to investigate LLMs’ abilities to adapt to a new language.

But to date, there has been very little research on testing LLMs’ abilities to *construct* ConLangs—this despite a long-term appeal of ConLangs to the general public ([Okrent, 2009](#); [Piperski, 2017](#)). One exception to this is work of [Diamond \(2023\)](#)—who uses the term ‘GenLangs’ in his paper. The author used ChatGPT, and prompted the model to name a new language, then create text in that language, along with an English translation. The model was then asked to write a short story plus its translation. Finally, the model was asked to write as long a story as it could in the language. Diamond examined the grammatical properties of the three created languages, noting that they differed in, for example, how verbs conjugate: in one language verbs were conjugated for tense and aspect as well as mood, whereas in another language verbs did not conjugate at all. One of the languages chose a base-twenty number system, something that occurs in about 21% of the world’s languages ([Comrie, 2013](#)), but would have been significantly *underrepresented* in the training data that the LLM was exposed to. One language chose nominal case prefixes, which is typologically very unusual. Diamond’s main goal though was to examine whether the corpora generated for the GenLangs obeyed the standard ‘Zipf’s Law’ power law distribution.

He argues that English text generated by the LLM has a distribution very close to that of natural text, especially when compared to LLM-generated gibberish.<sup>5</sup> The GenLangs showed more variation, with some languages having distributions close to their English translations, but with one with a decidedly different distribution (though still different from gibberish).

In his conclusions, Diamond notes that ‘practical’ ConLangs of the first type we outlined above (such as Esperanto) took many years of human effort to develop. He contrasts this with the possibility of generating languages using AI: “For the first time in history, we have the opportunity to discover what happens when artificial languages are developed by artificial intelligence.” While Diamond’s work is a proof of concept that creation of an AI-generated ConLang in one fell swoop is a possibility, for reasons we have outlined, in this paper we take a different approach.

As we were preparing this paper, a short paper appeared by Alper et al. (2025) describing their ConlangCrafter system. Their model is similar in some respects to our approach in that, unlike (Diamond, 2023), it breaks the problem down into separate modules of phonology, morphology, syntax, lexicon generation and translation. Their system starts with a high-level description of the language to be generated, which can be as specific or inspecific as the user wants. The system then creates each component according to the specifications, refining each stage with internal feedback to make each component more consistent: for example if the language to be generated should be VSO, an SOV sentence would make the system less consistent than if all sentences are consistently VSO. The output is a system that is then used to translate a set of 10 test sentences from English into the target language. They evaluate the systems for Diversity, Language Consistency and Translation Consistency. *Diversity* is calculated by having the system construct a set of languages, encoding each language as a one-hot vector over WALS features (Comrie, 2013), and computing the pairwise Hamming distance. Note that one of Alper et al. (2025)’s goals is to create a system that allows for as diverse a set of ConLangs as possible.

<sup>5</sup>Note though that LLM-generated texts are well-known to display platykurtosis, exhibiting relatively few outliers compared to natural text (Badurowicz et al., 2024), so that with a large enough amount of text one would be able to detect a difference between natural English, and LLM-generated English.

*Language Consistency*, ranging between 0 and 1, computes the proportion of generated sentences deemed consistent, and *Translation Consistency* similarly measures consistency for the translations. For the largest model they used—DeepSeek-R1 they report improvements in each of these measures compared to a baseline using the same model, where the baseline is “a naive single-stage generation method” that “attempts to generate a full language sketch and translations of given sentences in our format with a single prompt, without multi-hop reasoning or iterative self-refinement” (page 4). For their smaller models, Gemini-2.5-Flash and Gemini-2.5-Pro, the baseline seemed to outperform the more elaborate model at least for the Language Consistency measure—see their Table 1, page 5. While Alper et al. (2025)’s work is obviously related to ours, there are also important differences, particularly when it comes to morphosyntactic feature generation. As we outline in this paper, we adopt a very fine-grained multistage approach to generating morphosyntax, which has the advantage that we can investigate the question of how well LLMs ‘know’ specific morphosyntactic features, abstracting away from other features.

## A.2 A Quick Intro to (Relevant Bits of) Linguistics

A lot of the discussion in this paper assumes some familiarity with linguistics, at least basic terminology and concepts. In this section we introduce concepts that we will assume that the reader will have or will have gained some familiarity with. Further details of most of the terminology that we will assume can be found in standard textbooks such as (Akmajian et al., 2001), and of course there are many good Wikipedia pages that cover the topics in depth. The World Atlas of Language Structures (Dryer and Haspelmath, 2013) gives detailed examples of many ways in which languages vary.

We will assume familiarity with concepts like *phoneme*, *morpheme* and *affix*. In many of the ConLangs, nouns, verbs and adjectives may inflect for things like *case*, *number*, *person*, *tense* and *aspect*. Within each of the above categories, languages make different choices as to what kind of information to mark.

Within case, a fundamental distinction is between languages that are *nominative-accusative* versus those that are *ergative-absolutive*. In the former, any subject of the sentence would be marked in the nominative case, and direct objects would

be marked accusative. In the latter, subjects of *transitive* sentences are marked ergative, whereas subjects of *intransitive* sentences, along with objects, are marked absolutive.

For number, many languages mark a distinction between *singular* and *plural*, but some languages also mark other numbers, such as *dual* for instances involving two referents.

*First, second* and *third person* distinctions are marked in a majority of languages, but languages also make finer grained distinctions. One notable one is in the *first person plural*, which in some languages is divided between *inclusive*—including the hearer—and *exclusive*, not including the hearer.

Languages mark various kinds of tense information on verbs, a common pattern being *past, present* and *future*. But languages may also mark various degrees of remoteness in the past tense (distant past, very distant past, etc.).

These are some of the kinds of variation we are interested in modeling. In any case we introduce examples of phenomena we are modeling where they arise.

### A.3 Phonotactics

#### A.3.1 Modeling phonotactics

The first stage in the construction of a language is to decide what the basic morphemes of the language should look like. Do they involve relatively simple syllables, as in for example Hawaiian where each syllable consists of a single consonant and one or two vowels (cf. well-known Hawaiian names like *Mauna Kea*)? Or does it allow much more complex syllables, as for example Russian, where complex syllables like *vzgl'ad* ‘view’, are common. And what are the basic set of sounds of the language? Again, Hawaiian has just twelve basic sounds, whereas English, for example has on the order of 50. Linguists typically refer to the latter issue as the *phoneme inventory* of the language, and the first as the *phonotactics*.

In this discussion we will conflate both of these under the rubric of *phonotactics*, and present an agentic prompting-based method for determining an initial set of morpheme shapes, and for refining that initial proposal.

We start with a language upon which to model the phonotactics of our ConLang. The initial prompt schema can be found in Appendix A.10.2. The prompt requests the LLM to create a standalone Python program that will generate mor-

phemes that are similar in terms of phoneme inventory and phonotactics to a given target language. The code should include two global dictionaries named *consonants* and *vowels*, which indicate the relative frequencies of the different sounds. There should be a function *generate\_morpheme* which will randomly generate one morpheme that conforms to the phonotactics.

With the target language of Welsh, for example, the following output is created (LLM=Claude Sonnet 3.5):

```

1 import random
2 import argparse
3
4 # Welsh-like phoneme set with relative
5   frequencies
6 consonants = {
7     "p": 3, "t": 5, "k": 4, "b": 2, "d":
8     3, "g": 2,
9     "m": 3, "n": 7, "ŋ": 1, "f": 2, "v":
10    1, "θ": 1,
11    "ð": 1, "s": 4, "ʃ": 2, "h": 2, "χ":
12    1, "l": 4,
13    "r": 3, "j": 2, "w": 2
14 }
15
16 vowels = {
17     "a": 8, "ɛ": 6, "e": 5, "i": 6, "ɪ":
18     4, "o": 5,
19     "o": 5, "u": 4, "ʊ": 3, "ə": 7, "ɨ":
20     2
21 }
22
23 # Morpheme templates (C = consonant, V =
24   vowel)
25 templates = [
26     "CV", "CVC", "CVCC", "CCVC", "CCVCC"
27     ,
28     "CV.CV", "CVC.CV", "CV.CVC", "CVC.
29     CVC", "CCVC.CV", "CV.CCVC"
30 ]
31
32 def choose_phoneme(phoneme_dict):
33     return random.choices(list(
34         phoneme_dict.keys()), weights=list(
35         phoneme_dict.values())[0])
36
37 def generate_morpheme():
38     template = random.choice(templates)
39     morpheme = []
40     for syllable in template.split('.'):
41         syll = []
42         for char in syllable:
43             if char == 'C':
44                 syll.append(
45                     choose_phoneme(consonants))
46             elif char == 'V':
47                 syll.append(
48                     choose_phoneme(vowels))
49             morpheme.append(' '.join(syll))
50     return ' '.join(morpheme)
51
52 def main():
53     parser = argparse.ArgumentParser(
54         description='Generate Welsh-like
55         ConLang morphemes')

```

```

41 parser.add_argument('--num_morphemes
    ', type=int, default=10, help='
    Number of morphemes to generate')
42 args = parser.parse_args()
43
44     for _ in range(args.num_morphemes):
45         print(generate_morpheme())
46
47 if __name__ == "__main__":
48     main()

```

The process is then iterated. The LLM is prompted with the original prompt, the code it produced, and some sample output from the code that it wrote in the first phase (or the error message produced if the code failed). It is also instructed to pay attention to the following:

```

1 There may also be phonemes that are
  particularly characteristic of the
  language
2 that are missing from your previous code
  , or ones that should not be
3 there. Please also check for that.

```

The full listing for a prompt generated for the second phase, with the target language Welsh, can be found in Appendix A.10.3.

This prompt causes the system to improve a number of aspects, most notably adding phonemes that were missing in the first round. A phoneme that is particularly characteristic of Welsh is the voiceless lateral fricative /ɬ/. This was missing from the initial set, but was added in the second:

```

1 import random
2 import argparse
3
4 # Updated Welsh-like phoneme set with
  relative frequencies
5 consonants = {
6     "p": 3, "t": 5, "k": 4, "b": 2, "d":
7     3, "g": 2,
8     "m": 3, "n": 7, "ɲ": 1, "f": 2, "v":
9     1, "θ": 1,
10    "ð": 1, "s": 4, "ʃ": 2, "h": 2, "χ":
11    1, "l": 4,
12    "r": 3, "j": 2, "w": 2, "ɬ": 2 #
13    Added 'ɬ' (voiceless alveolar
14    lateral fricative)
15 }
16
17 vowels = {
18     "a": 8, "ɛ": 6, "e": 5, "i": 6, "i":
19     4, "ɔ": 5,
20     "o": 5, "u": 4, "u": 3, "ə": 7, "i":
21     2, "æ": 3 # Added 'æ'
22 }
23
24 # Updated morpheme templates (C =
  consonant, V = vowel)
25 templates = [
26     "CV", "CVC", "CVCC", "CCV", "CCVC",
27     "CV.CV", "CVC.CV", "CV.CVC", "CVC.
28     CVC", "CCV.CV", "CV.CCV"
29 ]

```

```

22
23 # Valid initial consonant clusters
24 initial_clusters = [
25     "pl", "pr", "bl", "br", "tr", "dr",
26     "kl", "kr", "gl", "gr",
27     "fl", "fr", "θr", "sp", "st", "sk",
28     "sm", "sn", "sf", "ʃr"
29 ]
30
31 # Valid final consonant clusters
32 final_clusters = [
33     "lp", "lb", "lt", "ld", "lk", "lg",
34     "lm", "ln", "lf", "lv", "lθ", "ls",
35     "rp", "rb", "rt", "rd", "rk", "rg",
36     "rm", "rn", "rf", "rv", "rθ", "rs",
37     "mp", "nt", "ɲk", "st", "ft", "θt",
38     "sp", "sk"
39 ]
40
41 def choose_phoneme(phoneme_dict):
42     return random.choices(list(
43     phoneme_dict.keys()), weights=list(
44     phoneme_dict.values()))[0]
45
46 def generate_consonant_cluster(clusters):
47     :
48     return random.choice(clusters)
49
50 def generate_morpheme():
51     template = random.choice(templates)
52     morpheme = []
53     for syllable in template.split('.'):
54         syll = []
55         if syllable.startswith('CC'):
56             syll.append(
57             generate_consonant_cluster(
58             initial_clusters))
59             syllable = syllable[2:]
60             for char in syllable:
61                 if char == 'C':
62                     if syll and syll[-1] in
63                     consonants:
64                         syll.append(
65                         choose_phoneme(vowels))
66                     else:
67                         syll.append(
68                         choose_phoneme(consonants))
69                     elif char == 'V':
70                         syll.append(
71                         choose_phoneme(vowels))
72                         if syllable.endswith('CC'):
73                             syll[-2:] = [
74                             generate_consonant_cluster(
75                             final_clusters)]
76                             morpheme.append(' '.join(syll))
77     return ' '.join(morpheme)
78
79 def main():
80     parser = argparse.ArgumentParser(
81     description='Generate Welsh-like
82     ConLang morphemes')
83     parser.add_argument('--num_morphemes
84     ', type=int, default=10, help='
85     Number of morphemes to generate')
86     args = parser.parse_args()
87
88     for _ in range(args.num_morphemes):
89         print(generate_morpheme())
90
91 if __name__ == "__main__":

```

72 main()

In the second iteration, the system also added initial and final consonant clusters, which were absent from the initial system, providing a `generate_consonant_cluster` function for this purpose. Unfortunately, in so doing, it violated the requirement laid out in the initial instructions: “Within each morpheme, please place spaces between every phoneme.” This error was pointed out in the prompt in the third iteration:

```
1
2 1. The following combined phonemes
   suggest that your code is not
   correctly
3 dealing with spaces between phones.
   Please correct this:
4
5 lf, lm, fl, sm, θr, gl, fr, rb, lθ, lt,
   lg, sf, rp, ft, rn, fr, pl, bl, lk,
   rf,
6 pr, sn, lv, rs, lp, rv, rθ, rd, gr, rm,
   rg, ln, sp, rk, sk, kr, ls
```

The system corrected this in the third iteration by correcting this function

```
1 1. In the `generate_consonant_cluster`
   function, I've added space
   separation
2 between the consonants in the cluster
   :
3 ```python
4 return ' '.join(list(random.choice(
   clusters)))
5 ```
6 This change ensures that all phonemes,
   including those in consonant
   clusters,
7 are properly space-separated.
```

Further iterations refine various other aspects of the model.

Since the goal is to create a language that has a phonotactics that is *similar* to that of the target language, we do not expect that the system will produce an exact match. Nonetheless, if we specify Welsh as the language upon which to model the system, one would hope that the system would produce something that is at least closer to Welsh than, say, Russian. One way to measure this is to compute the perplexity of a sample of the words generated by the system against simple ngram language models trained on Wiktionary data for a variety of languages. Unfortunately this is not a perfect solution since there is a wide range of levels of phonetic transcription in Wiktionary, with data for some languages (e.g. Japanese), showing exceedingly fine-grained phonetic transcriptions, whereas for other languages, such as French, the

transcription is much more coarse-grained. Thus a ‘Japanese’-based phonotactics that produces reasonably plausible output such as the following:

```
tj i . k i
d o
k a a
e
r a u
a
m i . b u
r u e . k i
j e i . s e
s i . s a . k o
s o . d a . n i
```

will nonetheless fail to match well against a model trained on the Wiktionary entries for Japanese. We discuss this directly below in Section A.3.2.

Of course, phonology involves more than phonotactics. In ongoing work, we develop a novel approach to creating plausible phonological rule sets. While this is not yet integrated into the system, we describe the approach in Appendix A.11, and release the code along with the rest of the software.

### A.3.2 Evaluating Phonotactics

If one asks for a language that is like a given target language in terms of its phonotactics, how much like that language is it? One way to measure this is to compare it against a traditional n-gram language model constructed over IPA transcriptions for the target language from Wiktionary. To this end, we used the OpenGrm Ngram library (Roark et al., 2012) to build 3-gram language models for each of 221 languages from Wiktionary. Of these, 49 languages have at least 10,000 entries, and we used this subset in the experiment below. We also filtered by the proportion of OOV symbols, omitting languages for which the OOV rate was 20% or higher.

We then use the final phonotactic model constructed using the procedure described above, and used it to generate 10,000 morphemes from the language, then computed its perplexity given the 3-gram language model. Table 5 shows each of the target languages, the top pick in terms of (lowest) perplexity given the 3-gram language model, and whether or not the correct language occurs in the top ten (lowest ten in terms of perplexity). Four LLMs were used: Claude 3.5 Sonnet, GPT-4o 2024-08-06, Qwen2.5 72B Instruct and Llama 3.3

70B Instruct. As can be seen from the table, for Claude, in 4 out of 10 cases the target language has the lowest perplexity, and in 5 out of ten cases the correct language is within the top 10. The other languages did not fare so well. However, all other LLMs performed worse than Claude.

In some cases the differences can be explained by differences in what the LLM may ‘think’ of as an appropriate level of phonetic transcription, and what the creators of the Wiktionary data for the given language may have thought. For example, noted above in Section A.3.1, the Wiktionary entries for Japanese have exceedingly fine-grained phonetic transcriptions, so that the word *raigetsu* (‘next month’) is transcribed as ‘r a i g e ſ i’ in Wiktionary, but would be transcribed much more phonemically in the system generated by the LLM as ‘r a i . g e t u’.

#### A.4 Stories and grammatically targeted texts

The morphosyntactic component works by transforming English text into the morphosyntactic structure of the target language. For example, as we document in more detail later on, for a language that has singular-dual-plural marking on nouns, the system would annotate SING, DUAL or PLUR on nouns according to whether in the English source sentence has a noun in the singular, the plural but with the meaning of two, or plural with a meaning of more than two.<sup>6</sup> For this purpose we need English texts from which to start, which we generated in an initial data-creation phase.

First, using Claude 3.5 Sonnet, we generated 16 stories, using the story generation prompt presented in Appendix A.10.4. One example of such a story is the following which, among other things, contains an example of a noun *tower*, which should be rendered as DUAL in a language that marks nouns with DUAL:

```

1 The Two Towers
2
3 In a small town, there were two tall
  towers. One
4 was old and made of stone. The other was
  new and
5 made of glass. The people in the town
  liked to
6 talk about which tower was better.
7
```

<sup>6</sup>There are clearly pitfalls to this approach, the most obvious being that terms in one language often do not map one-to-one in their meanings or usages in another. To take an example germane to the current discussion, *scissors* is a plural noun in English (also in Spanish—*tijeras*), but in German it is singular (*Schere*).

```

8 Tom was a young boy who lived in the
  town. He
9 loved to look at both towers. Every day
  after
10 school, he would sit in the park between
  the
11 towers and draw pictures of them.
12
13 One day, a group of kids came up to Tom.
  They
14 asked him which tower he liked more. Tom
  thought
15 for a moment and said, "I like both
16 towers. They're different, but both are
  special."
17
18 The kids laughed at Tom. They said he
  had to pick
19 one. Some kids liked the old stone tower
  because
20 it was strong and had been there for a
  long
21 time. Other kids liked the new glass
  tower because
22 it was shiny and modern.
23
24 Tom felt sad. He didn't want to choose.
  He thought
25 both towers were important to the town.
26
27 That night, there was a big storm. The
  wind blew
28 hard, and rain poured down. In the
  morning, the
29 town woke up to see that both towers had
  been
30 damaged. The old stone tower had cracks,
  and some
31 of the glass on the new tower was broken
  .
32
33 The town mayor called a meeting.
  Everyone argued
34 about which tower to fix first. They
  didn't have
35 enough money to repair both at the same
  time.
36
37 Tom raised his hand and spoke up. He
  said, "Why
38 don't we work together to fix both
  towers? The old
39 tower is our history, and the new tower
  is our
40 future. We need both."
41
42 The people in the town thought about
  what Tom
43 said. They realized he was right. The
  two towers
44 were both important parts of their town.
45
46 So, the townspeople worked together.
  They fixed
47 the old stone tower and the new glass
  tower. It
48 took a long time, but they did it as a
  team.
49
50 When the work was done, the two towers
```

Target	Claude		GPT		Qwen		Llama	
	Top	Top 10	Top	Top 10	Top	Top 10	Top	Top 10
English	<b>English</b>	✓	<b>English</b>	✓	<b>English</b>	✓	Anc. Greek	✓
French	<b>French</b>	✓	Turkish	✓	Turkish	✓	Armenian	✓
German	<b>German</b>	✓	Anc. Greek		English	✓	Maltese	
Hawaiian	Basque		Romanian		Romanian		Armenian	
Japanese	Turkish		Romanian		Romanian		Armenian	
Old English	Turkish		Turkish		Anc. Greek		Turkish	
Russian	French		Turkish		Turkish		Indonesian	
Spanish	Turkish	✓	Romanian		Turkish		Arabic	
Turkish	<b>Turkish</b>	✓	<b>Turkish</b>	✓	Esperanto	✓	Romanian	✓
Welsh	English		Turkish		Catalan		Tagalog	

Table 5: Results comparing lowest perplexity match for ten target language phonotactics against 3-gram language models built on Wiktionary IPA transcriptions for languages with at least 10K Wiktionary entries. Shown are the predicted (lowest-perplexity) language, and whether the correct language is within the top 10. LLMs used were Claude 3.5 Sonnet, GPT-4o 2024-08-06, Qwen2.5 72B Instruct and Llama 3.3 70B Instruct.

```

stood tall
51 again. This time, they had something new
. Between
52 them was a bridge that connected the
towers. Tom's
53 idea had brought the town together.
54
55 From that day on, people stopped arguing
about
56 which tower was better. They understood
that both
57 towers made their town special. And Tom?
He still
58 sat in the park, drawing pictures of the
two
59 towers, now joined by a bridge of
friendship.

```

In addition to these stories, we generated sets of sentences that highlight particular grammatical features. See subsequent prompts in Appendix A.10.4. For example, the following set of sentences gives examples that highlight the first-person-plural inclusive/exclusive distinction, which is of course not marked morphologically in English, but is in many languages:

```

1 1. We should all pitch in to clean up
the park. (INCLUSIVE)
2 2. We're going to the movies later, want
to join us? (INCLUSIVE)
3 3. We've decided to go with a different
contractor for this project. (
EXCLUSIVE)
4 4. Let's work together to solve this
problem, shall we? (INCLUSIVE)
5 5. We're not happy with your performance
lately. (EXCLUSIVE)
6 6. We need to be more careful about
recycling in our office. (INCLUSIVE)
7 7. We've already eaten dinner, but
thanks for the invitation. (
EXCLUSIVE)

```

## A.5 Lexicon creation

After the morphosyntactic modules have run the result are files containing the source sentences and the morphosyntactic annotations. The phonotactic model is then used by the `lexicon.py` module to create lexical entries for each of the morphemes, including stem forms and the various morphosyntactic affixes created during the preceding phase. In the current version, stems have a minimum length in terms of numbers of phonemes (defaulting to 5) and affixes a maximum length, currently set to 3.

Once the lexicon has been created containing all forms, the phonemic transcription of each sentences is added, and the result saved out as a JSON Lines file.

## A.6 Orthography

While most languages throughout history have been unwritten, it is fairly standard for ConLangs to have a written form. In many cases this involves inventing a new script, such as the Elvish Scripts from the Lord of the Rings stories. The writing system might be segmental (as with Elvish) or some sort of morphosyllabic or mixed logographic-phonetic system, like Chinese or Japanese. Inventing new scripts, and proposing logographic scripts both involve some complexity that make them interesting research projects in and of themselves. To create a new script requires a generative image model tuned for creating script-like output, rather than general images.<sup>7</sup> For mixed-logographic scripts, one has to define a manageable set of symbols—on the

<sup>7</sup>There is an additional issue, namely that once the script is created, ideally one would like to develop an encoding system for it so that one can represent it in Unicode, and fonts so that one can render text in it.

order of a few hundred—that can be used singly, or in combination, to represent morphemes, or some non-phonological aspect of the morpheme, such as its meaning.

Our current approach is more modest, namely to have the system create a segmental writing system using a known script and to write a Python program that can convert from IPA transcriptions into the written form. The prompt we use for this is given in Appendix A.10.6. This task still requires some creativity on the part of the LLM, since there are in principle many ways in which a script could be used to represent phonemes in a language. The following listing is an example output where the system (in this case using Claude 3.5 Sonnet) was instructed to produce a Cyrillic-based orthography for a language with Welsh-like phonotactics. Among the interesting choices are the spelling of the voiceless lateral fricative /ɬ/ as ⟨ʃ⟩. This is not part of the core Cyrillic alphabet, as used for Slavic languages such as Russian or Bulgarian: it is an extension that was invented to write the same sound in Chukchi (Chukotka–Kamchatkan language family).<sup>8</sup>

A sample listing is given below:

```
def ipa_to_orthography(ipa_word):
    # Define the mapping from IPA to Cyrillic orthography
    ipa_to_cyrillic = {
        'p': 'п', 't': 'т', 'k': 'к', 'b': 'б', 'd': 'д', 'g': 'г',
        'm': 'м', 'n': 'н', 'ŋ': 'н', 'f': 'ф', 'θ': 'θ', 's': 'с',
        'ʃ': 'ш', 'x': 'х', 'v': 'в', 'ð': 'ð', 'l': 'л', 'r': 'р',
        'w': 'у', 'j': 'й', 'tʃ': 'ч', 'dʒ': 'ж', 'ɬ': 'ʃ', 'h': 'х',
        'ɣ': 'г', 'p': 'п', 'a': 'а', 'ɛ': 'э', 'e': 'е', 'i': 'и',
        'ɪ': 'й', 'o': 'о', 'u': 'у', 'y': 'у', 'ɔ': 'э',
        'ɛ': 'э', 'aɪ': 'ай', 'aʊ': 'ау', 'ɔɪ': 'ой', 'æ': 'æ'
    }

    # Remove syllable boundaries and split into phonemes
    phonemes = ipa_word.replace('-', '').split()

    # Convert phonemes to Cyrillic letters
    cyrillic_word = ''
    i = 0
    while i < len(phonemes):
        # Check for two-phoneme combinations first
        if i < len(phonemes) - 1:
            digraph = phonemes[i] + phonemes[i+1]
            if digraph in ipa_to_cyrillic:
                cyrillic_word += ipa_to_cyrillic[digraph]
                i += 2
                continue

        # Single phoneme conversion
        if phonemes[i] in ipa_to_cyrillic:
            cyrillic_word += ipa_to_cyrillic[phonemes[i]]
        else:
            cyrillic_word += phonemes[i] # Keep unknown symbols as-is
        i += 1

    return cyrillic_word
```

A sample text and transcription are given in Figure 3.

Different LLMs exhibited different choices when it came to spelling given phonemes. For example for the lateral fricative /ɬ/, with Latin script, Claude 3.5 Sonnet consistently chose ⟨lh⟩ as did Qwen2.5-72B-Instruct. GPT-4o-mini also mostly chose this though in one instance it chose ⟨hl⟩. Gemini 2.5 Pro consistently chose ⟨ll⟩, following

Welsh orthographic convention. When rendering in the Cyrillic script, Claude as noted above chose ⟨ʃ⟩. GPT-4o-mini on the other hand represented it with the IPA symbol ⟨ɬ⟩. Qwen2.5-72B-Instruct mostly chose the Cyrillic equivalent of ⟨l⟩ with a circle under it indicating voicelessness, though in one instance it chose the plain Cyrillic equivalent of ⟨l⟩. Gemini 2.5 Pro chose a different Cyrillic letter, ⟨л⟩, which is used for Khanty.<sup>9</sup> Both Gemini and Claude showed greater ‘creativity’ in finding a representation for the /ɬ/ in Cyrillic than did the other two LLMs.

With the orthographic rules in place, the `lexicon.py` module used in creating the phonological forms of the words (Section A.5) is run again, this time converting the phonemic transcriptions into orthographic renditions. These are added to the lexicon as spellings for the stems and affixes, and the corpus of sentences updated with the written forms of the sentences.

One outstanding question is how one evaluates the orthography. The rules generated by the system are consistent in that it produces one-for-one mappings between phonemes and graphemes, leading to a very shallow orthography. But are the mappings reasonable? As noted above, Claude Sonnet 3.5 ‘discovered’ the use of ‘ʃ’ for /ɬ/, which is a perfectly defensible choice. But real orthographies can often make seemingly odd choices. Some of these, such as the use of ⟨gh⟩ in English to represent some instances of post-vocalic /f/, are due to historical change. In other cases the choice was made more recently and cannot obviously be explained in that way. For example, the use of ⟨xh⟩ in Albanian to represent /z/ is a somewhat odd choice when compared against the use of /x/ in most other languages that use the Latin script. While not an orthography per se, but rather a pedagogical aid, the Mandarin phonetic alphabet Hanyu Pinyin for the most part made conventional choices for representing Mandarin segments. Nonetheless, its choice of ⟨q⟩ to represent /tʃʰ/ is odd. It is important to bear in mind that the use of a script in a language’s writing system involves conventional choices for how each symbol in the script is used. By and large, writing systems tend to obey these conventions, but there is no hard and fast rule that they must do so.

<sup>8</sup>[https://en.wikipedia.org/wiki/Chukchi\\_language](https://en.wikipedia.org/wiki/Chukchi_language).

<sup>9</sup>[https://en.wikipedia.org/wiki/El\\_with\\_descender](https://en.wikipedia.org/wiki/El_with_descender)

1. Маушээйр гүтөгһү шусинүни рич глфэх трчэснүайр дуһээйр гечээтуглалцач.  
he-NOM we-GEN school-SING-LOC COMP smart student-SING-NOM the-NOM be-PRES-IND-3SG-ACT  
"He is the smartest student in our school."
2. Гүтөгһүр фһнуу кһчүгһү йакукнүни хилэ блтүкаатофцач строк, кура прһөгһнүайр дуһээйр нулө гечэкһнмүөлалцач.  
we-NOM next week-GEN vacation-SING-LOC on go-PRES-1PL-ACT are, so office-SING-NOM the-NOM closed be-FUT-COND-3SG-ACT  
"We are going on vacation next week, so the office will be closed."
3. Сһбээйр рэүэ римонүңэк тотөкайатуглалцач.  
she-NOM spicy food-SING-ACC like-NEG-PRES-IND-3SG-ACT  
"She doesn't like spicy food."

Figure 3: A sample of a language with Welsh-like phonotactics and a Cyrillic-based orthography.

## A.7 Handbook generation

The final stage involves creating a handbook, basically a short grammatical description of the created language. The template prompt for this stage is given in Appendix A.10.7. The LLM is given the phonology, breaking down the phonemes into consonants and vowels, as is standard practice in grammar writing. Next the model is given the code that generates the orthography. A sample of the lexicon is presented, and morphosyntactic settings along with reminders about what these mean. A short set of sample texts are then presented. The specification of the format of the handbook is given last.

Note that the very first instruction given is to create a name for the language. As we see in the next section, there are some interesting biases exhibited by the LLM in its choice of names. An example handbook for “Kethévik”, an ergative-absolutive language with Turkish-like features based on Welsh phonotactics and using the Roman script is given below:

Kethévik handbook
TITLE: A Grammar and Handbook of Kethévik
PHONOLOGY
Kethévik has a rich phonological inventory → with 26 consonants and 15 vowels:
Consonants: p, t, k, b, d, g, m, n, ŋ, f, → θ, s, ʃ, χ, v, ð, l, r, w, j, tʃ, dʒ, ʎ, → h, ʎ̥, r̥
The language features several uncommon → consonants, including the voiceless → lateral fricative /ʎ/ and its devoiced → counterpart /ʎ̥/, as well as the → voiceless uvular fricative /χ/. The → presence of both /r/ and /r̥/ → (voiceless alveolar trill) is also → noteworthy.
Vowels: a, ε, e, i, i, ɔ, o, u, ʊ, ə, i, → ai, au, ɔi, æ

The vowel system is quite extensive,  
→ including both monophthongs and  
→ diphthongs. The presence of both tense  
→ and lax vowels (e.g., /i/ vs /i/, /u/  
→ vs /ʊ/) allows for fine distinctions  
→ in vowel quality.

### ORTHOGRAPHY

Kethévik uses a modified Latin alphabet.  
→ Most consonants are represented  
→ straightforwardly, with a few  
→ exceptions:

- /ŋ/ is written as 'ng'
- /θ/ is written as 'th'
- /ʃ/ is written as 'sh'
- /χ/ is written as 'kh'
- /ð/ is written as 'dh'
- /tʃ/ is written as 'ch'
- /dʒ/ is written as 'j'
- /ʎ/ and /ʎ̥/ are both written as 'lh'
- /r̥/ is written as 'rh'

Vowels use diacritics to distinguish  
→ between similar sounds:

- /e/ is written as 'é'
- /o/ is written as 'ó'
- /i/ is written as 'i'
- /ʊ/ is written as 'ü'
- /ə/ is written as 'ë'
- /i/ is written as 'y'

The diphthongs /ai/, /au/, and /ɔi/ are  
→ written as 'ai', 'au', and 'oi'  
→ respectively. The vowel /æ/ retains  
→ its IPA symbol in the orthography.

### MORPHOSYNTAX

Kethévik is a highly agglutinative  
→ language with a rich system of  
→ suffixes. Nouns are marked for number  
→ (singular -nū, plural -rëp) and case.  
→ The case system includes:

- Nominative: -aurh
- Accusative: -ngek
- Dative: -éw
- Genitive: -dhü
- Ablative: -un
- Locative: -hi
- Instrumental: -réng

Verbs are marked for tense (present -at, ↪ past -rī, future -kon), mood ↪ (indicative -ug, conditional -müth), ↪ person, and number. They also ↪ distinguish between active (-jach) and ↪ passive (-mó) voice. Negation is ↪ marked by the suffix -kai.

Other notable morphological features  
↪ include:  
- Comparative: rich  
- Superlative: ain  
- Relative clause marker: -wé

#### SYNTAX

Kethévik follows an SOV  
↪ (Subject-Object-Verb) word order.  
↪ Adjectives precede the nouns they  
↪ modify (AN), while adpositions follow  
↪ their noun phrases (NP).

Example of basic word order:  
Krchiraurh linunügek rütəriuglaljach.  
I-NOM book-ACC write-PAST-IND-3SG-ACT  
"I wrote the book."

Modifiers typically come before the noun:  
Sklash linunüaurh dudheaurh  
↪ gécheatuglaljach.  
new book-NOM the-NOM be-PRES-IND-3SG-ACT  
"The new book exists."

Postpositions are used instead of  
↪ prepositions:  
Shusinühi rhibé  
school-LOC in  
"in the school"

#### EXAMPLE TEXT

Here are a few examples from the provided  
↪ text:

1. Mauhëaurh gwtogdhü shusinühi rich  
↪ glfeh trchesnüaurh dudheaurh  
↪ gécheatuglaljach.  
he-NOM we-GEN school-SING-LOC COMP  
↪ smart student-SING-NOM the-NOM  
↪ be-PRES-IND-3SG-ACT  
"He is the smartest student in our  
↪ school."
2. Shawaaurh krthannüdhü dudhedhü maula  
↪ kura yaimé ain thrkêlh khómînüaurh  
↪ dudheaurh gécheatuglaljach.  
today-NOM year-SING-GEN the-GEN of so  
↪ far SUP hot day-SING-NOM the-NOM  
↪ be-PRES-IND-3SG-ACT  
"Today is the hottest day of the year  
↪ so far."
3. Prpivrëpaurh dudheaurh sptophi  
↪ rhóbirëpun dudheun stchés  
↪ gwfofatugoljach.  
leaf-PLUR-NOM the-NOM autumn-LOC  
↪ tree-PLUR-ABL the-ABL from  
↪ fall-PRES-IND-3PL-ACT

"The leaves fall from the trees in  
↪ autumn."

#### LEXICON

A small sample of the Kethévik lexicon:

bltük (v.) - to go  
dudhe (det.) - the  
géche (v.) - to be  
gwtog (pron.) - we  
khómī (n.) - day  
krchir (pron.) - I  
linu (n.) - book  
maushë (pron.) - he  
rütæ (v.) - to write  
shusi (n.) - school  
sklash (adj.) - new  
snbew (pron.) - she  
thrkêlh (adj.) - hot

### A.7.1 Biases in language name choice

As noted above, when creating the handbook, the system was instructed to create a name for the Con-Lang. Interestingly, though ultimately not surprisingly, the system's choices were heavily biased by the script chosen for the language's orthography. See Table 6. The tendencies seem clearest for Claude, but can be found for other models too.

The data in Table 6 were derived by creating 117 handbooks using Gemini-2.6-Pro, 126 using Claude 3.5 Sonnet, 122 using Qwen2.5-72B-Instruct and 125 using GPT-4o-mini. The reason for the difference in numbers relates to the fact that some models are better at others at creating working code to generate the orthographies. In this case we ran the handbooks with 4 base phonotactics—Japanese, Welsh, French, Spanish; 4 morphosyntactic bundles—turkish, arabic, mizo and hixkaryana; and 4 scripts—Cyrillic, Arabic, Latin, Greek. All of these were generated using Claude. Then for each of the 64 combinations, the four LLMs were used to create an orthography and a handbook. Since some of the LLMs failed in several cases, either at the orthography phase, or at the handbook generation phase, we reran a second time, this time using Claude's orthographies, and just using the four LLMs to create the handbooks.

When the Latin script was chosen, the names chosen had a fantasy feel to them—the kind of name one might associate with mythical races in fantasy novels: *Seran*, *Althar*, *Sérulang*, *Nani*. For Cyrillic, the names look decidedly more 'Slavic' or at least Central Asian, though sometimes 'boring' names like *Cyrillic Ergative*, *Cyrillic* or variants like *Kyrrilic* are chosen. For Arabic script, in con-

trast, the names sound a bit Afghan or Persian. Finally when the Greek script was chosen, the names were all formed from Greek roots or at least roots that look Greek. Note that the names chosen by Qwen were actually presented in Greek script, but have been transliterated in the table.

These choices clearly reflect the biases that are present in the data that the LLM will have been exposed to. The Latin script is the most widely used script in the world today, being used to write languages on five continents. It is therefore hard to localize it to any particular area. Presumably the choice of fantasy-sounding names reflects an association between ConLangs and the world of fantasy fiction.

Cyrillic on the other hand, is much more localized, being used to write languages of Eastern Europe and Central Asia, in particular for languages spoken in the area of the former Soviet Union. Thus names like *Krylovik* which sounds decidedly Slavic, or *Kherzovan*, which sounds ‘Central Asian’.

Arabic is somewhat less localized than Cyrillic historically, having been used historically to write the languages of Muslim populations in many parts of the world—see the introduction of [Doctor et al. \(2022\)](#) for some discussion—but today is largely associated with Western Asia and North Africa, and is particularly associated with major languages like Arabic, Persian, Urdu, Panjabi and Sindhi. Hence names like *Shadhiri* or *Khorasani*.

Finally, Greek is the exception among these four scripts in that while there is no principled reason it could not have been adapted to other languages,<sup>10</sup> in fact it is *only* used to write Greek. Hence the bias towards distinctly Greek-sounding names.

This conflation of script and language is perhaps unfortunate, but at the same time it reflects a similar conflation that even professional computational linguists frequently make, as documented in [Gorman and Sproat \(2022\)](#).

## A.8 Translation of new texts

Based on the grammatical information created in the morphosyntactic component, the handbook, and a sample of interlinear gloss translations, the system can then be prompted to provide translations for new texts. The template prompt for this stage is given in Appendix A.10.8. As before, the “translations” are really interlinear glosses, and after this translation is performed, the lexicon cre-

<sup>10</sup>And indeed it was the parent of multiple other scripts, including Latin and Cyrillic.

ation module described in Sections A.5 and A.6 are called to augment the lexicon as needed with new entries and their orthographic forms.

A couple of example translations for a language called Zashuni based on arabic (VSO order), with dual number marking, using Japanese-like phonotactics and Latin orthography is given below. The LLM used was Claude 3.5 Sonnet. Note the correct translation of *two other chicks* into dual number marking, as well as *both towers* into the dual, even though *both* had not occurred in the original set of example sentences used to create the morphosyntax. However the use of the plural in the following sentence on the adjective *different*, which probably should be dual, suggests that there is room for improvement.

### Zashuni translation

1.

Sarah knew she had to help. She climbed  
 ↪ the tree carefully, holding the baby  
 ↪ bird close. When she reached the nest,  
 ↪ she saw two other chicks waiting for  
 ↪ their sibling. Sarah placed the  
 ↪ lost baby back with its family and  
 ↪ climbed down.

know-PAST-3SG-ACT Sarah-NOM she-NOM  
 ↪ have-PAST-3SG to help-ACT .  
 ↪ climb-PAST-3SG-ACT she-NOM  
 ↪ DEF-tree-SING-ACC carefully ,  
 ↪ hold-PRES-3SG-ACT DEF-bird-SING REL  
 ↪ baby-SING-ACC-DEF close . when  
 ↪ reach-PAST-3SG-ACT she-NOM  
 ↪ DEF-nest-SING-ACC , see-PAST-3SG-ACT  
 ↪ she-NOM chick-DUAL-ACC REL  
 ↪ other-DUAL-ACC wait-PRES for  
 ↪ sibling-SING their-SING-ACC-DEF .  
 ↪ place-PAST-3SG-ACT Sarah-NOM  
 ↪ DEF-baby-SING REL lose-PASS-ACC-DEF  
 ↪ back with family-SING its-SING-ACC-DEF  
 ↪ and climb-PAST-3SG-ACT down .

ta.hu-han-i-i Sarah-o teso-o zecu-han-i  
 ↪ gogo wo.to-i . po.ze-han-i-i teso-o  
 ↪ cu-gimo-o-u ro.ton , ku.hu-sa-i-i  
 ↪ cu-gu.ka-o a sa.ke-o-u-sa dino . teti  
 ↪ jo.pi-han-i-i teso-o cu-mo.hu-o-u ,  
 ↪ mehi-han-i-i teso-o sa.zu-e-u a  
 ↪ ra.mu-e-u cihi-sa tasi ga.ki-o  
 ↪ shacu-o-u-sa . he.me-han-i-i Sarah-o  
 ↪ cu-sa.ke-o a po.tʃu-u-u-sa ta.ni raca  
 ↪ we.ra-o yega-o-u-sa ruce po.ze-han-i-i  
 ↪ ke.do .

	Gemini-2.5-Pro		Claude 3.5 Sonnet		Qwen2.5-72B-Instruct		GPT-4o-mini	
	Name	Count	Name	Count	Name	Count	Name	Count
Latin	Seran	3/30 (0.10)	Zohulang	2/32 (0.06)	Althar	3/28 (0.11)	Zaru	2/32 (0.06)
	Véronien	2/30 (0.07)	Zashuni	2/32 (0.06)	Alorin	3/28 (0.11)	Yonu	2/32 (0.06)
	Sotari	2/30 (0.07)	Thrlangai	2/32 (0.06)	Lirin	2/28 (0.07)	Nani	2/32 (0.06)
	Zanari	1/30 (0.03)	Sérulang	2/32 (0.06)	Érhláth	1/28 (0.04)	Frchesh	2/32 (0.06)
	Véronel	1/30 (0.03)	Sénalang	2/32 (0.06)	Élvarin	1/28 (0.04)	Zohari	1/32 (0.03)
Greek	Kallisti	9/30 (0.30)	Zethalian	2/30 (0.07)	Kyrinian	5/31 (0.16)	Lurian	2/31 (0.06)
	Aethelian	6/30 (0.20)	Zetako	2/30 (0.07)	Synthetike	1/31 (0.03)	Kirin	2/31 (0.06)
	Korythian	4/30 (0.13)	Neosyllabic	2/30 (0.07)	Skylan	1/31 (0.03)	Kysar	2/31 (0.06)
	Zorani	1/30 (0.03)	Neolexia	2/30 (0.07)	Phrantik	1/31 (0.03)	Koinos	2/31 (0.06)
	Rhovanic	1/30 (0.03)	Neolekta	2/30 (0.07)	Phragmatiko	1/31 (0.03)	Zmthif	1/31 (0.03)
Arabic	Zarani	5/28 (0.18)	Thulāni	2/32 (0.06)	Alathian	5/31 (0.16)	Koshari	3/31 (0.10)
	Zarish	3/28 (0.11)	Thraenian	2/32 (0.06)	Alarabiya	5/31 (0.16)	Ghrin	2/31 (0.06)
	Zarathian	2/28 (0.07)	Sitara	2/32 (0.06)	Aramith	2/31 (0.06)	Zukari	1/31 (0.03)
	Kaskari	2/28 (0.07)	Shalāni	2/32 (0.06)	Aralish	2/31 (0.06)	Zohor	1/31 (0.03)
	Zulkhari	1/28 (0.04)	Shalāmik	2/32 (0.06)	Aralang	2/31 (0.06)	Zaru	1/31 (0.03)
Cyrillic	Vesnyan	3/29 (0.10)	Cyrillic Ergative	6/32 (0.19)	Kyrillic	9/32 (0.28)	Zohun	2/31 (0.06)
	Kherzovan	2/29 (0.07)	Cyrillic	5/32 (0.16)	Varnan	5/32 (0.16)	Zaru	2/31 (0.06)
	Zoryanin	1/29 (0.03)	Krylovik	3/32 (0.09)	Krylthar	5/32 (0.16)	Lurian	2/31 (0.06)
	Zhalian	1/29 (0.03)	Vornese	2/32 (0.06)	Kryslan	3/32 (0.09)	Lodi	2/31 (0.06)
	Voryenian	1/29 (0.03)	Koshanese	2/32 (0.06)	Vrylak	1/32 (0.03)	Kirin	2/31 (0.06)

Table 6: Top five language names chosen by each LLM for each of the four scripts..

Tahunanii Saraho tesoo zecuhani gogo wotoi  
 ↳ pozehanii tesoo cugimoou roton,  
 ↳ cuhusaii cugucao a saceousa dino teti  
 ↳ shopihanii tesoo cumohuou, mehihanii  
 ↳ tesoo sazueu a ramueu cihisa tasi  
 ↳ gacio shacuoua hemehanii Saraho  
 ↳ cusaceo a pochuuusa tani raca werao  
 ↳ yegaoua ruce pozehanii cedo.

2.

One day, a group of kids came up to Tom.  
 ↳ They asked him which tower he liked  
 ↳ more. Tom thought for a moment and  
 ↳ said, "I like both towers. They're  
 ↳ different, but both are special."

day-SING one-SING-GEN-INDEF ,  
 ↳ come-PAST-3PL-ACT group-SING REL  
 ↳ kid-PLUR-GEN-INDEF up to Tom .  
 ↳ ask-PAST-3PL-ACT they-NOM he-ACC which  
 ↳ tower-SING like-PAST-3SG-ACT he-NOM  
 ↳ more-SING-ACC . think-PAST-3SG-ACT Tom  
 ↳ for moment-SING-ACC-INDEF and  
 ↳ say-PAST-3SG-ACT , " \_  
 ↳ like-PRES-1SG-ACT i-NOM both  
 ↳ tower-DUAL-ACC . be-PRES-3PL they-NOM  
 ↳ different-PLUR-NOM-INDEF , but  
 ↳ be-PRES-3PL both  
 ↳ special-PLUR-NOM-INDEF . "

deayi-o cata-o-nin-ra , sa.ha-han-nen-i  
 ↳ ke.zo-o a ru.to-hu-nin-ra nu.wi gogo  
 ↳ Tom . bo.be-han-nen-i zate-o maco-u  
 ↳ te.ka je.de-o muta-han-i-i maco-o  
 ↳ ha.ta-o-u . ra.ro-han-i-i Tom tasi  
 ↳ hu.ha-o-u-ra ruce si.ge-han-i-i , " \_  
 ↳ muta-sa-u-i dabi-o hi.so je.de-e-u .  
 ↳ zaru-sa-nen zate-o toyato-hu-o-ra ,  
 ↳ tude zaru-sa-nen hi.so be.sa-hu-o-ra .  
 ↳ "

Deayio cataoninra, sahananni cezoo a  
 ↳ rutohuninra nuwi gogo Tom bobehanneni  
 ↳ zateo macou teca yedeo mutahanii macoo  
 ↳ hataou rarohanii Tom tasi huhaoura  
 ↳ ruce sigehanii, "mutasau dabiyo hiso  
 ↳ yedeu zarusanen zateo toyatohuora,  
 ↳ tude zarusanen hiso besahuora. "

## A.9 Applications to Low-Resource Languages

LLMs' performance on linguistic tasks depends on their being exposed to large amounts of training data, but this immediately brings up the question of what these models can do to help with low-resource languages which, by definition, have very little in the way of training data.

One piece of work which attracted a lot of attention was the *Machine Translation from One Book* project (Tanzer et al., 2024), which purported to build a machine translation system by presenting a single grammar book of a low-resource language, the Trans-New-Guinean language Kalamang, as part of the prompt to a version of Gemini. In principle the LLM might have 'read' and 'understood' the grammar presentation, and used that knowledge, along with the examples presented in the book, to construct an understanding of Kalamang grammar, which it could then use to translate novel examples. Aycock et al. (2025) argue that, in fact, the system very likely picked up whatever it learned about Kalamang from the examples, rather than from any understanding of the grammatical description.

But no matter. It is still an interesting question to see to what extent LLMs can be used to help with low-resource languages. Such a project falls within the scope of what one might call the "AI Linguist", which is modeled on the term "AI Scientist" (Lu et al., 2024). The AI Linguist is an

artificial system capable of behaving like a trained field linguist, able to assemble disparate pieces of information about a language, and synthesize them into an understanding of how that language works. An example of such an approach is that of [Torrent et al. \(2024\)](#), who use LLMs as a linguist’s aid in research in Construction Grammar.

One possible application of the ConLang approach we have presented here is to improving translation from high-resource languages into low-resource languages. As long as the LLM has been exposed to enough data from a low-resource language, translation from that language into a high-resource language can often produce reasonable results. This is simply because the LLM has a good model of the target language and so can construct sensible translations given the meanings of the words in the source language. Going in the other direction is another matter, since the LLM will not typically have a very good model of the target language.

One possible approach to improving translation into low-resource languages is to distill knowledge about the language into a set of transformations that can modify the high-resource source into a structure similar to that of the target language. For example if translating the English sentence “John ate cucumbers” into an SOV Ergative-Absolutive language with overt case suffixes, one might first transform the sentence into something like “John-ERG cucumber-PL-ABS eat-PAST”. Approaches along these lines are not new: [Ray \(2025\)](#) discusses one of the first “mechanical translation” systems by [Pucci, 1931](#)), which involved annotating grammatical information on source sentences to aid in the translation to the target language. But it is also a natural extension of the approach to ConLangs presented here.

We present here a simple experiment along these lines for translating from English into Ainu, a severely endangered language isolate of Hokkaido, Japan. While the results are largely negative, we present some evidence that with improvements to the ConLang morphosyntactic approach we have presented here, one could achieve actual gains.

The basic idea is to improve translation from English into Ainu, by annotating the English source sentence to look more like the target language, using the approach developed in Section 4. The first task is to create a set of morphosyntactic specifications appropriate for Ainu. For this we used NotebookLM (<https://notebooklm.google.com/>)

to summarize important Ainu inflection affixes from the grammatical description in [Nakagawa \(2024\)](#).<sup>11</sup> The resulting table was then fed to Claude Code (<https://docs.claude.com/en/docs/claude-code/sdk/sdk-overview>), along with our morphosyntax specifications described earlier, with an instruction to create a similar specification tailored to Ainu. The result was then hand-checked and modified in cases where the automatically generated specifications deviated from our own understanding of Ainu grammar.

For translation material we chose 50 example sentences from the Ainu textbook by [Nakagawa \(2013\)](#), where we translated the original Japanese sentence into English. Note that there are also online Ainu lessons in English such as <https://unilang.org/course.php?res=58&subid=1>, which provide reasonable translation pairs, but given our experience with using these online materials, we suspect data leakage.

We used two LLMs, Claude Sonnet 3.5 and Gemini 2.5 Pro to annotate the English sentences using the morphosyntactic component of our system. In addition to these two annotations, we also prepared by hand a very literal back translation from Ainu into English of the target Ainu sentence. Our purpose in doing this is to test the limits of the proposed approach by providing a version of English sentence that is close to an ideal modification towards the target-language structure. An example of an Ainu sentence, its English translation, and the three modifications of the English sentence is given in Table 7. Note that both Claude and Gemini render rather sparse annotations, with Claude in particular omitting the possessive/verbal second-singular prefix. Ainu has relatively sparse inflection, and this result mirrors our observation that the models generally do better when there is more information that they need to mark.

We then used Claude Sonnet 3.5 to translate from English to Ainu, under two conditions: with the English source sentence alone, and the English source sentence coupled with one of the annotations. We ran this five times for each of the three annotation methods. Corpus bigram, trigram and tetragram BLEU scores are shown in Table 8. As before, Claude performs better than Gemini on average, but translating with the Claude annotation still yields lower BLEU scores than translating with the En-

---

<sup>11</sup>We thank Hiroshi Nakagawa for kindly allowing us to use the PDF of his grammar for research purposes.

Ainu	makanak e=iki wa e=teke arka?
English	How did you hurt your hand?
Claude Sonnet 3.5	you ACT-hurt your hand-SING how hurt-PAST?
Gemini 2.5 Pro	you your hand how 2-SG-hurt?
Hand-annotated	how 2SGNOM-do and 2SG-hand hurt?

Table 7: Ainu example sentence with translation and morphosyntactic glossing from various systems.

System	Condition	Bigram BLEU	Trigram BLEU	Tetragram BLEU
Claude	Unannotated	<b>18.15</b>	<b>11.71</b>	<b>7.69</b>
Claude	Annotated	15.46	10.06	5.87
Gemini	Unannotated	<b>18.40</b>	<b>11.11</b>	<b>6.27</b>
Gemini	Annotated	13.25	7.29	3.82
Hand	Unannotated	17.00	10.29	5.99
Hand	Annotated	<b>23.17</b>	<b>15.23</b>	<b>10.36</b>

Table 8: BLEU scores for each of the three annotation conditions, averaged over five runs.

glish source alone. On the other hand, the hand annotated data shows a dramatic reversal, with the annotation yielding a substantial improvement in BLEU scores over just using the English source alone.

Clearly our current morphosyntactic annotation system is not yet adequate to the task of aiding translation for low-resource languages. But the results with the hand annotation suggest that an ideal future annotation system could yield measurable gains in low-resource language translation. An area of future research is improvement of the morphosyntactic annotation: the current output ConLangs are still too parasitic on English, so development of methods to make the annotations less English-like is desirable in any case.

## A.10 Prompts

### A.10.1 System Prompt

We prefixed the user prompts below with the following:

You are an expert linguist who is also an expert Python programmer. You know a wide variety of languages, and your hobby is creating Constructed Languages — ConLangs.

### A.10.2 Phonotactics Prompt

#### Phonotactics prompt

The first phase in creating any ConLang is  
 ↪ to create a set of words for the language. Typically this involves defining  
 ↪ the phoneme set, including some information about their relative frequency  
 ↪ as well as their phoneme classes (consonant, vowel, tone) and phonetic  
 ↪ features.

Then one needs to define a set of morpheme  
 ↪ templates, which define possible morpheme shapes. This could be as simple  
 ↪ as "CVC" for a language that allows CVC syllables, to something more complex like  
 ↪ "CCVC.CCVC" for languages that allow disyllabic morphemes with each syllable of  
 ↪ the form "CCVC".

We are going to create a language that has  
 ↪ a phoneme set and morpheme shapes that are similar to those of {{ language  
 ↪ }}. In this first phase, I want you to design a set of phonemes similar to those  
 ↪ of {{ language }}, with similar relative frequencies for the phonemes to  
 ↪ those of that language. All phonemes should be represented in the International  
 ↪ Phonetic Alphabet (IPA).

Then you will design a set of morpheme  
 ↪ templates similar to the patterns found  
 ↪ in {{ language }}.

Your design should be implemented as a  
 ↪ standalone Python program that takes  
 ↪ as a

command-line argument the number of  
↳ morpheme forms to generate. For  
↳ example:

```
python3 your_program.py --num_morphemes  
↳ 100
```

would randomly generate 100 morphemes.

Your code should have a function  
↳ `generate_morpheme` that takes no  
↳ arguments,  
and returns a single morpheme.

There are two main points to pay attention  
↳ to.

1) Within each morpheme, please place  
↳ spaces between every phoneme

2) Mark internal syllable boundaries with  
↳ ".", again separated by spaces.

For example, the following is a valid  
↳ output.

```
"o . k a . tʃ a"
```

The following would be wrong, because the  
↳ phonemes are not space-separated:

```
"o.ka.tʃa"
```

Your code should include two global  
↳ dictionaries, one named "consonants"  
↳ and the  
other named "vowels". These should contain  
↳ the phonemes mapped to their relative  
frequencies. Thus for example, a partial  
↳ consonant dictionary might include:

```
consonants = {"m": 3, "n": 7 ...}
```

Please make sure you include needed  
↳ functions and imports in your code.

Finally, place your resulting code in the  
↳ block `<OUTPUT></OUTPUT>`.  
PLEASE DO NOT PUT ANYTHING EXCEPT THE  
↳ PYTHON CODE WITHIN THIS BLOCK.

This task will depend on your deep  
↳ knowledge of linguistics, and Python  
programming. Good luck!

### A.10.3 Phonotactics Improvement Prompt

The following is an example of a prompt generated for the second round of the agentic phonotactic system for a Welsh-based phonotactics.

#### Phonotactics improvement prompt

The following are the instructions you  
↳ were originally given:

```
<ORIGINAL_INSTRUCTIONS>
```

The first phase in creating any ConLang is  
↳ to create a set of words for the  
language. Typically this involves defining  
↳ the phoneme set, including some  
information about their relative frequency  
↳ as well as their phoneme classes  
(consonant, vowel, tone) and phonetic  
↳ features.

Then one needs to define a set of morpheme  
↳ templates, which define possible  
morpheme shapes. This could be as simple  
↳ as "CVC" for a language that allows CVC  
syllables, to something more complex like  
↳ "CCVC.CCVC" for languages that allow  
disyllabic morphemes with each syllable of  
↳ the form "CCVC".

We are going to create a language that has  
↳ a phoneme set and morpheme shapes  
that are similar to those of Welsh. In  
↳ this first phase, I want you to  
design a set of phonemes similar to those  
↳ of Welsh, with similar  
relative frequencies for the phonemes to  
↳ those of that language. All phonemes  
should be represented in the International  
↳ Phonetic Alphabet (IPA).

Then you will design a set of morpheme  
↳ templates similar to the patterns  
↳ found  
in Welsh.

Your design should be implemented as a  
↳ standalone Python program that takes  
↳ as a  
command-line argument the number of  
↳ morpheme forms to generate. For  
↳ example:

```
python3 your_program.py --num_morphemes  
↳ 100
```

would randomly generate 100 morphemes.

Your code should have a function  
↳ `generate_morpheme` that takes no  
↳ arguments,  
and returns a single morpheme.

There are two main points to pay attention  
↳ to.

1) Within each morpheme, please place  
↳ spaces between every phoneme

2) Mark internal syllable boundaries with  
↳ ".", again separated by spaces.

For example, the following is a valid  
↳ output.

```
"o . k a . tʃ a"
```

The following would be wrong, because the  
↳ phonemes are not space-separated:

```
"o.ka.tʃa"
```

Your code should include two global  
 ↪ dictionaries, one named "consonants"  
 ↪ and the  
 other named "vowels". These should contain  
 ↪ the phonemes mapped to their relative  
 frequencies. Thus for example, a partial  
 ↪ consonant dictionary might include:

```
consonants = {"m": 3, "n": 7 ...}
```

Please make sure you include needed  
 ↪ functions and imports in your code.

Finally, place your resulting code in the  
 ↪ block <OUTPUT></OUTPUT>.  
 PLEASE DO NOT PUT ANYTHING EXCEPT THE  
 ↪ PYTHON CODE WITHIN THIS BLOCK.

This task will depend on your deep  
 ↪ knowledge of linguistics, and Python  
 programming. Good luck!

</ORIGINAL\_INSTRUCTIONS>

Here is the code from your last round:

```
<CODE>
import random
import argparse

# Welsh-like phoneme set with relative
↪ frequencies
consonants = {
    "p": 3, "t": 5, "k": 4, "b": 2, "d": 3,
    ↪ "g": 2,
    "m": 3, "n": 7, "ŋ": 1, "f": 2, "v": 1,
    ↪ "θ": 1,
    "ð": 1, "s": 4, "ʃ": 2, "h": 2, "χ": 1,
    ↪ "l": 4,
    "r": 3, "j": 2, "w": 2
}

vowels = {
    "a": 8, "ɛ": 6, "e": 5, "i": 6, "i": 4,
    ↪ "ɔ": 5,
    "o": 5, "u": 4, "ʊ": 3, "ə": 7, "ɨ": 2
}

# Morpheme templates (C = consonant, V =
↪ vowel)
templates = [
    "CV", "CVC", "CVCC", "CCVC", "CCVCC",
    "CV.CV", "CVC.CV", "CV.CVC", "CVC.CVC",
    ↪ "CCVC.CV", "CV.CCVC"
]

def choose_phoneme(phoneme_dict):
    return
    ↪ random.choices(list(phoneme_dict.keys()),
    ↪ weights=list(phoneme_dict.values()))[0]

def generate_morpheme():
    template = random.choice(templates)
    morpheme = []
    for syllable in template.split('.'):
        syll = []
        for char in syllable:
            if char == 'C':
```

```
        syll.append(choose_phoneme(consonants))
    elif char == 'V':
        syll.append(choose_phoneme(vowels))
    morpheme.append(''.join(syll))
    return ' '.join(morpheme)
```

```
def main():
    parser =
    ↪ argparse.ArgumentParser(description='Generate
    ↪ Welsh-like ConLang morphemes')
    parser.add_argument('--num_morphemes',
    ↪ type=int, default=10, help='Number
    ↪ of morphemes to generate')
    args = parser.parse_args()

    for _ in range(args.num_morphemes):
        print(generate_morpheme())

if __name__ == "__main__":
    main()
</CODE>
```

Here are some outputs from your last  
 ↪ phonotactic model:

```
m a . p ə
r ə s . d ō v
ð e . r ɛ ʃ
f ə . b i
ʃ e r
n χ e ŋ j
g p ɛ m
w i h . p ə b
w e n ʃ
p t ɛ p n
w ə f . h a s
n e n k
r ɔ b . ŋ ɛ m
n s i f . k i
s e n h
g i . n l u d
t f i χ . n u
r ō . k e n
k ə . h ō
l i m f
```

Apart from correcting any errors noted  
 ↪ above, please also consider the  
 ↪ outputs  
 from the last round to see if your  
 ↪ phonotactic rules could be improved.

In particular please check this output and  
 ↪ see if there are any words that look  
 phonotactically odd. Are there words or  
 ↪ syllables that begin with  
 rather unlikely consonant sequences? For  
 ↪ example, cross-linguistically, the  
 following syllables have strange onsets  
 ↪ since they begin with consonant  
 sequences, or end with consonant sequences  
 ↪ that are unlikely onsets or codas  
 given the Sonority Hierarchy:

```
p d ɛ d p
m d ɔ v t
t n ɔ t
```

Also check your syllable boundary  
 ↪ assignments.

There may also be phonemes that are  
↪ particularly characteristic of the  
↪ language  
that are missing from your previous code,  
↪ or ones that should not be  
there. Please also check for that.

As before, place your resulting code in  
↪ the block <OUTPUT></OUTPUT>.

ONLY if no errors are noted, and if your  
↪ assessment of the rules is that they  
are adequate, then instead of code return  
↪ the following:

```
<OUTPUT>NO FURTHER CHANGES</OUTPUT>
```

Note that this is round 2. At some point  
↪ after about 5 rounds,  
if there are no bugs in the code that you  
↪ need to fix, please stop:

```
IT IS ALWAYS A GOOD IDEA TO QUIT WHILE YOU  
↪ ARE AHEAD
```

Collect your numbered sentences together  
↪ into a single text and place that  
within the tags <OUTPUT></OUTPUT>.

You may explain your rationale for the set  
↪ of sentences you came up with, but  
please put NOTHING except the sentence  
↪ text within the <OUTPUT></OUTPUT>  
↪ tags.  
</INSTRUCTIONS>

### Comparative examples

```
<INSTRUCTIONS>
```

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
various degrees of COMPARISON of  
↪ adjectives. Examples:

COMPARATIVE

"Mary is taller than John."

SUPERLATIVE

"Chimpanzees are the closest to humans."

```
{{general_guidelines}}  
</INSTRUCTIONS>
```

## A.10.4 Story Generation Prompts

### Story generation prompt

```
<INSTRUCTIONS>  
Write me a very short story about  
↪ "{{story_topic}}". The theme and  
↪ development  
should be of your own choosing but please  
↪ observe the following constraints:
```

1. The text must be in English.
2. Use simple words (5th grade reading  
↪ level), and simple sentence structure.
3. No more than about 500 words.

Please put the story you write within  
↪ <OUTPUT></OUTPUT> tags.  
</INSTRUCTIONS>

### Core argument examples

```
<INSTRUCTIONS>
```

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
basic core grammatical relations, namely  
↪ SUBJECT, OBJECT and INDIRECT  
OBJECT. Use a variety of both TRANSITIVE  
↪ and INTRANSITIVE verbs.

An example that illustrates SUBJECT,  
↪ OBJECT and INDIRECT OBJECT in a  
↪ TRANSITIVE  
sentence might be:

"John gave the book to Mary."

An example with an INTRANSITIVE sentence  
↪ might be:

"John left yesterday."

```
{{general_guidelines}}  
</INSTRUCTIONS>
```

### Sentence design general guidelines

```
<INSTRUCTIONS>  
<!-- Common to all the instructions. -->  
General guidelines:
```

- All sentences must be in English.
- Use simple words (5th grade reading  
↪ level), and simple sentence structure.
- Each sentence should be no more than  
↪ about 15 words.
- Each sentence must be unique --- do not  
↪ repeat any sentences.
- Each sentence must be on a single line.
- Number all sentences consecutively thus:

1. sentence one ...
2. sentence two ...
- ...

### Inclusive/exclusive examples

```
<INSTRUCTIONS>
```

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate

the difference between the INCLUSIVE and  
↪ EXCLUSIVE interpretations of  
↪ "we"/"us".  
INCLUSIVE "we" is interpreted as including  
↪ the listener, whereas EXCLUSIVE "we",  
excludes the listener. English does not  
↪ mark this distinction grammatically,  
but one can infer from the context,  
↪ usually, which one is meant. Thus:

#### INCLUSIVE

"We need to leave now, so please get your  
↪ coat."  
"You are a valued member of our company."

#### EXCLUSIVE

"We didn't ask you to do that."  
"You were pretty mean to us."

Try to make your examples as clear as you  
↪ can in terms of which interpretation  
(INCLUSIVE or EXCLUSIVE) is the most  
↪ natural one.

{{general\_guidelines}}  
</INSTRUCTIONS>

### Negative examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
NEGATION of verbs. Examples:

"John didn't eat yet."

"You musn't leave."

{{general\_guidelines}}  
</INSTRUCTIONS>

### Nominal number examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
NUMBER in nouns, in particular SINGULAR,  
↪ DUAL and PLURAL. English of course  
↪ does  
not specifically mark DUAL, but in that  
↪ case you would use "two" with the  
↪ plural  
of the noun.

Examples follow:

SINGULAR:

"The man arrived yesterday."

DUAL:

"I saw two sheep."

PLURAL:

"The dogs ran after the ball."

{{general\_guidelines}}  
</INSTRUCTIONS>

### Oblique argument examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
various non-core grammatical relations,  
↪ including GENITIVE (i.e. POSSESSIVE),  
LOCATIVE, INSTRUMENTAL and ABLATIVE  
↪ (typically meaning motion away from  
↪ something).

Example sentences that illustrate each of  
↪ these might be:

GENITIVE:

"I borrowed John's book."

LOCATIVE:

"Mary lived in Rome."

INSTRUMENTAL:

"He cut the cucumber with scissors."

ABLATIVE:

"He left from the station this morning."

{{general\_guidelines}}  
</INSTRUCTIONS>

### Person examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
different PERSONS as SUBJECTS or OBJECTS  
↪ of sentences. Give examples with

FIRST PERSON SINGULAR ("I"/"me")  
FIRST PERSON PLURAL ("we"/"us")  
SECOND PERSON SINGULAR ("thou"/"thee")  
SECOND PERSON PLURAL ("you"/"you")  
THIRD PERSON SINGULAR ("he"/"him",  
↪ "she"/"her", "it"/"it")  
THIRD PERSON PLURAL ("they"/"them")

{{general\_guidelines}}  
</INSTRUCTIONS>

## Tense/aspect examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
TENSE and ASPECT in verbs. Of particular  
↪ interest are PRESENT, PAST, FUTURE,  
PERFECT, REMOTE PAST.

For example:

PRESENT:

I am leaving now.

PAST:

John went there yesterday.

FUTURE:

The club will meet tomorrow.

PERFECT:

Nancy has finished her work.

PAST PERFECT:

By Friday morning, all the mail had  
↪ arrived.

REMOTE PAST:

I did that a long time ago.

{{general\_guidelines}}  
</INSTRUCTIONS>

## Voice examples

<INSTRUCTIONS>

I'd like you to write me  
↪ {{number\_of\_sentences}} example  
↪ English sentences that illustrate  
VOICE in verbs, particularly ACTIVE and  
↪ PASSIVE.

For example:

ACTIVE:

The dog chased the deer.

PASSIVE:

The jewelry was stolen from the safe.

{{general\_guidelines}}  
</INSTRUCTIONS>

## A.10.5 Morphosyntax Prompts

### Cumulative morphosyntax prompt for grammatical case

<INSTRUCTIONS>

We are going to develop a 'translator' for  
↪ our ConLang, which will 'translate'  
from English sentences into an abstract  
↪ language that has English content  
↪ words,  
but may mark them with additional  
↪ morphosyntactic information, and where  
↪ the  
English phrases may be reordered.

Morphosyntactic markers may show up as  
↪ prefixes or suffixes. Our convention  
↪ for  
marking suffixes is to place a dash before  
↪ the affix, and for prefixes to place  
a dash after the affix:

"-XXX" is a suffix: when one attaches  
↪ "-XXX" to "word" one gets "word-XXX"

"XXX-" is a prefix: when one attaches  
↪ "XXX-" to "word" one gets "XXX-word"

In this portion we will consider the  
↪ nominal (noun) category of case  
↪ marking.

In the description below we will use  
↪ suffixes for the sake of example, but  
↪ a  
language may use either suffixes or  
↪ prefixes to mark this information.

First you need to decide whether the  
↪ language is Nominative-Accusative or  
Ergative-Absolutive.

Consider the English original sentences:

"John saw Mary"  
"John left"

In a Nominative-Accusative language the  
↪ subject of a sentence is marked  
Nominative ("NOM") and the direct object  
↪ is Accusative ("ACC"). If cases are  
marked as suffixes, then "John" would be  
↪ marked as "John-NOM" in both cases and  
Mary as "Mary-ACC":

John-NOM saw Mary-ACC  
John-NOM left

In an Ergative-Absolutive language, the  
↪ subject of a transitive sentence is  
marked Ergative ("ERG"), whereas objects  
↪ of transitive sentences and subjects  
↪ of  
intransitive sentences are marked  
↪ Absolutive ("ABS"). Again, assuming  
↪ these are  
marked as suffixes, then in our example  
↪ sentences above we would have:

John-ERG saw Mary-ABS  
John-ABS

Beyond these two, there is a more-or-less  
↪ universal hierarchy of cases with  
earlier ones in the following list being  
↪ more likely than later ones:

GEN: Genitive. E.g. "John's book"  
↪ -> "John-GEN book"  
DAT: Dative. E.g. "gave to John" ->  
↪ "gave John-DAT"  
LOC: Locative: E.g. "in Rome" ->  
↪ "Rome-LOC"  
ABL: Ablative: E.g. "by Rome"/"with  
↪ Rome" -> "Rome-ABL"  
INS: Instrumental: E.g. "with glue"  
↪ -> "glue-INS"

Languages may also mark the oblique  
↪ argument and the object of an  
↪ adposition with a specific  
↪ case-marking.  
For example, if a language marks such NPs  
↪ with a genitive suffix, we will have:

"for John" -> "for John-GEN"  
"last night" -> "last-GEN night-GEN".

Languages may also not mark case at all,  
↪ in which case we specify "NONE".

Our language has "{{case\_marking}}" case  
↪ marking, and it is marked by  
"{{case\_marking\_strategy}}".  
In our language, oblique noun phrases are  
↪ marked by "{{oblique\_case\_marking}}".

If the above case marking is not "NONE",  
↪ then be sure to pay attention to  
whether the case markers are specified as  
↪ prefixes or suffixes.

Your task is to 'translate' the text under  
↪ SAMPLE TEXT below. This text is  
itself a translation of an original text  
↪ that you did on the previous  
round. This original text is given under  
↪ ORIGINAL TEXT for your reference, but  
you should use the SAMPLE TEXT as the  
↪ starting point.

Any pronouns that you case mark should be  
↪ presented in their lemma form with the  
case marking: e.g., "our-GEN" should  
↪ become "we-GEN".

<!-- TODO(rws): Possibly omit the  
↪ following instruction. -->  
<!-- (ct): Omitted. Definiteness will be  
↪ handled by another feature.  
Do not mark the articles "the" or "a" but  
↪ rather omit these entirely. -->

Do not change any affixes that have been  
↪ added on a previous phase.

Place a period at the end of each sentence,  
↪ as in the original text. Don't

forget to translate the title, if there is  
↪ one. Downcase all content words,  
except proper names, which are to be left  
↪ capitalized. Thus "Man" would become  
"man", "We" would become "we", but "John"  
↪ would remain "John".

IMPORTANT: MAKE SURE YOU KEEP THE SAME  
↪ LINE STRUCTURE AS THE SAMPLE TEXT.  
↪ THAT  
IS IF THE SAMPLE TEXT HAS 23 LINES YOUR  
↪ RESULTING OUTPUT SHOULD HAVE 23 LINES,  
AND THE TEXT IN LINE k IN YOUR OUTPUT  
↪ SHOULD CORRESPOND TO THE TEXT IN LINE  
↪ k IN  
THE INPUT. DO NOT ADD QUOTES AROUND YOUR  
↪ OUTPUT.

```
# ORIGINAL TEXT BEGINS  
{{original_text}}  
# ORIGINAL TEXT ENDS
```

```
# SAMPLE TEXT BEGINS  
{{previous_translation}}  
# SAMPLE TEXT ENDS
```

Please put your response within  
↪ <OUTPUT></OUTPUT> tags.

</INSTRUCTIONS>

## A.10.6 Orthography Prompt

### Orthography prompt

<INSTRUCTIONS>  
We will now develop an orthography for our  
↪ ConLang. At present we will develop  
an orthography using the {{script}} script,  
↪ which may include extensions with  
diacritics, as in the  
↪ {{script}}-Extended-A and  
↪ {{script}}-Extended-B Unicode blocks.

The set of phonemes (in IPA) that you  
↪ developed for this language, is as  
↪ follows:

Consonants: {{consonants}}  
Vowels: {{vowels}}

A sample of words assuming the above IPA  
↪ symbols can be found below within the  
<LEXICON></LEXICON> block.

```
<LEXICON>  
{{lexicon_sample}}  
</LEXICON>
```

In developing your orthography, feel free  
↪ to take as examples the spelling  
systems of any languages you are familiar  
↪ with that use the {{script}} script in  
their writing system.

Your orthography does not need to be  
↪ completely "phonetic" in that the

pronunciation is completely  
↪ reconstructable from the spelling, but  
↪ it should be  
reasonable in that it does not deviate too  
↪ much from a "shallow orthography".

Once you have developed your spelling  
↪ system, you should write a Python  
↪ program  
that will convert from the phonetic  
↪ transcriptions of words into the  
↪ written  
form.

Notice that the input examples above have  
↪ the phonemes separated by spaces: in  
your code you should assume that the  
↪ phonetic input will have the phonemes  
separated in this way. However, the  
↪ letters in the spelled form should NOT  
↪ be  
space-separated. If there are symbols in  
↪ the script that naturally correspond  
to more than one phoneme, feel free to  
↪ include a rule for that, but again be  
aware that the phonemes will come in  
↪ space-separated.

Your function should be called  
↪ "ipa\_to\_orthography", and it should  
↪ take as input  
a word transcribed in IPA, and return a  
↪ spelling for that word.

Please also provide a main function so  
↪ that one can feed words in IPA to your  
program from the command line: have it  
↪ read from standard in.

Please make sure you include needed  
↪ functions and imports in your code.

Place your code in the block  
↪ <OUTPUT></OUTPUT>.  
PLEASE DO NOT PUT ANYTHING EXCEPT VALID  
↪ PYTHON CODE WITHIN THIS BLOCK.  
THIS MUST BE IMPORTABLE AS IS.

This task will depend on your deep  
↪ knowledge of linguistics, and Python  
programming. Good luck!  
</INSTRUCTIONS>

## A.10.7 Handbook Prompt

### Handbook prompt

<INSTRUCTIONS>  
Finally we will write a grammar and reader  
↪ for our ConLang.

#### 1. NAME

The first thing you need to do is come up  
↪ with a LANGUAGE\_NAME for the ConLang.

#### 2. PHONOLOGY

Then you describe the phonology.

The set of phonemes (in IPA) that you  
↪ developed for this language, is as  
↪ follows:

Consonants: {{consonants}}  
Vowels: {{vowels}}

#### 3. ORTHOGRAPHY

Next you will describe the orthography,  
↪ which is handled by the following  
↪ code:

```
# Begin orthography code
{{orthography_code}}
# End orthography code
```

#### 4. LEXICON

A subset of the lexicon with terms needed  
↪ for the sample story is as follows:

```
# Begin lexicon
{{lexicon}}
# End lexicon
```

#### 5. MORPHOSYNTAX

The morphosyntactic description on which  
↪ you based your morphosyntax was as  
follows. Below I remind you what the  
↪ different annotations mean, and which  
particular annotations were chosen for our  
↪ language:

##### NOMINAL MARKING

Number:  
SING: Singular (one)  
PLUR: Plural (more than one)  
DUAL: Dual (exactly two)  
Our language has: "{{number\_marking}}".

Case:  
NOM-ACC or ERG-ABS  
GEN: Genitive. E.g. "John's book"  
↪ -> "John-GEN book"  
DAT: Dative. E.g. "gave to John" ->  
↪ "gave John-DAT"  
LOC: Locative: E.g. "in Rome" ->  
↪ "Rome-LOC"  
ABL: Ablative: E.g. "by Rome"/"with  
↪ Rome" -> "Rome-ABL"  
INS: Instrumental: E.g. "with glue"  
↪ -> "glue-INS"  
Our language has "{{case\_marking}}" case  
↪ marking.

##### VERBAL MARKING

Tense-Aspect:  
PRES: Present tense. E.g. "goes" ->  
↪ "go-PRES"  
PAST: Past tense. E.g. "went" ->  
↪ "go-PAST"  
FUT: Future tense. E.g. "will go"  
↪ -> "go-FUT"

REMPAST: Remote past. E.g. "went (a long  
 ↳ time ago)" -> "go-REMPAST"  
 IMPF: Imperfective. E.g.  
 ↳ "went" -> "go-IMPF"  
 PERF: Perfective. E.g. "has  
 ↳ gone" -> "go-PERF"

Our language marks "{{tense\_aspect}}".

Head marking and/or person agreement:

VNOM: Nominative. E.g. "John leaves" ->  
 ↳ "John leave-VNOM"  
 VACC: Accusative. E.g. "see the dog" ->  
 ↳ "see-VACC dog"  
 VERG: Ergative. E.g. "John eats  
 ↳ something" -> "John eat-VERG"  
 VABS: Absolutive. E.g. "see the dog" ->  
 ↳ "see-VABS dog"; "John leaves" ->  
 ↳ "John leave-VABS"  
 VDAT: Dative. E.g. "give to the dog" ->  
 ↳ "give-VDAT dog"

Or a more detailed person-agreement system  
 ↳ such as:

1SGNOM: 1st person singular  
 ↳ nominative: E.g. "I leave" -> "I  
 ↳ leave-1SGNOM"  
 2SGNOM: 2nd person singular  
 ↳ nominative: E.g. "thou leavest" ->  
 ↳ "thou leave-2SGNOM"  
 3SGNOM: 3rd person singular  
 ↳ nominative: E.g. "John leaves" ->  
 ↳ "John leave-3SGNOM"  
 1SGACC: 1st person singular  
 ↳ accusative: E.g. "leave me" ->  
 ↳ "leave-1SGACC me"  
 1SGDAT: 1st person singular  
 ↳ dative: E.g. "give to me" ->  
 ↳ "give-1SGDAT me"

or

1SGERG: 1st person singular  
 ↳ ergative: E.g. "I see" -> "I  
 ↳ see-1SGERG"  
 1SGABS: 1st person singular  
 ↳ ergative: E.g. "I run" -> "I  
 ↳ run-1SGABS"; "see me" -> "see-1SGABS  
 ↳ me"

Our language marks "{{head\_marking}}".

In addition to the above, you may have  
 ↳ invented some further morphosyntactic  
 markers, which you can discover in the  
 ↳ lexicon in all upper-case entries.

#### INCLUSIVE-EXCLUSIVE DISTINCTION

The inclusive-exclusive distinction  
 ↳ applies in particular to the First  
 ↳ Person  
 Plural pronoun "we", and anything that  
 ↳ agrees with it (such as a verb).

In English "we" is ambiguous as to whether  
 ↳ or not it includes the listener. On

the other hand, many languages distinguish  
 ↳ between

1PLINCL	Inclusive First Person
↳ Plural	"we" including the
↳ listener	
1PLEXCL	Exclusive First Person
↳ Plural	"we" excluding the
↳ listener	

Thus if we translate:

"we like you"

into a language with the  
 ↳ INCLUSIVE\_EXCLUSIVE distinction, we  
 ↳ would translate  
 "we" here as "1PLEXCL", since it is clear  
 ↳ from the meaning that in this case  
 "we" does not include the listener. On the  
 ↳ other hand

"we all agree"

spoken to a group of people, would have  
 ↳ "we" translated as "1PLINCL".

Our language has:

↳ "{{inclusive\_exclusive}}".

#### GENDER

Many languages have grammatical gender, as  
 ↳ distinct from the natural gender  
 exhibited in English. Well-known examples  
 ↳ include Masculine-Feminine gender  
 systems of Romance languages; the  
 ↳ Masculine-Feminine-Neuter system of  
 ↳ German or  
 Russian; or the Common-Neuter system in  
 ↳ Dutch or in some Scandinavian  
 languages. Many languages of Africa have  
 ↳ non-sex-based gender systems. Since  
 sex-based systems are the majority (75% of  
 ↳ the sample of gendered languages in  
 the World Atlas of Language Structures),  
 ↳ we will assume a gender-based  
 system. Options are:

MASC:	Masculine:	E.g.
↳ John-MASC,	pumpkin-MASC	
FEM:	Feminine:	E.g.
↳ Sally-FEM,	sea-FEM	
NEUT:	Neuter:	E.g.
↳ snake-NEUT		
COMM:	Common gender:	E.g.
↳ John-COMM,	Sally-COMM	

The following gendered systems are  
 ↳ possible:

1. MASC, FEM
2. MASC, FEM, NEUT
3. COMMON, NEUT

For personal names, gender appropriate  
 ↳ affixes were chosen. For common nouns,  
 since English does not have gender, you  
 ↳ made a decision as to what gender to

assign to the noun, with the injunction:  
↪ be creative, but be consistent!

Our language marks "{gender\_marking}".

#### NUMERAL CLASSIFIERS

Many languages have classifiers that go  
↪ with numbers, and mark the type of noun  
associated with the number.

There are a variety of classifier systems,  
↪ with some languages like Japanese, or  
Chinese, having dozens of classifiers.  
↪ Classifiers are usually semantically  
based, depending on visual or taxonomic  
↪ features of the noun's meaning.

A typical classifier system might include:

CLFLAT:           Used with flat objects  
↪ like paper:                   three  
↪ CLFLAT paper   "3 pieces of paper"  
CLROUND: Used with round objects like  
↪ coins:                        twenty CLROUND  
↪ coin   "20 coins"  
CLLONG:           Used with long objects  
↪ like string, or rivers: one CLLONG  
↪ string "1 piece of string"  
CLANIMAL: Used with  
↪ animals:  
↪ two CLANIMAL cow   "2 cows"

Since the set of possible classifiers is  
↪ rather open ended, I am simply going  
↪ to  
tell you if our language (a)  
↪ HAS\_CLASSIFIERS or (b) NONE (= no  
↪ classifiers).  
You based your classifier system on what  
↪ you know about classifiers in  
↪ languages  
that you have read about. The classifier  
↪ should be introduced between the noun  
and the number in your translation.

For our language:  
↪ "{numeral\_classifier}".

#### 6. SYNTAX/WORD ORDER

Main:  
VSO: "drink John water"  
SVO: "John drink water"  
SOV: "John water drink"  
VOS: "drink water John"  
OVS: "water drink John"  
OSV: "water John drink"

Adjective-Noun:  
NA: "book red"  
AN: "red book"

Adposition-Noun:  
PN: "on table"  
NP: "table on"

Our language has

"{main\_word\_order}" main word order.  
"{adj\_noun\_word\_order}" adjective-noun  
↪ word order.  
"{adposition\_noun\_word\_order}"  
↪ adposition-noun word order.

#### 7. TEXTS

An example text is as follows, where after  
↪ "==" the first line is the  
morphosyntactic gloss, the second is the  
↪ phonetic transcription and the final  
↪ is  
the orthographic transcription:

```
# BEGIN EXAMPLE TEXT
{{sample_text}}
# END EXAMPLE TEXT
```

=====

Arrange your handbook according to the  
↪ format specified between the BEGIN and  
END lines below. Try to make your handbook  
↪ between 500 and 1000 words long:

```
<!-- TODO(rws): Have this output HTML? Or
↪ LaTeX? -->
# BEGIN HANDBOOK TEMPLATE
TITLE: A Grammar and Handbook of
↪ LANGUAGE_NAME
```

#### PHONOLOGY

# Details of the phonology, with some  
↪ commentary.

#### ORTHOGRAPHY

# Details of the orthography, with some  
↪ commentary.

#### MORPHOSYNTAX

# Details of the morphosyntax, with some  
↪ commentary.

#### SYNTAX

# Details of the syntax/word order, with  
↪ some commentary.

#### EXAMPLE TEXT

# GIVE SEVERAL EXAMPLES FROM THE PROVIDED  
↪ TEXT. Give the orthographic version of  
# the text, and please also include the  
↪ gloss and an English translation.

#### LEXICON

```
# Lexicon
# END HANDBOOK TEMPLATE
```

Place your resulting text in the block  
↪ <OUTPUT></OUTPUT>.

This task will depend on your deep  
↪ knowledge of linguistics, and Python

```
programming. Good luck!  
</INSTRUCTIONS>
```

```
# NEW_SAMPLE TEXT END
```

```
Please put your response within  
↪ <OUTPUT></OUTPUT> tags.
```

## A.10.8 Translation Prompt

### Translation prompt

You have created a language called  
↪ {{language\_name}}, and you have also  
↪ written  
a short handbook of the grammar of that  
↪ language.

The handbook you wrote reads as follows:

```
# BEGIN HANDBOOK  
{{handbook}}  
# END HANDBOOK
```

As part of creating this language, you  
↪ also provided a glossing "translation"  
from English into the morphosyntactic  
↪ structure of the target language for  
↪ the  
following text:

```
# SAMPLE TEXT BEGINS  
{{sample_text}}  
# SAMPLE TEXT END
```

Your translation was as follows:

```
# BEGIN TRANSLATION  
{{translation}}  
# END TRANSLATION
```

I would now like you to use this  
↪ information to "translate" a new text,  
↪ to be  
given below.

As before please be sure to give content  
↪ words in their lemma (dictionary)  
form. Please downcase all English words  
↪ that are not obviously names:  
i.e. "From" should become "from", but  
↪ "John" should be left as "John". Place  
↪ a period at  
the end of each sentence, as in the English  
↪ original. Don't forget to translate  
the title, if there is one. Downcase all  
↪ content words, except proper names,  
which are to be left capitalized. Thus  
↪ "Man" would become "man", but "John"  
would remain "John".

```
IMPORTANT: MAKE SURE YOU KEEP THE SAME  
↪ LINE STRUCTURE AS THE ORIGINAL FILE.  
↪ THAT  
IS IF THE INPUT FILE HAS 23 LINES YOUR  
↪ RESULTING OUTPUT SHOULD HAVE 23 LINES,  
AND THE TEXT IN LINE k IN YOUR OUTPUT  
↪ SHOULD CORRESPOND TO THE TEXT IN LINE  
↪ k IN  
THE INPUT. DO NOT ADD QUOTES AROUND YOUR  
↪ OUTPUT.
```

```
# NEW_SAMPLE TEXT BEGINS  
{{new_sample_text}}
```

## A.11 Phonological changes

Besides phonotactics, all languages also exhibit phonological processes that may alter the shapes of words and morphemes from what one would expect from the underlying phonotactic model. Historically such changes start out as articulatory or acoustically motivated phonetic changes, which over time may lose their immediate phonetic motivation and become part of the morphophonological pattern of the language (Blevins, 2004). For example, the *umlaut* vowel change observed in the formation of some plural nouns in German like *Maus* 'mouse' (plural *Mäuse*), *Haus* 'house' (plural *Häuser*), or *Mann* 'man' (plural *Männer*), originated from a Proto-West-Germanic plural formation with the vowel /i/, which caused a fronting of the vowel of the preceding noun stem. The fronting itself was a phonetically motivated case of assimilation. The /i/ was subsequently lost leaving only the vowel change in the stem. This vowel change, now dissociated from the original phonetic motivation, came itself to be associated with plural formation, and in fact spread in its use to nouns that did not originally have the /i/ marking in the plural.

There have been hundreds of phonological changes documented for the world's languages. Many of these can be grouped into larger categories, including:

**Assimilation:** A sound becomes more similar to one in its neighborhood. E.g. the Germanic umlaut example given above.

**Dissimilation:** A sound becomes less similar to one in its neighborhood. E.g. French *pelegrin* 'pilgrim', from Latin *peregrinus*, with a change of the first /r/ to /l/, dissimilating from the second /r/.

**Deletion:** The removal of a sound. E.g. the loss of final consonants in French, such as *chat* /ʃa/ 'cat' where the original final /t/, still spelled, is lost in the phonological form.

**Epenthesis:** The insertion of a sound. E.g. in Spanish, initial consonant clusters with /s/ are prepended by an /e/, thus *estación* 'station'.

**Lenition:** Or weakening. E.g. Intervocalic /t/ and /d/ in most dialects of American English are weakened to a *flap* sound so that *butter* is /bʌtəʃ/.

**Fortition:** Or strengthening. E.g. Breton *hard mutation* where initial voiced stops are changed to voiceless stops, as in *breur* ‘brother’ *ho preur* ‘your (pl.) brother’.

**Sound shifts:** Such as the Great English Vowel Shift that shifted most long vowels, with high vowels becoming diphthongs—/ī/→/ai/, /ū/→/aʊ/, mid vowels becoming high vowel (also pronounced as diphthongs in most varieties of English)—/ō/→/ū/, /ē/→/ī/, and low vowels becoming mid vowels—/ā/→/ē/.

Within each of these broad categories are many variations depending in part on the phonotactic pattern of the language to which the changes were applied.

A main problem for any ConLang creator is thus to decide on a reasonable set of phonological rules to apply. There has in fact been a lot of interest in this problem in the ConLang community with various advice on how to ‘age’ one’s ConLang, i.e. how to make it change over time in a way that resembles phonological changes that have been attested in the world’s languages.

One resource for this is Diachronica<sup>12</sup>, which compiles phoneme inventories for a large number of historically reconstructed languages, and lists phonological changes that historical linguists have proposed to derive daughter language forms.

In this work we propose a novel approach that builds on Diachronica and guides an LLM to construct a set of phonological rules that can plausibly apply to a language with the specified phonotactics. Basically the process involves finding a phoneme inventory of a language X in the Diachronica data that is most similar to the phonotactics of our language, and then selecting some rules based on the rules listed for X as the basis of our phonological rule set.

The steps can be summarized as follows:

- First, Diachronica only provides phoneme inventories for the initial reconstructed proto-language. For example, the document lists

a reconstructed inventory for Proto-Indo-European, and rules to go from Proto-Indo-European to Common Germanic, from Common Germanic to West Germanic, from West-Germanic to Anglo-Frisian, from Anglo-Frisian to Old English, etc. But phoneme inventories are not directly provided for Common Germanic, West Germanic and so forth. In other words for a chain of rule sets  $A \rightarrow B \rightarrow C \rightarrow D \dots$ , we have the reconstructed phoneme inventory for *A*, but not for *B*, *C*, *D*. In order to consider the reconstructed rules for, say,  $C \rightarrow D$  as a model for our ConLang, we therefore need to have a phoneme inventory for *C*. This can be constructed, approximately, by applying  $A \rightarrow B$  to the inventory in *A*, deducing the likely phoneme set for *B*, and then repeating the process. So the first, offline task for the LLM was to fill out the dataset provided by Diachronica by considering the phonological rules in each set, and deducing the phoneme sets of the corresponding daughter language.

- Armed with the enhanced version of the Diachronica data, we then find the closest *N* phoneme inventories to the one already developed for our ConLang.
- The LLM is then presented with the phoneme inventory already created for the ConLang, a list of sample word forms generated by the phonotactic grammar, and for each of the *N* phoneme inventories from Diachronica, the name of the reconstructed language, the reconstructed phoneme inventory, and the reconstructed phonological rules. The LLM is then instructed to look at the rules, and write a Python program that implements a reasonable subset of the rules that can be applied to the output of the phonotactic grammar. As with the phonotactic construction, the system is agentic in that it iterates the process, presenting the previous program and its outputs, and any issues noted, instructing the LLM to try to improve the implementation.

We describe each of these steps in more detail in the sections below.

### A.11.1 Enhancing Diachronica

The first stage involves expanding Diachronica to provide phoneme inventories for intermediate stage

<sup>12</sup><https://chridd.nfshost.com/diachronica/index-diachronica.pdf>

reconstructed languages. Starting with the proto language for each major branch, rules to derive the daughter language are consulted, and the phoneme inventory for the daughter language is estimated. The procedure is then iterated, with the new daughter language and subsequent rules used to derive the next level daughter languages. At each stage the LLM is prompted to provide its initial guess as to the phoneme inventory for the daughter language using the prompt template Diachronica Expansion given in Appendix A.12. In a second stage, the LLM is asked to possibly refine its estimate: see the prompt template in Diachronica Expansion Stage 2.

For example the following phoneme set was reconstructed for Old English. Note that this is constructed from Proto-Indo-European to Common Germanic to West Germanic to Anglo-Frisian to Old English:

a, a:, b, d, dʒ, e, e:, f, g, i, i:, j, k, k<sup>w</sup>, l, m, n, o, o:, p, r, s, t, tʃ, u, u:, w, x, æ, æ:, ɸ, θ, ɣ, ȝ, ū:, ȝ, θ, ȝ:

In the second version, Claude 3.5 Sonnet made the following decisions, in some cases revising the inventory:

1. Removed nasal vowels (ẽ:, õ:, ĩ:, ũ:): The rule “V[+nas] → V[-nas]” indicates that all nasal vowels became non-nasal.
2. Retained ‘h’: This phoneme was present in Anglo-Frisian and there’s no rule indicating its loss, so it should be kept in Old English.
3. Removed ‘ã’: The rule “ũ: → õ:” suggests this phoneme changed, and then the nasalization was lost as per the second rule.
4. Retained ‘k’: While there’s a rule about ‘k’ becoming ‘tʃ’ in certain circumstances, it’s not a blanket change. The original ‘k’ should still exist in other contexts.
5. Added ‘j’: This comes from the rule “k ȝ g → tʃ j dʒ” although it’s noted to occur only in certain circumstances.
6. Retained ‘u’ and ‘i’: The rule about their loss is context-dependent (word-final position), so they should still exist in other positions.

The second revised version was as follows:

a, a:, b, d, dʒ, e, e:, f, g, h, i, i:, j, k, k<sup>w</sup>, l, m, n, o, o:, p, r, s, t, tʃ, u, u:, w, x, æ, æ:, j, θ.

According to Wikipedia<sup>13</sup>, the actual Old English phoneme inventory was as follows:

a, a:, b, d, e, e:, f, h, i, i:, j, k, l, m, n, o, o:, p, r, s, ʃ, t, tʃ, u, u:, w, x, æ, æ:, ʃ, θ, ȝ, y, y:.

Notable differences from the Diachronica-derived reconstruction include:

- Cases like /dʒ/, which was an allophone of /tʃ/, but which the reconstruction includes as a separate phoneme. Similarly /g/ was an allophone of /k/.
- Decisions like whether to consider the /k<sup>w</sup>/ of a word like *cwēn* ‘queen’ as a separate phoneme, or as a sequence of /kw/.
- Missing /ȝ/ and /ʃ/.
- Use of /a/ rather than /ɑ/.
- The missing front rounded vowels /y/ and /y:/.

In addition, Old English had several diphthongs which were not listed as separate phonemes in the Diachronica-derived reconstruction.

Despite these differences, the reconstruction is not too bad, considering that it is derived from a hypothetical reconstruction of Proto-Indo-European, run through four sets of imprecisely stated rules.

### A.11.2 Selecting the closest phoneme set

Given the expanded Diachronica set the next task is to find the language(s) with the closest phoneme set to that of the constructed ConLang. For each phoneme set from Diachronica, and for each phoneme in that set, the distance of the closest phoneme in the ConLang phoneme set is found. Distance between two phonemes is computed by counting the matching phonetic features of the two phonemes, using the phonetic feature set from PHOIBLE 2.0 (Moran and McCloy, 2019). The distance between the two phoneme sets is simply the sum of the phoneme distances, with an additional penalty for the difference in the cardinalities of the two sets. Diachronica phoneme sets are then rank ordered by increasing distance from the ConLang set.

<sup>13</sup>[https://en.wikipedia.org/wiki/Old\\_English\\_phonology](https://en.wikipedia.org/wiki/Old_English_phonology)

### A.11.3 Constructing the phonology

Once the closest Diachronica-derived phoneme set is chosen, the next phase is to come up with a set of phonological rules. These rules are based on the sets of rules that are listed in Diachronica that mapped from the language associated with the phoneme set, to various daughter languages. Here we allow the LLM a fair amount of freedom, basically telling it to “Devise a set of rules that makes sense for the phoneme set of our language given the observed phonotactics” and to implement this as an ordered set of rules in Python. The prompt for this initial set is given under “Phonological Rules” in Appendix A.12. One important instruction is that as part of the rule definition, the LLM must provide a set of example input-output pairs that show what the model is supposed to do. These will then be used in subsequent phases to check that the rules are doing what they are supposed to, and to give feedback to the model to improve the rules.

An initial set of rules proposed for a Welsh-based phonology was as in the code fragment below, where the closest phoneme set turned out to be Early Modern English, and the basis rule sets chosen were Early Modern English to American English, Early Modern English to Australian English and Early Modern English to British English—see Sections 17.7.2.1.5 to 17.7.2.1.7 in Diachronica.

```
1 import re
2
3 def vowel_shift(inp: str) -> str:
4     # Example inputs/outputs:
5     # u -> ʊ
6     # o -> u
7     # ɔ -> o
8     vowel_map = {
9         'u': 'ʊ',
10        'o': 'u',
11        'ɔ': 'o',
12        'ɛ': 'e',
13        'æ': 'ɛ'
14    }
15    return ' '.join(vowel_map.get(phone,
16                    phone) for phone in inp.split())
17
18 def consonant_lenition(inp: str) -> str:
19     # Example inputs/outputs:
20     # V t V -> V ð V
21     # V k V -> V x V
22     # V p V -> V f V
23     inp = re.sub(r'([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ]) t ([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ])',
24                 r'\1 ð \2', inp)
25     inp = re.sub(r'([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ]) k ([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ])',
26                 r'\1 x \2', inp)
27     inp = re.sub(r'([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ]) p ([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ])',
28                 r'\1 f \2', inp)
29
30     return inp
31
32 def nasal_assimilation(inp: str) -> str:
33     # Example inputs/outputs:
34     # n p -> m p
35     # n k -> ŋ k
36     # n . b -> m . b
37     inp = re.sub(r'n (\.)?([pbm])', r'm \1\2', inp)
38     inp = re.sub(r'n (\.)?([kg])', r'ŋ \1\2', inp)
39     return inp
40
41 def palatalization(inp: str) -> str:
42     # Example inputs/outputs:
43     # t i -> tʃ i
44     # d i -> dʒ i
45     # s i -> ʃ i
46     inp = re.sub(r't ([iei])', r'tʃ \1', inp)
47     inp = re.sub(r'd ([iei])', r'dʒ \1', inp)
48     inp = re.sub(r's ([iei])', r'ʃ \1', inp)
49     return inp
50
51 def vowel_nasalization(inp: str) -> str:
52     # Example inputs/outputs:
53     # a n -> ɤ̃ n
54     # o m -> ɔ̃ m
55     vowels = 'æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ'
56     nasals = 'mŋŋ'
57     for v in vowels:
58         for n in nasals:
59             inp = re.sub(f'{{v}} ({{n}})', f'{{ṽ}} \1', inp)
60     return inp
61
62 def cluster_simplification(inp: str) -> str:
63     # Example inputs/outputs:
64     # k t -> t
65     # p t -> t
66     # ŋ k -> ŋ
67     inp = re.sub(r'[kp] t', 't', inp)
68     inp = re.sub(r'ŋ k', 'ŋ', inp)
69     return inp
70
71 def vowel_reduction(inp: str) -> str:
72     # Example inputs/outputs:
73     # a . k u -> ə . k u
74     # k u . b r o -> k u . b r ə
75     words = inp.split('#')
76     for i, word in enumerate(words):
77         syllables = word.strip().split('.')
78         if len(syllables) > 1:
79             for j, syl in enumerate(syllables[:-2]):
80                 syllables[j] = re.sub(r'([æɛiɪɔʊʊəɪəɪaɪəʊɪeɪəʊ])', 'ə', syl)
81             words[i] = '.'.join(syllables)
82     return '#'.join(words)
83
84 def rules(inp: str) -> str:
85     output = vowel_shift(inp)
86     output = consonant_lenition(output)
87     output = nasal_assimilation(output)
```

```

83 output = palatalization(output)
84 output = vowel_nasalization(output)
85 output = cluster_simplification(
output)
86 output = vowel_reduction(output)
87 return output

```

The rules are then run on words generated by the phonotactic model, as well as the example input-output sequences listed in the phonological rule code itself, and any errors noted. These errors, along with the original instructions and the code produced on the previous phase, are fed to the model in a second round of prompting. For example, the code above misses spaces between phonemes in some cases, so these and other issues are pointed out to the model. On the second round the model is able to correct the spacing issue along with other problems, listing its corrections as follows:

1. Fixed the spacing issue by ensuring that all phonemes are properly space-separated, including word and syllable boundaries.
2. Corrected the 'consonant\_lenition' function to properly apply the rules for  $t \rightarrow \delta$ ,  $k \rightarrow x$ , and  $p \rightarrow f$  between vowels.
3. Fixed the 'vowel\_nasalization' function to maintain the space between the nasalized vowel and the following nasal consonant.
4. Improved the 'vowel\_reduction' function to correctly reduce vowels in non-final syllables and the final syllable of words with more than one syllable.
5. Updated the vowel set in various functions to include the new 'ʉ' phoneme introduced by the vowel shift.

The second round code is as follows:

```

1 import re
2
3 def vowel_shift(inp: str) -> str:
4     # Example inputs/outputs:
5     # u -> ʉ
6     # o -> u
7     # ɔ -> o
8     vowel_map = {
9         'u': 'ʉ',
10        'o': 'u',
11        'ɔ': 'o',
12        'e': 'e',
13        'æ': 'ɛ'
14    }
15    return ' '.join(vowel_map.get(phone,
16                    phone) for phone in inp.split())

```

```

17 def consonant_lenition(inp: str) -> str:
18     # Example inputs/outputs:
19     # V t V -> V δ V
20     # V k V -> V x V
21     # V p V -> V f V
22     vowels = 'æɛeiioʊuʉəɪæiaiaʊɔieioʊʉ'
23     inp = re.sub(f'([{{vowels}}]) t ([{{
vowels}}])', r'\1 δ \2', inp)
24     inp = re.sub(f'([{{vowels}}]) k ([{{
vowels}}])', r'\1 x \2', inp)
25     inp = re.sub(f'([{{vowels}}]) p ([{{
vowels}}])', r'\1 f \2', inp)
26     return inp
27
28 def nasal_assimilation(inp: str) -> str:
29     # Example inputs/outputs:
30     # n p -> m p
31     # n k -> ŋ k
32     # n . b -> m . b
33     inp = re.sub(r'n (\.)?([pbm])', r'm
\1\2', inp)
34     inp = re.sub(r'n (\.)?([kg])', r'ŋ
\1\2', inp)
35     return inp
36
37 def palatalization(inp: str) -> str:
38     # Example inputs/outputs:
39     # t i -> tʃ i
40     # d i -> dʒ i
41     # s i -> ʃ i
42     inp = re.sub(r't ([iei])', r'tʃ \1',
inp)
43     inp = re.sub(r'd ([iei])', r'dʒ \1',
inp)
44     inp = re.sub(r's ([iei])', r'ʃ \1',
inp)
45     return inp
46
47 def vowel_nasalization(inp: str) -> str:
48     # Example inputs/outputs:
49     # a n -> ã n
50     # o m -> õ m
51     vowels = 'æɛeiioʊuʉəɪæiaiaʊɔieioʊʉ'
52     nasals = 'mŋŋ'
53     for v in vowels:
54         for n in nasals:
55             inp = re.sub(f'({v}) ({n})',
f'\1\i \2', inp)
56     return inp
57
58 def cluster_simplification(inp: str) ->
str:
59     # Example inputs/outputs:
60     # k t -> t
61     # p t -> t
62     # ŋ k -> ŋ
63     inp = re.sub(r'[kp] t', 't', inp)
64     inp = re.sub(r'ŋ k', 'ŋ', inp)
65     return inp
66
67 def vowel_reduction(inp: str) -> str:
68     # Example inputs/outputs:
69     # a . k u -> ə . k u
70     # k u . b r o -> k u . b r ə
71     words = inp.split('#')
72     for i, word in enumerate(words):
73         syllables = word.strip().split('.')
74         if len(syllables) > 1:
75             for j in range(len(syllables)

```

```

) - 2): # All but the last two
syllables
76         syllables[j] = re.sub(r'
([æeɪiɔʊoʊiæiaʊɔɪeɪəʊ]), 'ə',
syllables[j])
77         syllables[-1] = re.sub(r'([æ
eɪiɔʊoʊiæiaʊɔɪeɪəʊ]), 'ə',
syllables[-1]) # Reduce final
syllable
78         words[i] = ' '.join(syllables)
79         return ' '.join(words)
80
81 def rules(inp: str) -> str:
82     output = vowel_shift(inp)
83     output = consonant_lenition(output)
84     output = nasal_assimilation(output)
85     output = palatalization(output)
86     output = vowel_nasalization(output)
87     output = cluster_simplification(
output)
88     output = vowel_reduction(output)
89     return output

```

## A.12 Phonology Prompts

The next two prompt templates relate to the expansion of the Diachronica data as described in Section A.11.1. Note that the abbreviations variable is filled in with a set of abbreviations listed in <https://chridd.nfshost.com/diachronica/index-diachronica.pdf>, page 6.

### Diachronica Expansion

In what follows you will find a set of  
 ↳ phonemes for a reconstructed language,  
 ↳ in  
 this case {{ parent }}. You will then be  
 ↳ given a set of phonological rules that  
 have been reconstructed to derive the  
 ↳ daughter language {{ daughter }}.

To the best of your ability, given the  
 ↳ phoneme set and the rules, derive a  
 probable phoneme set for the daughter  
 ↳ language. If you happen to know the  
 accepted reconstruction of the phoneme set  
 ↳ for {{ daughter }}, you may let that  
 guide your reconstruction.

The reconstruction will necessarily be  
 ↳ approximate since the rules are not  
 necessarily complete and are often vaguely  
 ↳ specified, but here are some  
 guidelines:

If a rule is listed without a context, e.g.

e → i

then you may assume that all instances of  
 ↳ /e/ are replaced with /i/. If on the  
 other hand it is listed with a context,  
 ↳ e.g.

e → i / \_ #

then you should assume that only some  
 ↳ instances of /e/ are changed and /e/  
 ↳ is  
 still retained in the system. Note that in  
 ↳ some cases conditions on rules are  
 described in parenthetical remarks after  
 ↳ the main body of the rule.

You will also occasionally see multistage  
 ↳ rules like

dʒ → tʃ → ʃ

This can be interpreted as equivalent to

dʒ → ʃ

Sometimes a rule will specify changes in  
 ↳ parallel, e.g.:

b<sup>fi</sup> d<sup>fi</sup> g<sup>fi</sup> → β ð γ

which is to be interpreted the same as

b<sup>fi</sup> → β

d<sup>fi</sup> → ð

g<sup>fi</sup> → γ

Unfortunately these rule sets are rather  
 ↳ sloppily done so you will have to use  
 your beset judgment.

The following abbreviations and other  
 ↳ conventions may be useful in  
 ↳ understanding  
 some of the rules:

```

## BEGIN ABBREVIATIONS
{{ abbreviations }}
## END ABBREVIATIONS

```

The reconstructed phonemes for {{ parent  
 ↳ }} are as follows:

```

{{ phonemes }}

```

The reconstructed rules deriving {{  
 ↳ daughter }} from {{ parent }} are as  
 follows:

```

## BEGIN RULES
{{ rules }}
## END RULES

```

Place your predicted phoneme set as a  
 ↳ comma-separated list in a  
 <PHONEMES></PHONEMES> tag.

### Diachronica Expansion Stage 2

In what follows you will find a set of  
 ↳ phonemes for a reconstructed language,  
 ↳ in  
 this case {{ parent }}. You will then be  
 ↳ given a set of phonological rules that  
 have been reconstructed to derive the  
 ↳ daughter language {{ daughter }}.

To the best of your ability, given the  
↳ phoneme set and the rules, derive a  
probable phoneme set for the daughter  
↳ language. If you happen to know the  
accepted reconstruction of the phoneme set  
↳ for {{ daughter }}, you may let that  
guide your reconstruction.

The reconstruction will necessarily be  
↳ approximate since the rules are not  
necessarily complete and are often vaguely  
↳ specified, but here are some  
guidelines:

If a rule is listed without a context, e.g.

e → i

then you may assume that all instances of  
↳ /e/ are replaced with /i/. If on the  
other hand it is listed with a context,  
↳ e.g.

e → i / \_ #

then you should assume that only some  
↳ instances of /e/ are changed and /e/  
↳ is  
still retained in the system. Note that in  
↳ some cases conditions on rules are  
described in parenthetical remarks after  
↳ the main body of the rule.

You will also occasionally see multistage  
↳ rules like

dʒ → tʃ → ʃ

This can be interpreted as equivalent to

dʒ → ʃ

Sometimes a rule will specify changes in  
↳ parallel, e.g.:

b<sup>h</sup> d<sup>h</sup> g<sup>h</sup> → β ð γ

which is to be interpreted the same as

b<sup>h</sup> → β

d<sup>h</sup> → ð

g<sup>h</sup> → γ

Unfortunately these rule sets are rather  
↳ sloppily done so you will have to use  
your beset judgment.

The following abbreviations and other  
↳ conventions may be useful in  
↳ understanding  
some of the rules:

```
## BEGIN ABBREVIATIONS
{{ abbreviations }}
## END ABBREVIATIONS
```

The reconstructed phonemes for {{ parent  
↳ }} are as follows:

```
{{ phonemes }}
```

The reconstructed rules deriving {{  
↳ daughter }} from {{ parent }} are as  
follows:

```
## BEGIN RULES
{{ rules }}
## END RULES
```

On a previous round you constructed the  
↳ following phoneme set for the  
daughter language {{ daughter }}:

```
{{ daughter_phonemes }}
```

Have another look at the parent phoneme  
↳ set for {{ parent }} above, the given  
rules, and check if anything is missing  
↳ from your proposed set for  
{{ daughter }}, or if there are phonemes  
↳ that should not be there. In  
particular, be on the look-out for cases  
↳ where a context-dependent rule might  
have been overapplied to eliminate a  
↳ parent-language phoneme that should  
↳ NOT  
have changed in all cases. Explain your  
↳ reasons for any changes.

As before, place your new predicted  
↳ phoneme set as a comma-separated list  
↳ in a  
<PHONEMES></PHONEMES> tag.

## Phonological Rules

Languages change and one of the ways in  
↳ which they change is for their sound  
systems to change.

Recall that in our language the set of  
↳ phonemes (in IPA), not including the  
stress, syllable-, morpheme-, and  
↳ word-boundary markers, is as follows:

```
Consonants: {{ consonants }}
Vowels: {{ vowels }}
```

Here are some sample words generated from  
↳ your phonotactic grammar:

```
{{ sample_words }}
```

We compared this set against a range of  
↳ reconstructed languages, and found  
{% if num == 1 %} one language which seems  
↳ closest in terms of phonological  
↳ makeup.

We give the name of the reconstructed  
↳ language, its reconstructed  
phoneset and one or more sets of rules  
↳ that have been reconstructed to derive  
daughter languages from this language.  
{% else %} {{ num }} languages which  
↳ seemss closest in terms of  
↳ phonological makeup. For each

of these, we give the name of the  
↳ reconstructed language, its  
↳ reconstructed  
phoneset and one or more sets of rules  
↳ that have been reconstructed to derive  
daughter languages from those languages.  
{% endif %}

```
{{ reconstructions }}
```

Based on the above:

1. Devise a set of rules that makes sense  
↳ for the phoneme set of our language  
given the observed phonotactics.
2. Implement each of these rules in  
↳ Python.
3. Put the rules together into a Python  
↳ library.

Each rule should take the form of a python  
↳ function that takes as input a string  
of space-delimited phonemes and possible  
↳ syllable boundaries, and outputs a  
string in the same format. You must make  
↳ sure that the output phonemes are  
space-delimited as in the input. All your  
↳ rules must allow for the presence of  
spaces in the input!!!!

We assume "#" as an end-of-word symbol:  
↳ you should assume that word boundaries  
in the input will be indicated by this  
↳ symbol. The symbol ".", if it occurs,  
represents a syllable boundary. The symbol  
↳ "-" if it occurs represents a  
morpheme boundary. For rules that depend  
↳ on stress, you are to  
assume that the language has {{  
↳ stress\_placement }} stress.

Start each rule with a comment that gives  
↳ some examples of what the input and  
output of the rule should be. See below  
↳ for format.

The template for a rule will be:

```
## BEGIN TEMPLATE
def ruleName1(inp: str) -> str:
    # Example inputs/outputs:
    # input1 -> output1
    # input2 -> output2
    ...
    return output
## END TEMPLATE
```

Note that any phonological rules that  
↳ should apply across syllables MUST  
↳ take  
the possible presence of "." into account.  
↳ Thus a nasal assimilation rule  
within a word should normally apply across  
↳ syllables, so that:

a n . b o n

would become

a m . b o n

This means that you will need to include  
↳ the possibility of an intervening  
↳ space  
plus "." in your regular expressions.

Similarly, morpheme boundaries ("-")  
↳ should NOT generally affect the output  
↳ of  
rules and so rules must be written to  
↳ apply across them.

Implement a wrapper function that calls  
↳ your set of rules in the intended order  
of their application:

```
## BEGIN TEMPLATE
def rules(inp: str) -> str:
    output = ruleName1(inp)
    output = ruleName2(output)
    # ...
    return output
## END TEMPLATE
```

Remember:

1. All phonemes are space-separated. All  
↳ your rules must take this into  
account. Be careful: make sure you  
↳ allow for EXACTLY ONE space between  
phonemes since rules that depend on more  
↳ than one space being there will not  
work. Your rules must also output  
↳ space-separated phonemes. Thus if  
↳ the  
input to a nasal assimilation rule is "n  
↳ p", the output should be "m p", not  
"mp".
2. If you use the Python regex library and  
↳ make use of groups, make sure you  
have enough capturing groups to support  
↳ the number of back references you  
assume. A common error is to have  
↳ backreferences like "\\1\\2", but  
↳ only have a  
single previous capturing group.
3. Avoid using regex look-behind since you  
↳ inevitably miss the point that  
look-behind patterns are fixed width,  
↳ which triggers the  
"sre\_constants.error: look-behind  
↳ requires fixed-width pattern"  
↳ error.
4. Finally, it is OK if your rules  
↳ introduce phonemes that are NOT in the  
input  
phoneme set since, after all, that is  
↳ what sound change is all about.

Make sure you have imports for all needed  
↳ libraries in your code.

Explain your reasoning. Then place your  
↔ resulting code in the block  
<OUTPUT></OUTPUT>.

This task will depend on your deep  
↔ knowledge of historical linguistics,  
↔ and  
Python programming. Be creative and good  
↔ luck!

## **B Morphosyntactic feature specifications**

Morphosyntactic feature specifications based on nine languages are presented in Tables 9–17.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		VSO
	oblique_word_order		VOX
	adj_noun_word_order		NA
	posspron_pron_word_order		NPoss
	num_noun_word_order		NumN
	adposition_noun_word_order		PN
Morphology	case	case_marking	nominative, accusative, genitive
		case_marking_strategy	suffix
	definiteness	oblique_case_marking	genitive
		definiteness	definite
		definiteness_marking_strategy	prefix
	adjective_agreement	definiteness_agreement	None
		adjective_agreement	number, case, definiteness
	comparative	adjective_agreement_strategy	suffix
		comparative	comparative, superlative
	tense_aspect	comparative_marking_strategy	suffix
		tense_aspect	present, past, future
	mood	tense_aspect_marking_strategy	suffix
		mood	indicative, subjunctive, imperative
	voice	mood_marking_strategy	suffix
		voice	active, passive
	person	voice_marking_strategy	suffix
		person_agreement	first, second, third
		person_marking_strategy	suffix
		verbal_number_agreement	singular, plural, dual
		verbal_number_marking_strategy	suffix
inclusive_exclusive		True	
nominal_number	nominal_number	singular, plural, dual	
	nominal_number_marking_strategy	suffix	
relativization	relativization_order	head-initial	
	relativization_marking	head-marking	
	relativizer_position	postpositional	
	relativizer_morpheme	word	
negation		prepositional word	
infinitive	infinitive	None	

Table 9: The configuration of the Arabic-like feature set. Though the Arabic inflectional paradigms are highly fusional rather than agglutinative, such features are expressed as suffixes or prefixes in this feature set for simplicity.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		VOS
	oblique_word_order		VOX
	adj_noun_word_order		NA
	posspron_pron_word_order		NPoss
	num_noun_word_order		NumN
	adposition_noun_word_order		PN
Morphology	case		None
	definiteness		None
	adjective_agreement		None
	comparative	comparative	comparative, superlative
		comparative_marking_strategy	postpositional word
	tense_aspect		None
	mood		None
	voice		None
	person	person_agreement	first, second, third
		person_marking_strategy	prepositional word
		verbal_number_agreement	singular, plural, dual, paucal
		verbal_number_marking_strategy	prepositional word
		inclusive_exclusive	
	nominal_number		None
	relativization	relativization_order	head-initial
		relativization_marking	None
		relativizer_position	None
relativizer_morpheme		None	
negation		prepositional word	
infinitive	infinitive	None	

Table 10: The configuration of the Fijian-like feature set.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order oblique_word_order adj_noun_word_order posspron_pron_word_order num_noun_word_order adposition_noun_word_order		SVO VOX NA PossN NumN PN
Morphology	case definiteness  adjective_agreement  comparative  tense_aspect  mood   voice  person   inclusive_exclusive nominal_number  relativization   negation infinitive	 definiteness definiteness_marking_strategy definiteness_agreement  adjective_agreement adjective_agreement_strategy  comparative comparative_marking_strategy  tense_aspect tense_aspect_marking_strategy  mood   mood_marking_strategy voice voice_marking_strategy person_agreement person_marking_strategy verbal_number_agreement verbal_number_marking_strategy  nominal_number nominal_number_marking_strategy relativization_order relativization_marking relativizer_position relativizer_morpheme  infinitive infinitive_position	None definite, indefinite prepositional word None number suffix comparative, superlative, equative prepositional word present, past, future, imperfect suffix indicative, subjunctive, imperative, conditional suffix active, passive suffix first, second, third suffix singular, plural suffix False singular, plural suffix head-initial head-marking postpositional word postpositional word infinitive suffix

Table 11: The configuration of the French-like feature set.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order oblique_word_order adj_noun_word_order posspron_pron_word_order num_noun_word_order adposition_noun_word_order		OVS OVX NA PossN NumN PN
Morphology	case definiteness  adjective_agreement  comparative  tense_aspect  mood   voice  person   inclusive_exclusive nominal_number  relativization   negation infinitive	 definiteness definiteness_marking_strategy definiteness_agreement  adjective_agreement adjective_agreement_strategy  comparative comparative_marking_strategy  tense_aspect tense_aspect_marking_strategy  mood   mood_marking_strategy voice voice_marking_strategy person_agreement person_marking_strategy verbal_number_agreement verbal_number_marking_strategy  nominal_number nominal_number_marking_strategy relativization_order relativization_marking relativizer_position relativizer_morpheme  infinitive infinitive_position	None definite, indefinite prepositional word None number suffix comparative, superlative, equative prepositional word present, past, future, imperfect suffix indicative, subjunctive, imperative, conditional suffix active, passive suffix first, second, third suffix singular, plural suffix True singular, plural suffix head-initial head-marking postpositional word postpositional word infinitive suffix

Table 12: The configuration of the Hixkaryana-like feature set. The configuration is largely based on Derbyshire (1985).

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		OSV
	oblique_word_order		XOV
	adj_noun_word_order		NA
	posspron_pron_word_order		PossN
	num_noun_word_order		NNum
	adposition_noun_word_order		NP
Morphology	case	case_marking	ergative, absolutive, genitive, instrumental
		case_marking_strategy	postpositional word
		oblique_case_marking	None
	definiteness		None
		adjective_agreement	None
	comparative	comparative	comparative, superlative
		comparative_marking_strategy	postpositional word
	tense_aspect		None
	mood		None
	voice		None
	person	person_agreement	first, second, third
		person_marking_strategy	prepositional word
		verbal_number_agreement	singular, plural
		verbal_number_marking_strategy	prepositional word
			False
	inclusive_exclusive		None
	nominal_number		head-initial
	relativization	relativization_order	dependent-marking
		relativization_marking	postpositional
		relativizer_position	affix
relativizer_morpheme		postpositional word	
negation		None	
infinitive		None	

Table 13: The configuration of the Mizo-like feature set. The configuration is largely based on [Chhange \(1989\)](#).

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		SOV
	oblique_word_order		XOV
	adj_noun_word_order		AN
	posspron_pron_word_order		PossN
	num_noun_word_order		NumN
	adposition_noun_word_order		NP
Morphology	case	case_marking	nominative, accusative, dative, genitive, ablative, locative, instrumental
		case_marking_strategy	suffix
		oblique_case_marking	genitive
	definiteness		None
		adjective_agreement	None
	comparative	comparative	comparative, superlative
		comparative_marking_strategy	prepositional word
	tense_aspect	tense_aspect	present, past, future
	mood	tense_aspect_marking_strategy	suffix
		mood	indicative, imperative, conditional
	voice	mood_marking_strategy	suffix
		voice	active, passive
	person	voice_marking_strategy	suffix
		person_agreement	first, second, third
		person_marking_strategy	suffix
		verbal_number_agreement	singular, plural
		verbal_number_marking_strategy	suffix
	inclusive_exclusive		False
	nominal_number	nominal_number	singular, plural
	relativization	nominal_number_marking_strategy	suffix
		relativization_order	head-final
		relativization_marking	dependent-marking
		relativizer_position	postpositional
relativizer_morpheme		affix	
negation		suffix	
infinitive	infinitive	infinitive	
	infinitive_position	suffix	

Table 14: The configuration of the Turkish-like feature set.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		SVO
	oblique_word_order		VOX
	adj_noun_word_order		NA
	posspron_pron_word_order		NPoss
	num_noun_word_order		NumN
	adposition_noun_word_order		PN
Morphology	case		None
	definiteness		None
	adjective_agreement		None
	comparative	comparative	comparative, superlative, equative
		comparative_marking_strategy	postpositional word
	tense_aspect		None
	mood		None
	voice		None
	person		None
	inclusive_exclusive		False
	nominal_number		None
	relativization	relativization_order	head-initial
		relativization_marking	head-marking
		relativizer_position	postpositional
		relativizer_morpheme	word
	negation		prepositional word
infinitive	infinitive	None	

Table 15: The configuration of the Vietnamese-like feature set.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		VSO
	oblique_word_order		VOX
	adj_noun_word_order		NA
	posspron_pron_word_order		NPoss
	num_noun_word_order		NumN
	adposition_noun_word_order		PN
Morphology	case		None
	definiteness	definiteness	definite
		definiteness_marking_strategy	prepositional word
		definiteness_agreement	None
	adjective_agreement		None
	comparative	comparative	comparative, superlative
		comparative_marking_strategy	suffix
	tense_aspect	tense_aspect	present, past, future
		tense_aspect_marking_strategy	suffix
	mood	mood	indicative, subjunctive, imperative, conditional
		mood_marking_strategy	suffix
	voice		None
	person	person_agreement	first, second, third
		person_marking_strategy	suffix
		verbal_number_agreement	singular, plural
		verbal_number_marking_strategy	suffix
	inclusive_exclusive		False
	nominal_number	nominal_number	singular, plural
	nominal_number_marking_strategy	suffix	
relativization	relativization_order	head-initial	
	relativization_marking	head-marking	
	relativizer_position	postpositional	
	relativizer_morpheme	word	
negation		prepositional word	
infinitive	infinitive	None	

Table 16: The configuration of the Welsh-like feature set.

Submodule	Features	Subfeatures	Value
Syntax	main_word_order		OSV
	oblique_word_order		OXV
	adj_noun_word_order		NA
	posspron_pron_word_order		NPoss
	num_noun_word_order		NNum
	adposition_noun_word_order		NP
Morphology	case	case_marking	ergative, absolutive, genitive, dative, locative, instrumental
		case_marking_strategy	prefix
	definiteness	oblique_case_marking	instrumental
		definiteness	definite, indefinite
	adjective_agreement	definiteness_marking_strategy	suffix
		adjective_agreement	number, case, definiteness
	comparative	adjective_agreement_strategy	prefix
		comparative	comparative, superlative, equative
	tense_aspect	comparative_marking_strategy	prefix
		tense_aspect	present, future, recent past, remote past
	mood	tense_aspect_marking_strategy	prefix
		mood	indicative, subjunctive, imperative, conditional
	voice	mood_marking_strategy	prefix
		voice	active, passive
	person	voice_marking_strategy	prefix
		person_agreement	first, second, third
		person_marking_strategy	suffix
		verbal_number_agreement	singular, plural, dual
	inclusive_exclusive	verbal_number_marking_strategy	prefix
		nominal_number	True
	relativization	nominal_number	singular, plural, dual
		nominal_number_marking_strategy	prefix
	negation	relativization_order	head-final
relativization_marking		head-marking	
relativizer_position		postpositional	
relativizer_morpheme		affix	
infinitive	negation	suffix	
	infinitive	infinitive	
	infinitive_position	prefix	

Table 17: The configuration of the “hard” feature set.

## **C Full results of the morphosyntax experiments**

Full details of the morphosyntax experiments are presented in [Table 18](#).

Model	Language	ICL	TER	SER	MFER	MSER	LemF1
claude-3-5-sonnet	arabic	No	75.76	0.00	34.18	17.09	93.12
	fijian	No	56.25	30.82	33.51	32.16	63.07
	french	No	98.63	15.05	36.22	25.64	88.99
	hixkaryana	No	61.86	15.73	42.60	29.17	84.23
	mizo	No	113.13	92.15	61.71	76.93	52.58
	turkish	No	67.67	33.83	20.70	27.27	91.37
	welsh	No	52.17	15.31	21.90	18.60	91.24
	vietnamese	No	75.25	31.25	48.57	39.91	58.10
	hard	No	99.26	66.18	76.16	71.17	62.87
	Average	No	77.78	33.37	41.73	37.55	76.17
gemini-2.5-flash	arabic	No	60.61	30.93	46.30	38.61	74.40
	fijian	No	22.50	0.00	28.76	14.38	83.54
	french	No	73.97	0.00	39.05	19.53	82.68
	hixkaryana	No	92.78	31.47	57.97	44.72	72.71
	mizo	No	75.42	61.43	47.36	54.40	65.91
	turkish	No	135.34	84.59	56.34	70.46	79.40
	welsh	No	52.17	0.00	32.22	16.11	90.16
	vietnamese	No	45.15	46.88	42.43	44.65	62.27
	hard	No	115.81	132.35	78.37	105.36	46.94
	Average	No	74.86	43.07	47.64	45.36	73.11
gemini-2.5-pro	arabic	No	60.61	15.46	38.82	27.14	87.89
	fijian	No	67.50	92.47	48.80	70.63	66.56
	french	No	86.30	15.05	44.39	29.72	81.99
	hixkaryana	No	123.71	47.20	50.30	48.75	73.02
	mizo	No	75.42	46.08	47.76	46.92	77.04
	turkish	No	135.34	84.59	56.30	70.45	77.65
	welsh	No	52.17	0.00	27.54	13.77	91.40
	vietnamese	No	30.10	31.25	33.85	32.55	73.74
	hard	No	115.81	215.07	80.33	147.70	40.14
	Average	No	83.00	60.80	47.57	54.18	74.38
gpt-4o-mini	arabic	No	90.91	77.32	77.70	77.51	51.66
	fijian	No	56.25	61.64	56.79	59.22	48.21
	french	No	73.97	45.15	69.12	57.13	60.17
	hixkaryana	No	108.25	94.41	62.51	78.46	55.91
	mizo	No	87.99	92.15	62.29	77.22	54.06
	turkish	No	101.50	67.67	71.06	69.36	52.06
	welsh	No	78.26	61.22	64.70	62.96	51.29
	vietnamese	No	75.25	62.50	54.31	58.40	51.63
	hard	No	115.81	82.72	81.56	82.14	50.08
	Average	No	87.58	71.64	66.67	69.16	52.79
gpt-5-mini	arabic	No	75.76	0.00	38.72	19.36	88.14
	fijian	No	67.50	92.47	61.71	77.09	46.76
	french	No	73.97	135.45	45.20	90.32	59.74
	hixkaryana	No	77.32	62.94	34.48	48.71	81.06
	mizo	No	100.56	61.43	60.64	61.04	50.46
	turkish	No	118.42	67.67	51.48	59.58	80.49
	welsh	No	39.13	0.00	21.92	10.96	89.40
	vietnamese	No	75.25	78.12	48.29	63.21	57.56
	hard	No	99.26	66.18	67.75	66.96	60.89
	Average	No	80.80	62.70	47.80	55.25	68.28
gpt-5	arabic	No	75.76	15.46	35.65	25.55	83.98
	fijian	No	67.50	77.05	60.00	68.53	45.97
	french	No	73.97	15.05	39.33	27.19	76.54
	hixkaryana	No	30.93	15.73	35.47	25.60	80.85
	mizo	No	100.56	76.79	63.14	69.97	46.99
	turkish	No	135.34	67.67	61.46	64.57	79.90
	welsh	No	78.26	45.92	48.36	47.14	74.32
	vietnamese	No	45.15	46.88	46.74	46.81	58.78
	hard	No	99.26	49.63	26.20	37.91	8.49
	Average	No	78.53	45.58	46.26	45.92	61.76

Model	Language	ICL	TER	SER	MFER	MSER	LemF1
claude-3-5-sonnet	arabic	Yes	60.61	0.00	26.57	13.28	97.48
	fijian	Yes	22.50	15.41	13.32	14.36	81.49
	french	Yes	73.97	15.05	19.54	17.30	93.85
	hixkaryana	Yes	77.32	47.20	44.70	45.95	87.32
	mizo	Yes	87.99	76.79	31.79	54.29	80.61
	turkish	Yes	50.75	16.92	10.43	13.67	96.76
	welsh	Yes	26.09	0.00	8.97	4.49	96.66
	vietnamese	Yes	15.05	15.62	18.30	16.96	84.29
	hard	Yes	82.72	49.63	43.97	46.80	83.74
	Average	Yes	55.22	26.29	24.18	25.23	89.13
gemini-2.5-flash	arabic	Yes	60.61	15.46	26.45	20.96	84.76
	fijian	Yes	11.25	0.00	17.67	8.84	87.52
	french	Yes	24.66	15.05	20.09	17.57	93.45
	hixkaryana	Yes	30.93	0.00	34.02	17.01	85.42
	mizo	Yes	37.71	0.00	17.18	8.59	90.10
	turkish	Yes	101.50	50.75	31.56	41.15	93.21
	welsh	Yes	39.13	0.00	22.42	11.21	96.99
	vietnamese	Yes	30.10	31.25	12.67	21.96	88.14
	hard	Yes	99.26	49.63	35.20	42.42	90.32
	Average	Yes	48.35	18.02	24.14	21.08	89.99
gemini-2.5-pro	arabic	Yes	30.30	0.00	13.80	6.90	95.64
	fijian	Yes	11.25	15.41	24.82	20.11	66.63
	french	Yes	24.66	0.00	8.59	4.29	96.86
	hixkaryana	Yes	92.78	31.47	29.94	30.71	82.77
	mizo	Yes	12.57	0.00	9.33	4.67	93.26
	turkish	Yes	84.59	33.83	18.36	26.10	92.68
	welsh	Yes	13.04	0.00	4.14	2.07	97.62
	vietnamese	Yes	0.00	0.00	12.74	6.37	90.55
	hard	Yes	99.26	33.09	28.65	30.87	87.64
	Average	Yes	40.94	12.64	16.71	14.68	89.30
gpt-4o-mini	arabic	Yes	90.91	77.32	79.23	78.28	50.12
	fijian	Yes	67.50	46.23	57.76	52.00	45.10
	french	Yes	73.97	45.15	59.53	52.34	72.37
	hixkaryana	Yes	108.25	94.41	63.22	78.81	54.59
	mizo	Yes	87.99	92.15	62.29	77.22	54.06
	turkish	Yes	101.50	67.67	70.23	68.95	53.08
	welsh	Yes	78.26	61.22	71.57	66.40	47.68
	vietnamese	Yes	75.25	62.50	52.37	57.43	53.57
	hard	Yes	115.81	82.72	81.03	81.88	50.83
	Average	Yes	88.83	69.93	66.36	68.14	53.49
gpt-5-mini	arabic	Yes	45.45	0.00	25.46	12.73	91.07
	fijian	Yes	101.25	138.70	69.42	104.06	37.02
	french	Yes	61.64	90.30	38.50	64.40	66.20
	hixkaryana	Yes	108.25	47.20	45.44	46.32	79.01
	mizo	Yes	125.70	92.15	64.11	78.13	46.30
	turkish	Yes	118.42	84.59	51.38	67.98	80.18
	welsh	Yes	26.09	0.00	17.76	8.88	92.07
	vietnamese	Yes	60.20	62.50	47.23	54.86	57.51
	hard	Yes	115.81	49.63	54.81	52.22	72.80
	Average	Yes	84.76	62.79	46.01	54.40	69.13
gpt-5	arabic	Yes	60.61	15.46	26.46	20.96	91.59
	fijian	Yes	33.75	15.41	38.24	26.83	66.67
	french	Yes	24.66	0.00	10.38	5.19	96.10
	hixkaryana	Yes	46.39	31.47	33.82	32.64	82.19
	mizo	Yes	37.71	15.36	25.53	20.45	79.47
	turkish	Yes	118.42	84.59	51.27	67.93	79.38
	welsh	Yes	52.17	45.92	38.47	42.19	80.22
	vietnamese	Yes	45.15	46.88	28.77	37.82	78.17
	hard	Yes	99.26	66.18	52.20	59.19	65.35
	Average	Yes	57.57	35.70	33.90	34.80	79.90

Table 18: Full results of the Morphosyntax module. The left table reports the performance without the “review” stage for few-shot in-context learning, and the right table reports the performance with the “review” stage.